The questions in the challenge are answered as follows:

**Part 1**: Part 1: Parse the DICOM images and Contour Files

**1.1.** How did you verify that you are parsing the contours correctly?

I visualized the manually annotated polygon on the contour plot of the image found by the function ([match\_dict\_2\_contour).](https://github.com/wmmxk/Quality_control_segmentation/blob/master/src/libs/match_dic_2_contour.py) Given a set of dic files and an annotation file, the true annotated polygon was compared to the contour plot for the image in each dic file. In specific, the number of pixels which are not zero on the polygon is used as the score evaluating the overlap between the true mask and the contour generated by cv2. The dic file whose contour overlaps most with the polygon is selected as the image corresponding to the mask. The contour generated by cv2 is fairly accurate but when comparing the true polygon to the contour, my solution does not always generate the correct answer. The current approach does not consider the overall overlapping between two curves appropriately. I was planning to search around how to evaluate the similarity between two curves as an inappropriate evaluation method does not work even though the contour generic by cv2 is accurate.

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

I moved the code for parsing contour and dicom file to the folder: src/.libs/helper. In the code, I added two lines to get ints for each coordinate in the polygon.

**Part 2:** Model training pipeline

**2.1.** 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 pipeline built in Part1 was kept for Part 2 for now. In the future, more computation will be conducted given the amount of work for loading data. Currently each DICOM file was read in multiple times because it is mapped to all the contour files of a patient and the searching is based on contour file.

**2.2.** How do you/did you verify that the pipeline was working correctly?

The performance of the generator pipeline depends on how to evaluate the overlap between the polygon and the contour generated by cv2, and the unique code in train\_generator module, which involves reading in images and lining them up in batches. I verified these two parts of my pipeline separately. First the train\_generator function selected the same file of DICOM type given a contour file as the match\_dic\_2\_contour itself. With this validated, I outputted the shape of the images in each batch for sanity check.

**2.3.** 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?

One deficiency is the image files are processed multiple times for each contour file given a patient ID. The other one is the image and mask are stored in different place on the disk, which slows down the reading process. To address these drawbacks, I propose to pre-process the DICOM files and contour files and save each pair of image and mask into one file. The last deficiency is that the pipeline does not allow multiple cpus to read in data.