Sneha Nagpaul
24th January 2018

First Part: Parse the DICOM images and Contour Files

Using the functions for parsing dicom files, i-contour files and changing them to masks, a pipeline is built making sure to pair the correct DICOM image(s) with the correct contour file. A matching mask that pertains to the relevant i-contour file is coupled with the image.

As expected of real data, I found there was a mismatch between dicom and contour files for each patient. So, I extracted their numbers from the file names and did a set intersection. Subsequently, I regenerated the file names from the numbers for dicom/contour pairs.

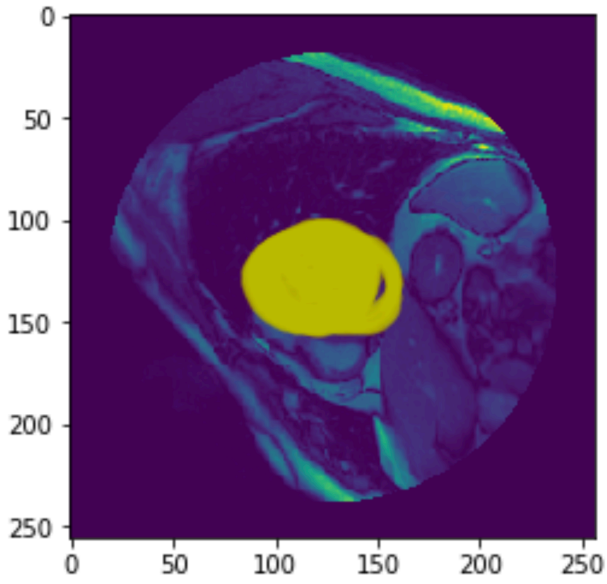
Once the pairing was found, I parsed using the provided functions, worked out a display for overlaying the image and 'mask' (Since I couldn't get PIL to run, I only plotted the contour co-ordinates as a scatterplot on the dicom image). I left the code to make the mask in the code as a given, but I did not test it.

After the pipeline was built:

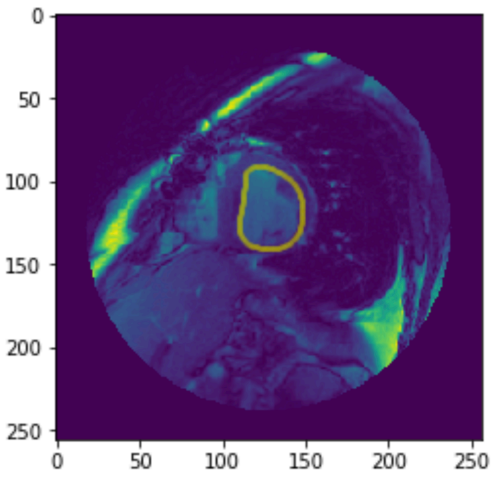
q1: How then to know if the contours were being parsed correctly?

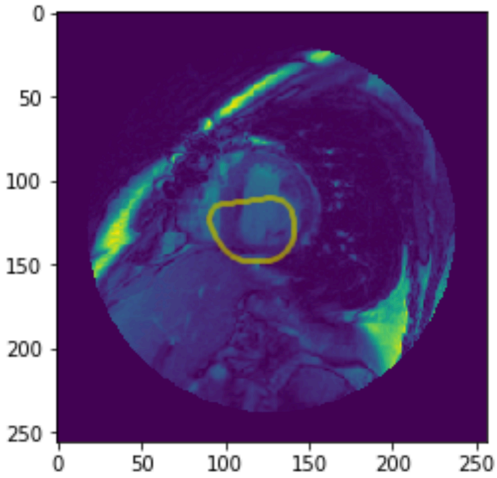
- To tackle this I slightly consulted a couple of research papers about heart scan segmentation (https://pdfs.semanticscholar.org/6f5a/98002de3cdcb25c1e63627719a8614b3472e.pdf).

- Visualized collectively by plotting all contours in one image. The image should fill up in a certain area of signaling various levels of the contours in the slices.



- Visualized separately with flipped x,y co-ordinates. The pictures didn’t line up visually. The picture on the left(produced using the given contour parser) is more indicative of a contour than the one that goes across a few shapes on the right(produced by flipping x and y co-ordinates on the parser).





- Would have asked for more information but I started late and ran out of time for that.

q2: What changes were made to the code for parsing, if any, in order to integrate it into a production code base?

- Probably doesn't need OOPs because the functions don't have to maintain an internal state.

- Made a minor change in returning image instead of dictionary as the rest of the dicom metadata wasn't in play.

Second Part: Model training pipeline

Using the saved information from the DICOM images and contour files, adding a step to the pipeline that will load batches of data for input into a 2D deep learning model. This pipeline should:

Cycle over the entire dataset, loading a single batch (e.g. 8 observations) of inputs (DICOM image data) and targets (boolean masks) at each training step.

A single batch of data consists of one numpy array for images and one numpy array for targets.

Within each epoch (e.g. iteration over all studies once), samples from a batch should be loaded randomly from the entire dataset.

Once the second part of the pipeline was built:
q1: What to change, if anything, from the pipelines built in Part 1 to better streamline the pipeline built in Part 2? If so, what? If not, is there anything that can be changed in the future?

- *The second part requires the use of a generator perhaps vs the first part which is a straightforward reading of all possible dicom/contour combinations.*
- *I'm not sure how to incorporate this functionality in high level model API's. If I had more time, I'd probably work with some of those classes to write model classes that do this as part of a class method or via a parameter to fit().*

q2: How to verify that the pipeline was working correctly?

- *I looked at a few instances manually and wrote a test to compare shapes of returned np arrays.*
- *loaded up the modules in python and loaded files using objects.*

q3: Given the pipeline you have built, are there any deficiencies or possible changes to be made given more time? If not, can there be further improvements/enhancements to the pipeline that can be built in?

- *There is some amount of hard coding that I'd like to change(URLs, image size etc).*
- *I am still unsure of the rest of the eco-system. It would definitely be more custom had I had the time to ask those questions.*
- *Write more comprehensive tests.*
- *Check if all the parameters passed into functions were viable. Handle more edge cases.*
- *I did some integration testing on the command line where I loaded modules and explored objects. I would have liked to clean up that code and add it to this repository.*
- *Would have included more comprehensive function contracts in the docstrings.*