## Data preprocessing for segmentation

Given the provided data, my first idea was to do segmentation for each slice. After segmenting, I got each artery of interest.

There are totally 25 cases. In each case, there are from 640 to 720 slices. Each slice may have more than one artery of interest. First, I parsed these dcm files to get images. Each slice is one image. For each slice, its height ranges from 100 to 160, and its width ranges from 640 to 720. I padded each image to with zeros to fit the coordinate system. Second, for each slice, I parsed four corresponding QVS files to find available contours. One slice may have more than one contours, and each contour has two series of coordinates, corresponding to lumen and outer wall separately. By applying ploygon2mask in the library of "skimage.draw" on the contours of each image, I got a mask image for each image. On each mask image, the value of pixels inside each of the outer wall contours are 1, while the rest are zero. The reason why I used outer wall contour is to make the model take the vessel wall pattern into consideration. Otherwise, the situation shown in Fig. 1 happened. Third, I cropped each image and its mask image to . As the same time, all the y-axis minus 280 ( ) to let the coordinates to fit the cropped images. Fourth, for each image, got the coordinate of the center of the lumen coordinates (as required in the assignment). As a result, each output of my data loader for segmentation has three components: image tensor, mask tensor and a list of coordinates of centers of arteries of interest. For training process, the mask was the target. For validation/testing process, mask and centers were ground truths for evaluation. Fifth, before outputting, each image went though some typical data augmentation steps, including random rotation, random flip, and normalization, which are provided by "torchvision" library. Data preprocessing for object location

|  |
| --- |
|  |
| Fig. 1(a) Original image |
|  |
| Fig. 1(b) Wrongly predicted mask |
|  |
| Fig. 1(c) Ground truth mask from lumen contour |
| Fig. 1 Bad example for segmentation. Because I used the mask from lumen contour, model wrongly ignored the vessel walls and near surrounding. |

After I did segmentation, I found results not satisfying. I realized that segmenting slices directly was too challenging. So, I turn to help of object location method. The data preprocessing was almost the same with that for segmenting. The only difference was that the target for training one image is no longer a mask. Instead, the target is a list of coordinates of centers of arteries of interest and the corresponding width and height. In this experiment, I doubled the width and height for each artery of interest, so that the model not only considered the pattern inside the artery, but also its near surroundings.

## Methods and experiment

For segmentation, I tried two methods. One is the famous Unet [1]. I downloaded one implementation from Github and adjusted it to my experiment. The other is a self-designed ResNet-Unet. The idea is to replace the upsampling operations with Unet with 2D transposed convolutions and add shortcut connections in each block. According to this experiment, this neural network is as good as Unet and saved a lot of time and space. The loss functions are binary cross entropy loss and dice loss (the negative of dice index). Both models output a mask of the same shape of input image. An "imantics" library was used to transfer the output mask to contours, so that I in turn got the coordinates of predicted centers.

For object location, I used YOLO v3 [2]. I downloaded the implementation from Github adjusted it to my data and evaluation.

## Evaluation

For segmentation, two evaluation metrics were used: dice index and average Euclidean distance. For object location, only average Euclidean distance was used. The first 20 cases were used as training/validation set, while the last 5 cases were testing set. The validation set was from a randomized case from the first 20 cases.

|  |
| --- |
|  |
| Fig. 2 (a) Original Image |
|  |
| Fig. 2(b) Correctly predicted mask |
| Fig. 2 One good example of Res18-Unet for segmentation |

One problem for the average Euclidean distance evaluation was that the predicted locations of my model do not always have one-to-one to the ground truth. For example, on an image, my model could predict four locations of centers of arteries, while this image's ground truth has only three centers of arteries. In this case, we need to find a match for each predicted location so that we can compute the distance. The match algorithm is shown in Algorithm 1.

All the experiments were conducted on Titan RTX (24 GB), on Windows 10 system. Implementation for this experiment is available at <https://github.com/SarielMa/wall_vessel_segmentation>.

## 

## Results

The evaluation result for segmentation is shown in Table. 1 and one sample is shown in Fig. 2. Because the mean distance metric only takes into consideration those matched predictions, we must consider those predictions occurring in complete wrong locations. "discovers" means how many locations of predictions were output by the models. "objects" means the number of locations should be detected by the models. "detected" means the number of locations that have been successfully detected (matched) by models. In this way, the ratio, , is the precision of the models.

For the object location, all the metrics are the same. The result is shown in Table.2 and one sample is in Fig. 3. Because the output of YOLO v3 are strongly controlled by the confidence threshold ("conf\_thred" in Table. 2), I tried several thresholds. As we can see, as the threshold grows up, the precision is in general growing as well. However, the mean distance is becoming bad. This is because higher threshold will omit more predicted locations, sometimes omit correctly predicted locations.

## Conclusion

According to the experiments, my models can detect almost all the arteries of interest. And in general, the predicted locations are satisfying (mostly, the Euclidean distance is less than 10 pixels). Comparing with segmentation methods, object location methods tend to have high precision while can detect the same number of object and have the same mean Euclidean distance. However, while detecting almost all the arteries of interest, my model also predicted many meaningless locations. Also, as for the segmentation task, the dice index is not satisfying (only 54%), which means the segmentation itself is not successful. I think what is the most challenging for this data set is not only its very small targets, but also too many similar patterns in the image (some parts of the images look very like vessels but are not). Treating very slices as independent images will not improve the performance too much. For the next step, I will dip into the data and try to find some deep information. For example, target arteries of the same type on adjacent slices should be very close to each other. I think I can leverage this to track where the vessel goes to improve the location precision. Also, by combining with location techniques, the performance of segmentation will also be improved. Due to very limited time for this week, these ideas will be conduct in the future. Thanks for your patience.

Table. 1 Result of segmentation methods

|  |  |  |
| --- | --- | --- |
|  | Res18-Unet | Unet |
| mean Euclidean distance | 8.04 | 8.6 |
| mean dice index | 0.54 | 0.54 |
| discovers | 1400 | 1252 |
| detected | 517 | 516 |
| objects | 518 | 518 |
| precision | 0.37 | 0.41 |

Table. 2 Result of object location methods (YOLO v3)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| conf\_thred | 0.05 | 0.15 | 0.25 | 0.35 |
| mean Euclidean distance | 6.25 | 9 | 10.36 | 12.85 |
| discovers | 1334 | 1060 | 940 | 843 |
| detected | 518 | 516 | 507 | 497 |
| objects | 518 | 518 | 518 | 518 |
| precision | 0.38 | 0.48 | 0.54 | 0.59 |

|  |
| --- |
|  |
| Fig. 3 One good example of YOLO v3 (Conf-thred = 0.25) |

## Reference

[1] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[2] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." In International Conference on Medical image computing and computer-assisted intervention, pp. 234-241. Springer, Cham, 2015.