Tomographic Medical Image Reconstruction with Deep Learning

User Manual

Asher Burrell, Christopher Hinton, & Ty Mercer

This user manual is intended for use within Dr. Mitra's BiCLab. It will be updated when/if a public release of the code is made available.

Table of Contents

Introduction

This document contains instructions on how to run the Data Generation and Reconstruction AI modules of the Tomographic Medical Image Reconstruction with Deep Learning project. It is mainly intended for use within Dr. Mitra's BiCLab; However, the User Manual section, which describes a hypothetical public release of the program (which is unlikely to occur, save for XCAT+), has been released as per the Senior Design class requirements.

This project is split into two modules: Data Generation and Reconstruction AI. Data Generation is focused on the creation of synthetic SPECT images via XCAT+ as well as the corresponding sinograms. It consists of programs designed to create XCAT phantoms (Requires license, which is not included with this code), turn those XCAT phantoms into XCAT+, and run GATE (requires installation, not included with this code) to obtain the corresponding sinograms. Reconstruction AI consists of a CNN designed to generate a synthetic SPECT image based on an input sinogram. It can

run on real or synthetic SPECT data, although our experiments have yielded much better results with our artificial data, as this type of data was used in training.

Note that all file dimensions listed in this guide are in terms of 32-bit real values, not bytes. To find the size in bytes, multiply each dimension by 4. Most user-facing programs other than XCAT+ are only compatible with Linux.

User Manual

This section includes all code that a user is expected to run, as a part of this system. It is divided into Data Generation and Reconstruction AI, based on which part of the system the user is interacting with.

Data Generation

These are the programs you may want to run in order to use the data generation portion of the system. The goal of the data generation system is to create large amounts of realistic synthetic SPECT data. This data was used within our project to train our Reconstruction AI, but may have other uses as well.

XCAT+

Input: XCAT phantom (or other binary image); Parameters specifying patient type; Optional parameters to change default values of organ voxels, allow for different sized input, or to turn off Gaussian blur. The input phantom can be any dimensions, but if it cannot be viewed as a coherent series of 128x128 slices the size parameter will need to be modified.

Output: Synthetic SPECT image (XCAT+ phantom). Dimensions of the output image are the same as the input.

XCAT+ is a program designed for use with XCAT (eXtended CArdiac Torso) phantoms, although it is compatible with any binary image. Its purpose is to generate synthetic SPECT images of certain organs (currently the heart and liver)

from these XCAT phantoms. The code for this program, and more extensive documentation, can be found at: https://github.com/DM-BiC-Lab/XCATplus.

XCAT+ works by replacing every voxel in the input phantom that corresponds to a target organ with an appropriate tracer value. These tracer values are generated by statistical functions, which were found by analyzing a minimum of ten real patient SPECT images for each patient type: Control (non-diseased) Rest, Control Stress, Diseased Rest, or Diseased Stress. The patient type to use is specified by the user. Organs that are supported by XCAT+ at time or writing at myocardium (heart tissue), left blood pool, right blood pool, combined blood pool (uses stats from both blood pools, for phantoms where they are not differentiated), and liver. A Gaussian blur is also applied to the output image, in order to smoothen the values.

Sinogram Simulator

Input: XCAT phantom, number of sinograms to generate. XCAT phantom must conform to the default values described at https://github.com/DM-BiC-Lab/XCATplus?tab=readme-ov-file#default-values.

Output: A folder containing the following, under the Sinogram Simulator/Output: one subfolder, which contains an individual folder for each generated sinogram, as a 128x128x240 raw binary file and a small information file; The original XCAT or XCAT+ phantom used for data generation; expected_reconstruction.raw, which is the XCAT phantom that was either used as input or generated from the XCAT, cropped around the region of interest to a 128x128x128 binary image; The parameters file, attenuation map, or log file from XCAT creation, if they are in the same folder as the XCAT; And a log of all Sinogram Simulator runs.

The Sinogram Simulator is a program that, given a synthetic SPECT image such as the ones generated by XCAT+, will run a physics simulation to generate raw data that corresponds to that image. This raw data is called a sinogram.

The Sinogram Simulator has XCAT+ built into it, so it should be used independently on an XCAT phantom. However, it does not allow the modification of XCAT+ parameters; Your XCAT phantom must conform to the default values listed at

https://github.com/DM-BiC-Lab/XCATplus?tab=readme-ov-file#default-values. This built-in XCAT+ does not apply Gaussian blur, and will use control-rest patient statistics by default. Stress or diseased statistics will be used instead if "stress" or "diseased" is in the XCAT phantom filename.

To run the Sinogram Simulator, users must first install GATE on their computer. This is the physics simulator used to simulate the SPECT imaging process.

The Sinogram Simulator can be run by navigating to the Sinogram Simulator folder, pasting the input file, and running the command "python run_simulation.py [xcat name] [number to generate]" (do not include quotes here when typing in the command line). This input file can also be placed in Inputs. [xcat name] should be replaced with the filename of your XCAT phantom. [number to generate] should be a positive integer indicating how many sinograms to generate. Do not include a file path here. [number to generate] may be omitted, and has a default value of 1.

The Sinogram Simulator is currently only compatible with Linux. Expect this program to take a minimum of 12 hours per sinogram, depending on the processing capacity of your computer. Do not attempt to run this simulation on a laptop. Do not modify the contents of Sinogram Simulator or any of its subfolders while the simulation is running. Do not run more than one Sinogram Simulator program at a time. Running programs that require a lot of computing power while the simulator is running is not advised. Do not modify the provided contents of Sinogram Simulator unless you know what you are doing and have read the documentation under Developer Manual.

Most programs that are included in the Sinogram Simulator folder by default are dependencies of this program. They are documented in the Developer Manual.

XCAT Generator

Input: Default Organ ID Parameters file, number of XCATs to generate

Output: One XCAT phantom, and corresponding attenuation map, log, and Organ ID Params files.

This program will generate an XCAT parameters file using known parameters, then run XCAT using those parameters to create an XCAT phantom. Variation in parameters is controlled such that it resembles normal variation in human cardiac activity. Non-heart-related variables are not modified.

This program requires you to be able to run XCAT and is only compatible with Linux. XCAT is proprietary to Duke University and is not included with this code. You will need to paste the XCAT_V2_Linux folder into Sinogram Simulator; This should include your XCAT executable, license, default .nrb files, and all other XCAT dependencies. If a new version of XCAT is released, this program may become deprecated. You must also extract all other files in the Sinogram Simulator XCAT Extension folder to Sinogram Simulator.

Once these conditions are met, you can run this program with the command, "python xcat_generator.py".

Automatic Simulation

Input: Default Organ ID Params file, number of times to run the simulation on each XCAT generated. This is 1 by default.

Output: Outputs of XCAT Generator and Sinogram Simulator, repeated every time Sinogram Simulator finishes running.

In order to maintain a continuous stream of data generation, we have included a program, auto_sim.py, that will continuously create new XCATs and run the Sinogram Simulator on them. Because it is dependent on XCAT, this program is also included in Sinogram Simulator XCAT Extension.

Automatic Simulation is a continuously looping program that runs the XCAT Generator and Sinogram Simulator in sequence. It will continue to generate XCAT phantoms and run the Sinogram Simulator on them until it is manually terminated.

All requirements and restrictions for XCAT Generator and Sinogram Simulator also apply to this program.

This program can be run with the command "python auto_sim.py [number to generate]". [number to generate] can be omitted (in which case it will default to 1), or replaced with the number of sinograms to generate from each XCAT. Running this program with commands such as nohup that prevent termination via Ctrl-C is not recommended, as this will require you to stop the program via the task manager.

Sinogram Augmenter

Input: Outputs of Sinogram Simulator: Input folder containing One or more sinograms (128x128x240 raw binary) and expected reconstruction (128x128x128, raw binary); path to input folder (if it is not in Sinogram Simulator/Output). These files should be in the location they were in following the execution of the Sinogram Simulator, or else in the same format/folder layout: The input folder must contain "expected_reconstruction.raw" and exactly 1 subfolder not named "augmentation", which contains any number of subfolders, each of which contains one .bin file (the sinogram; Non-.bin files are ignored).

Output: A number of input-output pairs, equal to the number of input sinograms times 55. Inputs are sinograms (128x128x120 raw binary), outputs are synthetic SPECT images (same as input).

This program is used to augment the synthetic data generated by the Sinogram Simulator, so that more data is available. This data was primarily used to train the Reconstruction AI.

The Sinogram Augmenter folder should be placed in the same directory as Sinogram Simulator. It should not be placed inside Sinogram Simulator.

This program can be run with the command, "python augment.py [folder_name] [folder_path]". [folder_name] should be replaced with the name of the folder in Sinogram Simulator/Output, or the other folder with the same structure, where the input data is stored. [folder_path] should usually be omitted; If you want to access a folder that is not in Sinogram Simulator/Output, you can include the path to that

folder (not including the folder itself) here. This is not recommended. Typing -all instead of [folder_name] will cause the Sinogram Augmenter to run on all folders in that directory, where the appropriate data can be found.

Output from this program will be stored under Augmented Data, in the Sinogram Augmenter folder.

This program alters the sinograms in the following ways in order to augment them:

- Resizes ("squashes") them from 128x128x240 to 128x128x120
- Rotates them at 5 angles (-10, -5, 0, 5, and 10 degrees) on the Z axis
    - Outputs 5 sinograms from the 1 input.
- Shifts them 10 times on the XY axis using a forward project → shift → back project algorithm
    - Applied to each result of rotation; Original and 10 shifted sinograms are kept per input to this step.

Sinogram Processor

Input: Outputs of Sinogram Simulator: One or more sinograms (128x128x256 raw binary) and expected reconstruction (128x128x128, raw binary); path to input folder (if it is not in Sinogram Simulator/Output). These files should be in the location they were in following the execution of the Sinogram Simulator, or else in the same format/folder layout.

Output: A number of input-output pairs, equal to the number of input sinograms. Inputs are sinograms, outputs are synthetic SPECT images.

This program is similar to the Sinogram Augmenter, but it does not create additional sinograms; It only outputs modified versions of the original input. All documentation is the same as Sinogram Augmenter, with the following exceptions:

- This program is in the Sinogram Processor folder, not Sinogram Augmenter.
- This program is called preprocess.py instead of augment.py.
- Output is stored in Processed Data, in the Sinogram Processor folder.
- All augmentations after "squashing" the data are omitted.

Reconstruction AI

The Reconstruction AI is a convolutional neural network (CNN) that is trained to take a sinogram as input and generate an approximation of the SPECT image that could be reconstructed from that sinogram. It is moderately successful at this task in its current iteration, but should not be relied on for medical purposes.

Reconstruction AI

Input: Sinogram (128x128x120, raw binary file). May be real or synthetic data.

Output: Reconstructed SPECT image (128x128x64, raw binary). Generated by the AI, not 100% accurate.

The reconstruction AI consists of a python program, xcat_ceda_pytorch.py, which can be called with the command "python ai.py [input]", where [input] is the input sinogram file. The output image will be generated in the same directory as ai.py. This program also uses a parameters file, ai_params.par, which represents the best result from our AI training; It is used to easily load the trained AI into ai.py.

Developer's Manual

This section includes details of code for the Tomographic Medical Image Reconstruction with Deep Learning project that a user should not have to interact with, but would still have access to.

Sinogram Simulator

Code for the Sinogram Simulator can be found in the "src" and "GATE Simulation" folders. Almost all of it was created by a former lab member, Tommy Galletta, not

our Senior Design group. We have mostly been running this code as a black box, however, we have included a basic overview of what each file does here.

SRC

This folder contains Python source code to run the GATE simulation. A brief description of each file, and what (if anything) you may need to update, can be found below.

XCAT_crop_into_GATE: This is the most likely file to be modified in src. It takes the XCAT phantom that is used as input, crops it to the area around the heart, applies the XCAT+ process to the XCAT phantom, and saves the XCAT phantom for use as input to the GATE simulation. Cropped XCAT and XCAT+ phantoms are saved to the GATE Simulation folder through this program.

Combine_parallel_outputs: The GATE simulation is actually a number of GATE processes run in parallel, each of which has its own output. This program combines these outputs into the final sinogram (combined_parallel_results.bin) and a header file that contains relevant metadata.

Fix_mhd: Updates the combined parallel output header file to include information about the sinogram (results) file.

Produce_macro_files: Creates macros to run the GATE simulation, using custom parameters based on the input variables and current time.

Produce_mhd_files: Creates mhd files that are used by the GATE simulation, using the parameters for the input.

GATE Simulation

This folder contains macros and other files that are used to run the GATE simulation. Generally speaking, these are called by each other or by

multiple_runs.py during normal operation. A brief description of each file, and what (if anything) is likely to be changed, can be found below.

XCAT Materials: A text file showing the values for different organs/body parts in the input (non-XCAT+) phantom. Formatted as:

Min Max Organ

for each line, where min is the min value that corresponds to the organ (inclusive), max is the max value (exclusive), and Organ is the organ name. These values should be changed to match that of your Organ ID parameters file. An example (that is consistent with the default values used throughout this project) is shown below:

```
1 0          200          G4_AIR
2 200        300      Lung
3 300        400      Body
4 400        500      Intestine
5 500        600      BoneMarrow
6 600        700      Pancreas
7 700    800          Brain
8 800    900          Heart
9 900    1000     Kidney
0 1000   1100     Blood
1 1100   1200     Liver
2 1200   1300     Spleen
3 1400   1500     SpineBone
4 1500   1600     Skull
5 1600   1700     Cortical
6 1700   1800     RibBone
```

SimVisu: A macro to show a visualization of the GATE simulation. Useful to make sure any changes have not caused anything to break.

SimAutoRun: A macro to set up and run the simulation. To change the XY dimensions of the sinogram outputs (currently 128x128), change the pixelNumberX and pixelNumberY variables.

setActivityPosAndScale.mac: Generated by produce_macro_files.py. Sets the scale and position of the GATE simulation to the appropriate values.

result_n.sin, result_n.hdr: Generated for n=0 to n=23. Results of the simulation running in 24 parallel processes, and combined into the final output by combine_parallel_outputs.py.

PopulateScene.mac: A macro to set up the GATE simulation. We have not experimented with modifying this file, but if you wanted to modify the GATE simulation, you would do so here.

parallel: A folder containing 24 macros to run the GATE simulation in 24 parallel processes.

MoveVisu.mac: A macro to advance the simulation at 16 37.5 second intervals. 37.5 seconds is simulation time, not real time. Originally published by OpenGATE.

GateMaterials.db: Contains information about elements and materials simulated in GATE, such as density and composition. Formatting is as follows:
[Elements] // list of elements, with S (Symbol), Z (Atomic #), and A (Atomic mass)
Hydrogen: S= H ; Z= 1. ; A= 1.01 g/mole
Repeated for elements He-Ge, Y, Mo, Ag, Cd, Sn, Te, I, Cs, Gd, Lu, W, Au, Tl, Pb, Bi, U
[Materials] // list of materials, with d (density), n (unknown), state, & composition. Composition is relative, by weight.
Carbon: d=1.8 g/cm3; n=1; state=solid
    +el: name=auto; n=1
DielectricOil: d=0.885 g/cm3; n=3; state=liquid
    +el: name=Carbon; n=12
    +el: name=Hydrogen; n=8
    +el: name=Chlorine; n=2
WaterS: d=1.00 g/cm3 ; n=2
    +el: name=Hydrogen ; f=0.112
    +el: name=Oxygen ; f=0.888
Includes materials used in the detector head (Glass, LuYAP-80, etc) and in a human body (blood, fat, muscle, etc).

cropped_XCAT_activity.mhd: Gives information about the cropped XCAT activity file (XCAT+), including image dimensions.

cropped_xcat.mhd: Gives information about the cropped XCAT file, including image dimensions.

cropped_XCAT_activity.bin: XCAT+ used as input for the simulation. Raw binary image, same dimensions/cropping as the original cropped XCAT.

cropped_XCAT.bin: XCAT phantom input, cropped on the horizontal (transverse) plane to 64 slices centered on the heart, and on the other planes to the edge of the non-zero values in the XCAT phantom. Raw binary image, size ?x?x64 (dimensions are 72x58 for out input XCAT phantoms, but phantoms generated with a different base will have different dimensions). Size information can be found in cropped_xcat.mhd. Used to determine interference from body tissues in GATE simulation.

Miscellaneous

These programs are included in the Sinogram Simulator folder, and are used to run the GATE simulation.

Parsing: Contains helper methods for file parsing.

Multiple_runs: Calls the appropriate bash scripts to run the GATE simulation. Accepts one argument, which is the number of times to run the simulation. Assumes the input XCAT is in Input/gens, as is the case when this program is called from run_simulation.py.