Challenge Report

Pablo F. Ordóñez

August 18, 2017

1 Part 1: Parse the DICOM images and Contour Files

1.1 Questions

- How did you verify that you are parsing the contours correctly?

  The fastest way is to make a blended image using the MRI and Mask images . If the borders of the polygon mask match the cardiac cavity in the MRI, then one can confirm that the parse was correct. A different approach is to use another segmentation algorithm and match the segmented cardiac cavity with the polygon mask.

  In Fig.1 the bluish image is the polygon mask on top of the MRI image.

- What changes did you make to the code, if any, in order to integrate it into our production code base?

  The function $parse_{dicomfile}$ returns a Dictionary with one key and a value. For this specific challenge, the dictionary is under-used, so returning the sole element as an array is enough. For the 3rd reconstruction, more information of the DICOM file is needed and the Dictionary will be helpful to contain this data.

  Some of the pictures have Hounsfield values bigger than 1000 which makes the visualization of the pictures more difficult. If no contrast is used in the study, most values higher than one thousand correspond to the bone. Therefore, downgrading those values to one thousand in order to improve the quality of the image is done.

Fig.2 shows MRI including high Hounsfield values
After downgrading the Hounsfield value that was above one thousand is showed in Fig.3

2 Part 2: Model training pipeline

2.1 Questions

- Did you change anything from the pipelines built in Parts 1 to better streamline the pipeline built in Part 2? If so, what? If not, is there anything that you can imagine changing in the future?

  The parsing file is well written and has the basic functions needed to read the necessary files. Perhaps using preprocessing to increase the quality of the images for annotations can be done. In my thesis work, it was proven that preprocessing decreases the training time.

- How do you/did you verify that the pipeline was working correctly?

  Before flattening and appending the images to build the input array, I displayed the blended images out in order to look for matching images between the cardiac cavity and the polygon mask. To prove randomness and use of all the pictures I displayed the shape of the array and the number of files in the cache table on the console. This table has shuffled sequence numbers from 0 to the total number of paired images and contour masks, where each number points to a specific path.

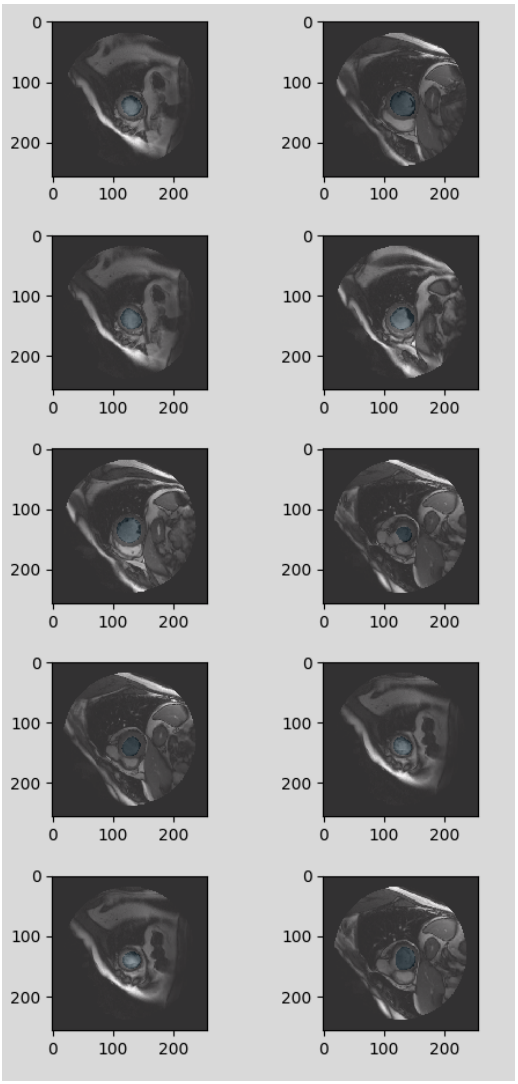

Figure 1: Blended Image from MRI & Mask Images

- Given the pipeline you have built, can you see any deficiencies that you would change if you had more time? If not, can you think of any improvements/enhancements to the pipeline that you could build in?

  Metrics and log files, I think that most of the frameworks lack metric functions not only for time but for other aspects such as memory and the power consumed. In our work, we tracked the images provided to the specific batch and the loss function for that batch.

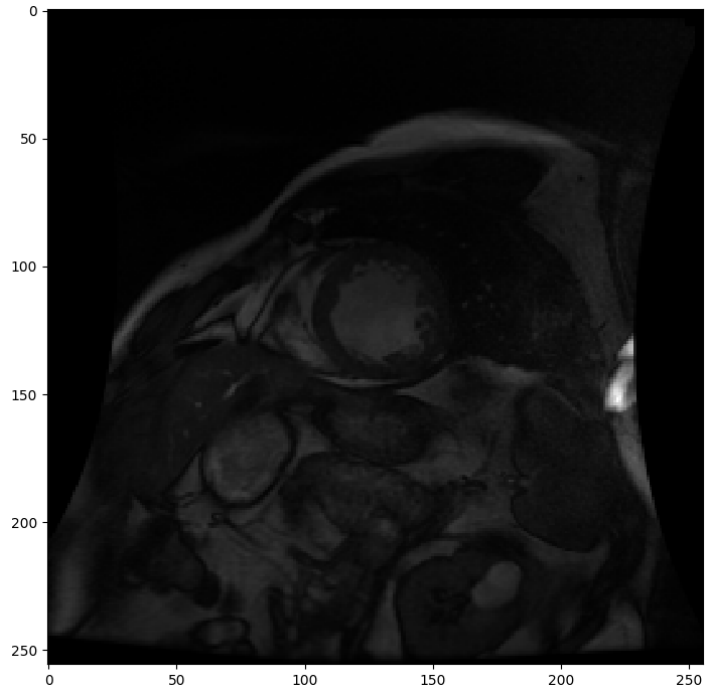

Figure 2: MRI with high Hounsfield value

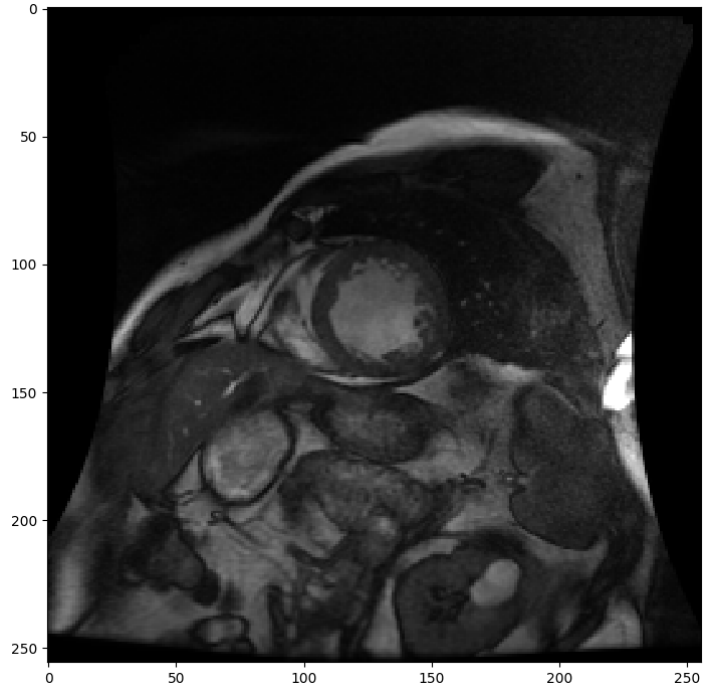

Figure 3: MRI without high Hounsfield value