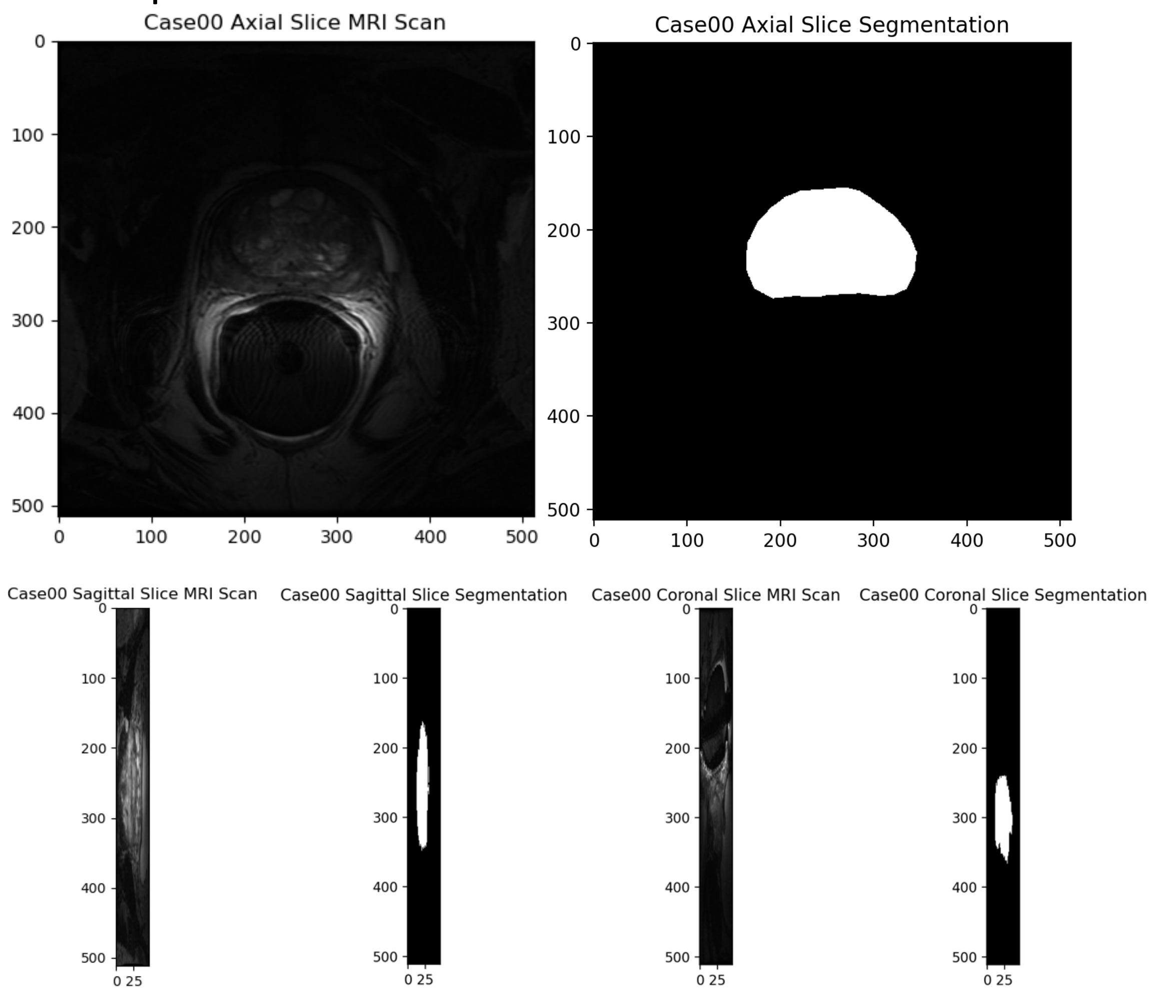
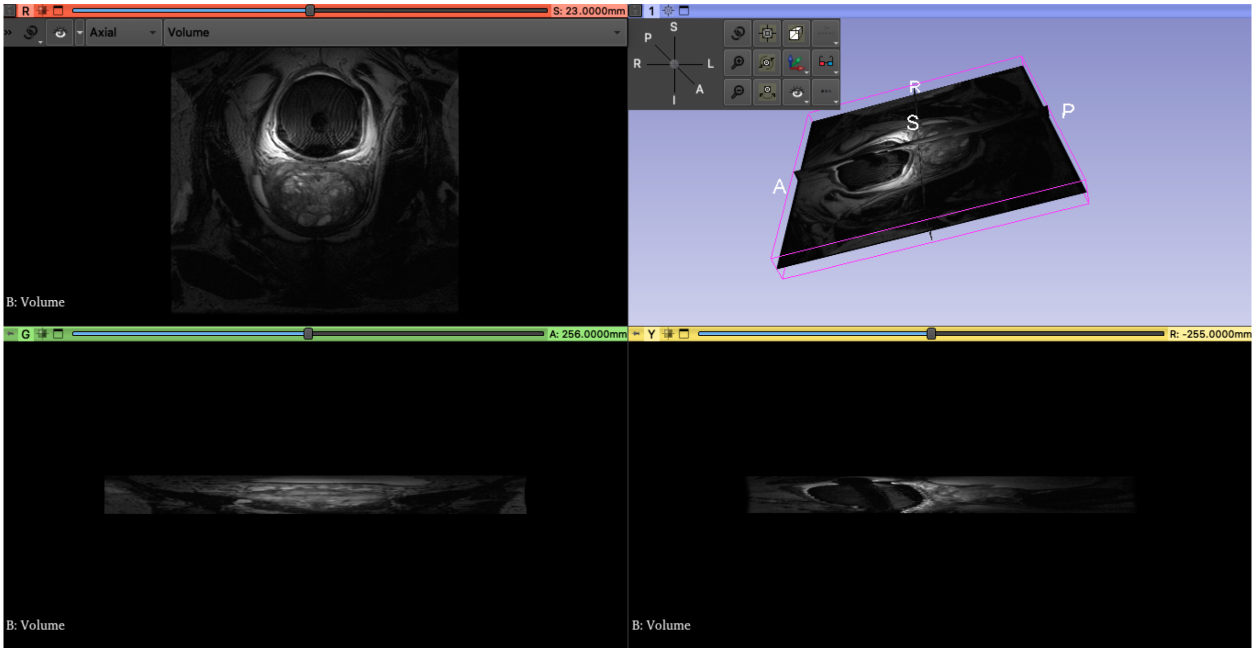
CISC/CMPE 472 – Assignment 3
Ngoc Bao Han, Nguyen (Mimi)
Mar 17th, 2023

1. Data exploration



Upon segmenting the center slice from the axial, sagittal, and coronal view from Case00, I can observe and get a better understanding of where the prostate is in the original image. A visualization of these 2 volumes in Slicer is shown below:



An immediate observation from viewing Case00 is that the Axial slice is wider and looks square compared to the Sagittal and Coronal views. I print out the dimension of the volume and I got (47, 512, 512).

I printed out a set of all different volume dimensions from the folder. This is the result of the set:
{(39, 512, 512), (15, 320, 320), (46, 512, 512), (34, 512, 512), (28, 384, 384), (28, 320, 320), (26, 512, 512), (18, 256, 256), (24, 320, 320), (20, 320, 320), (45, 512, 512), (29, 512, 512), (23, 512, 512), (47, 512, 512), (54, 512, 512), (17, 320, 320), (23, 256, 256), (42, 512, 512), (40, 512, 512)}
There is a good variation of:

- Axial depth: [15, 54]
- Sagittal depth: [256, 512]
- Coronal depth: [256, 512]

All volumes seem to have all 3 slice views.

2. Dataset creation

N/A

3. Data splitting

Train data		Validate data		Test data	
Images	Segmentations	Images	Segmentations	Images	Segmentations
image_74.png	segmentation_74.png	image_30.png	segmentation_30.png	image_84.png	segmentation_84.png
image_1.png	segmentation_1.png	image_59.png	segmentation_59.png	image_57.png	segmentation_57.png
image_22.png	segmentation_22.png	image_15.png	segmentation_15.png	image_72.png	segmentation_72.png
image_70.png	segmentation_70.png	image_11.png	segmentation_11.png	image_5.png	segmentation_5.png
image_16.png	segmentation_16.png	image_40.png	segmentation_40.png	image_49.png	segmentation_49.png

image_42.png	segmentation_42.png	image_65.png	segmentation_65.png	image_44.png	segmentation_44.png
image_58.png	segmentation_58.png	image_19.png	segmentation_19.png	image_29.png	segmentation_29.png
image_98.png	segmentation_98.png	image_91.png	segmentation_91.png	image_81.png	segmentation_81.png
image_69.png	segmentation_69.png	image_71.png	segmentation_71.png	image_18.png	segmentation_18.png
image_9.png	segmenttation_9.png	image_86.png	segmentation_86.png	image_0.png	segmentation_0.png
image_77.png	segmentation_77.png	image_97.png	segmentation_97.png	image_25.png	segmentation_25.png
image_55.png	segmentation_55.png	image_43.png	segmentation_43.png	image_36.png	segmentation_36.png
image_96.png	segmentation_96.png	image_10.png	segmentation_10.png	image_75.png	segmentation_75.png
image_80.png	segmentation_80.png	image_8.png	segmentation_8.png	image_39.png	segmentation_39.png
image_52.png	segmentation_52.png	image_3.png	segmentation_3.png	image_90.png	segmentation_90.png
image_23.png	segmentation_23.png	image_7.png	segmentation_7.png	image_12.png	segmentation_12.png
image_28.png	segmentation_28.png	image_83.png	segmentation_83.png	image_78.png	segmentation_78.png
image_26.png	segmentation_26.png	image_85.png	segmentation_85.png	image_79.png	segmentation_79.png
image_54.png	segmentation_54.png	image_60.png	segmentation_60.png	image_2.png	segmentation_2.png
image_34.png	segmentation_34.png	image_6.png	segmentation_6.png	image_37.png	segmentation_37.png
image_73.png	segmentation_73.png				
image_47.png	segmentation_47.png				
image_68.png	segmentation_68.png				
image_41.png	segmentation_41.png				
image_17.png	segmentation_17.png				
image_56.png	segmentation_56.png				
image_45.png	segmentation_45.png				
image_14.png	segmentation_14.png				
image_87.png	segmentation_87.png				
image_94.png	segmentation_94.png				
image_50.png	segmentation_50.png				
image_92.png	segmentation_92.png				
image_13.png	segmentation_13.png				
image_31.png	segmentation_31.png				
image_99.png	segmentation_99.png				
image_82.png	segmentation_82.png				
image_64.png	segmentation_64.png				
image_33.png	segmentation_33.png				
image_48.png	segmentation_48.png				
image_21.png	segmentation_21.png				
image_38.png	segmentation_38.png				
image_51.png	segmentation_51.png				
image_24.png	segmentation_24.png				
image_63.png	segmentation_63.png				
image_62.png	segmentation_62.png				
image_35.png	segmentation_35.png				
image_88.png	segmentation_88.png				
image_67.png	segmentation_67.png				
image_20.png	segmentation_20.png				
image_89.png	segmentation_89.png				
image_61.png	segmentation_61.png				
image_4.png	segmentation_4.png				
image_32.png	segmentation_32.png				
image_95.png	segmentation_95.png				
image_27.png	segmentation_27.png				
image_53.png	segmentation_53.png				
image_66.png	segmentation_66.png				
image_76.png	segmentation_76.png				

image_93.png image_46.png	segmentation_93.png ation_46.png				

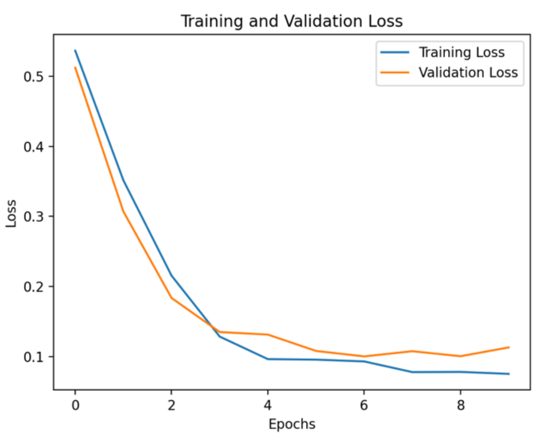
4. Initial preprocessing

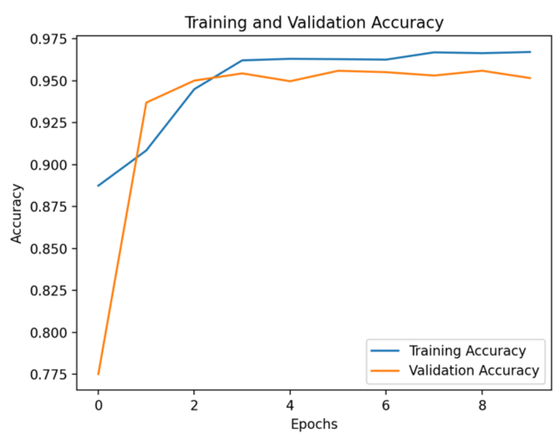
Read image: Since I'm getting image shape as (128, 128, 3) back from the image I produce, the first thing that I have to do is to cast it back into (128, 128, 1) shape. Thus, I use numpy.mean to get the average of all three RGB channels and keep only one channel for the result image. I also threshold the circle value between (230, 255) so that it is not to noisy and hard for the model to detect the edge. That makes the image a bit cleaner and easier to define the circle edge. I didn't use any edge filter or sharpening filter as I figure those make the circle looks distorted.

5. Cross-entropy loss

I. Experiment 1: 20 epochs + Early stopping

I set my model up to train for 20 epochs. I also coupled early stopping with model checkpoint. Wtih this approach, I'm able to help stop the training once no metric improvement is seen for several epochs. The early stopping will halt the training once it does not see the validation accuracy improve for 3 consecutive epochs (patience = 3).

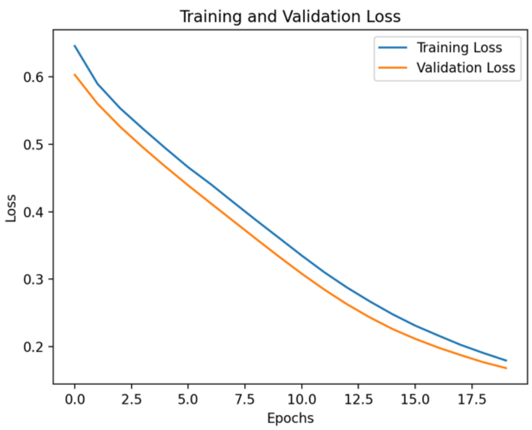


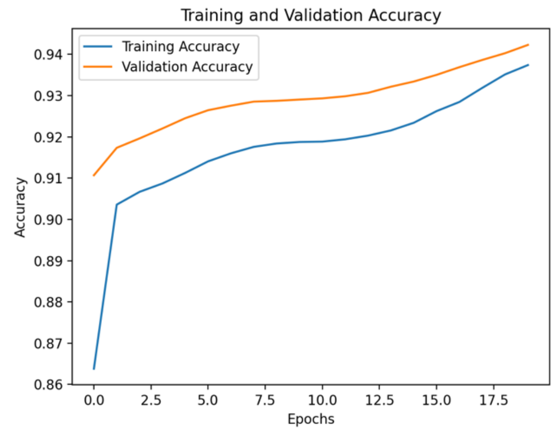


II. Experiment 2: Trying other optimizers (SGD, Adagard, RMSprop)

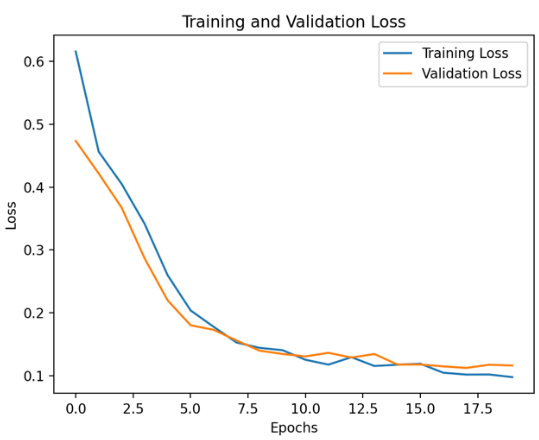
Adam optimizer provides fast convergence rate and being robust to noise in the gradients, in this 2nd experiment, I tried to test out other popular optimizer as well!

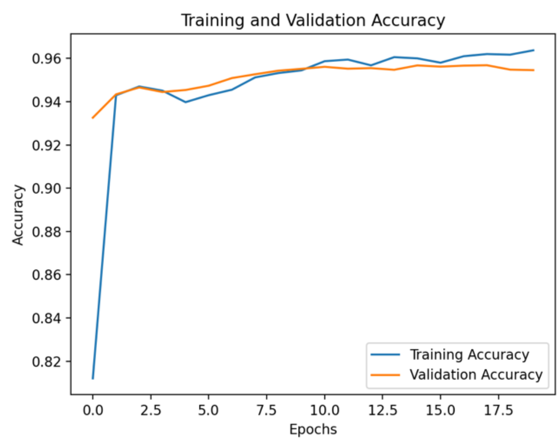
a. SGD optimizer is a simple optimization algorithm that updates the parameters of the network based on the gradient of the loss function with respect to the parameters. It is computationally efficient and easy to implement, but can be slow to converge, as seen in the slow decrease in loss value over epochs, but with a simple task that we have. It achieves high accuracy still.



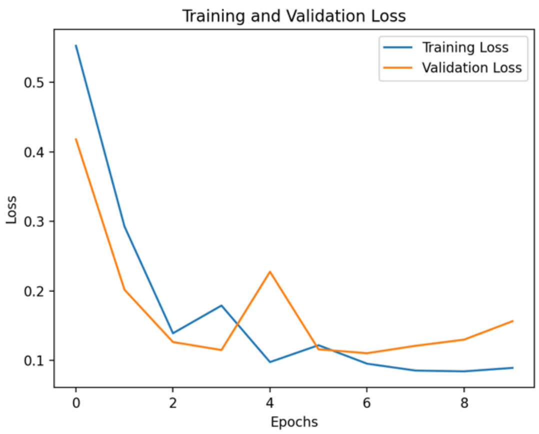


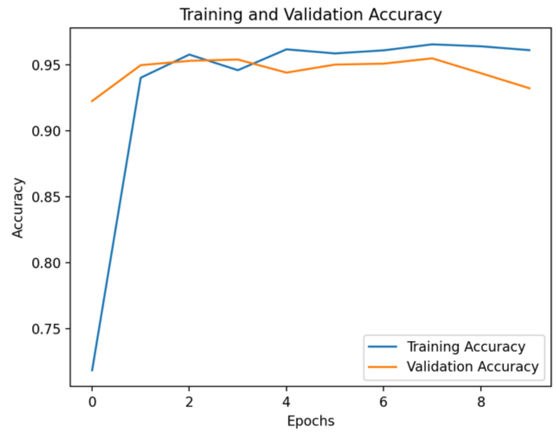
b. Adagard optimizer is an adaptive learning rate method that adapts the learning rate for each parameter based on the history of its gradients. It works well on sparse data. Thus, loss value decreases quickly through epochs and accuracy is well as good as Adam optimizer.





c. RMSprop optimizer is another adaptive learning rate method that uses a moving average of the squared gradients to adjust the learning rate for each parameter. It is designed to handle moving objects, but we will try it anyway for our task as it is a popular and well researched optimizer!

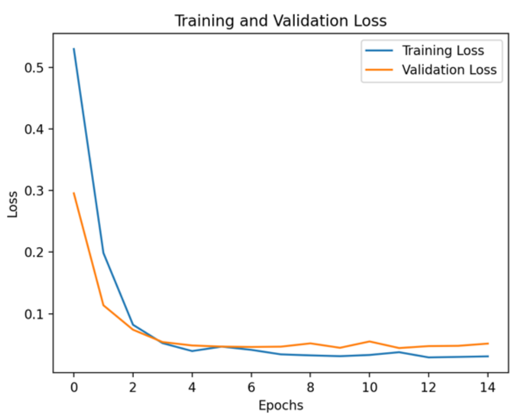


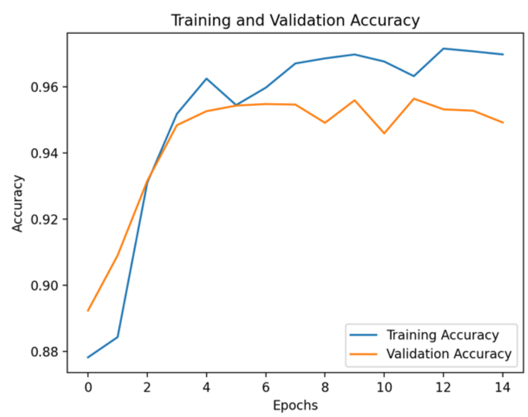


Optimizer experiment conclusion: In the context of our task, the choice of optimizer can have significant impact on the convergence rate and accuracy of the model. Adam optimizer will remain the optimizer we are moving forward with due to its ability to its performance on accuracy and loss convergence. However, it was interesting to experiment with other optimizers and hyperparameters.

6. IOU Loss

For IOU loss



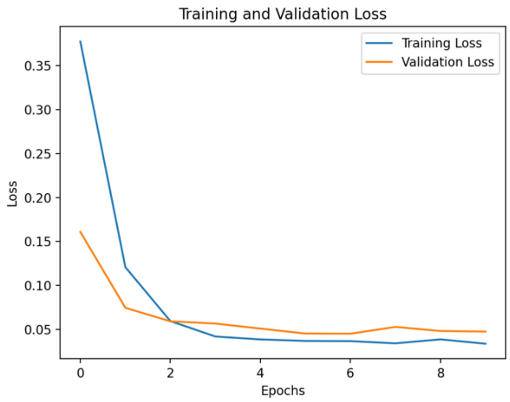


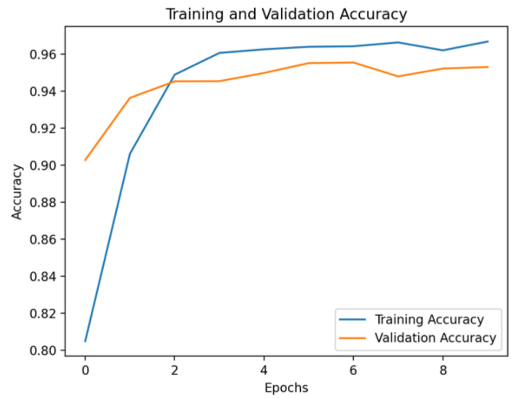
The loss values drastically reduces within the first few epochs, showing quick adaptation to training process and the entire training takes 14 epochs to converge. While the accuracy bumps up and down for a good ~10 epochs, it also converges by the 14 and shows promisingly high numbers!

7. Additional metrics

a. For IOU loss (best weights prioritize loss, not accuracy)

b. For DICE loss (best weights prioritize loss, not accuracy)





c. For Hausdoff
I couldn't figure out hausdorff

8. Qualitative evaluation

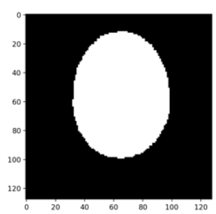
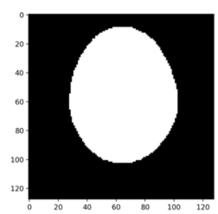
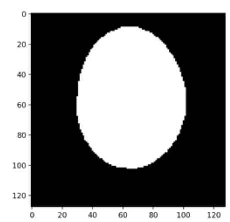
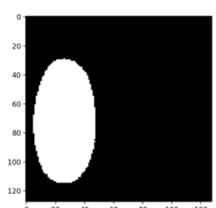
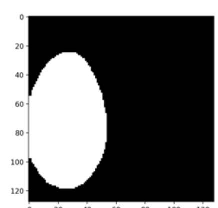
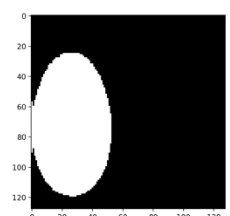
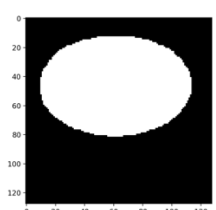
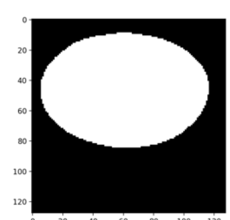
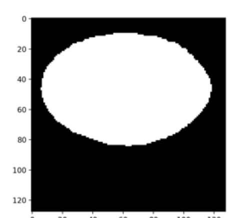
Original image	Cross_entropy loss predictions	IOU loss predictions	DICE loss predictions
image_84.png 			
image_57.png 			
image_72.png 			

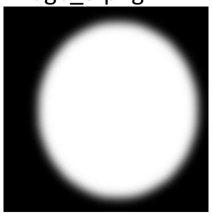
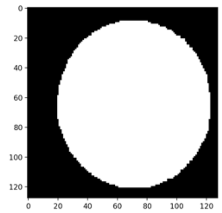
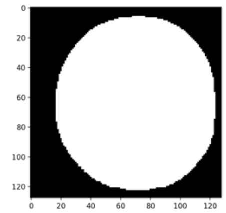
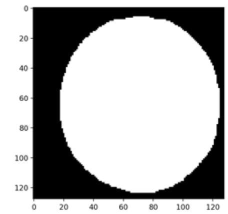
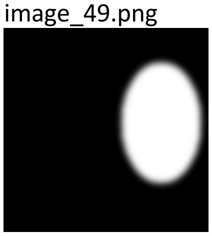
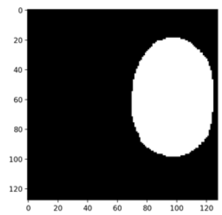
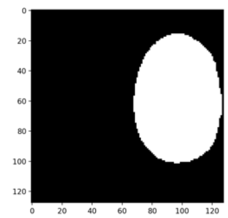
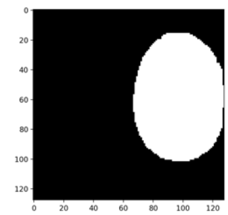
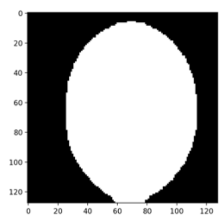
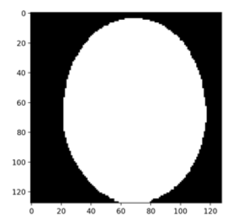
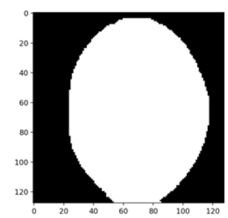
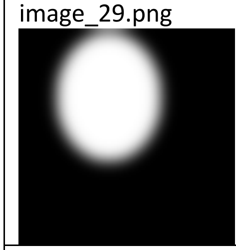
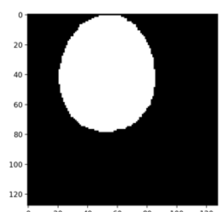
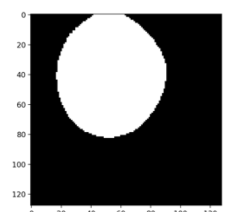
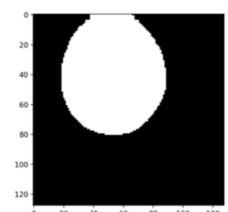
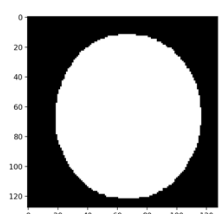
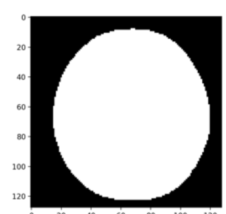
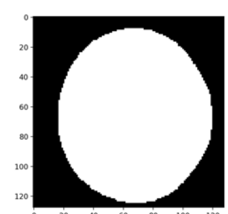
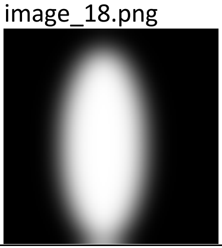
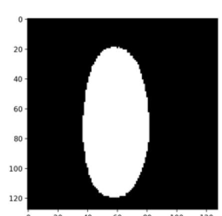
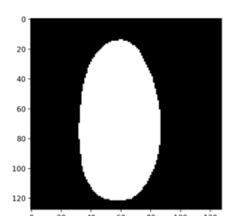
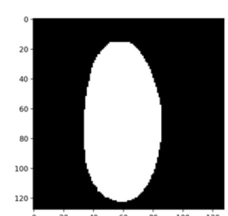
image_5.png 			
image_49.png 			
image_44.png 			
image_29.png 			
image_81.png 			
image_18.png 			

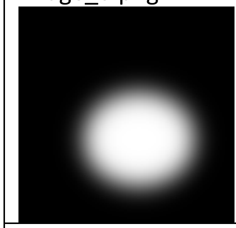
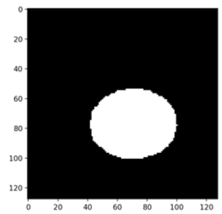
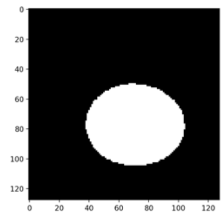
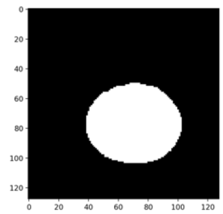
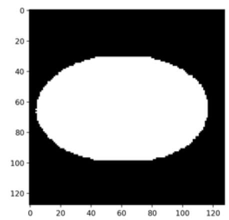
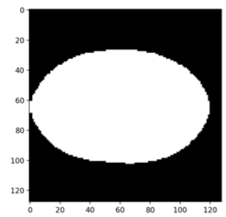
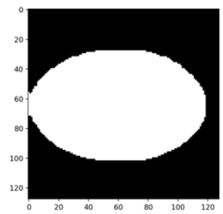
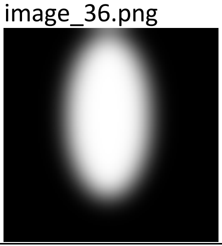
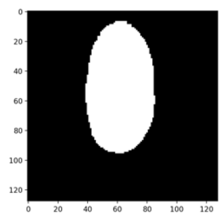
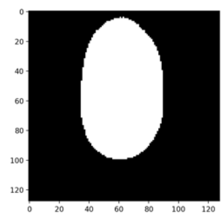
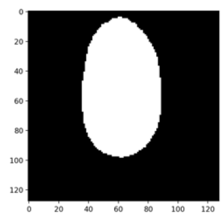
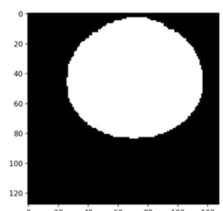
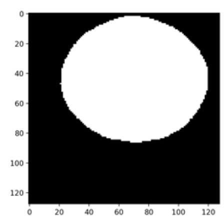
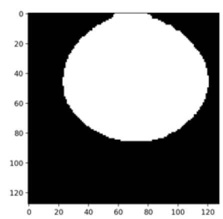
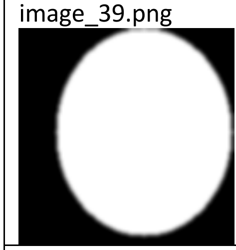
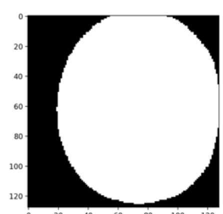
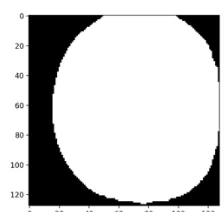
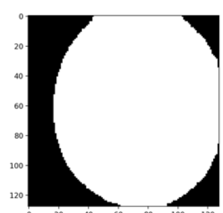
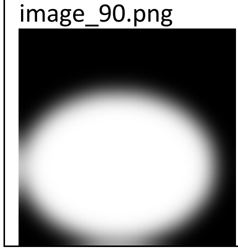
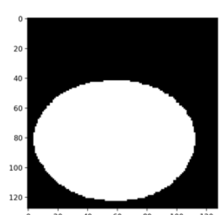
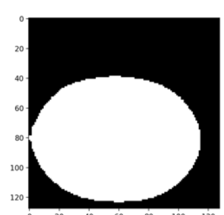
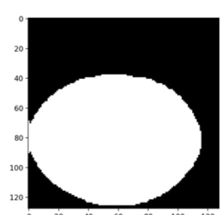
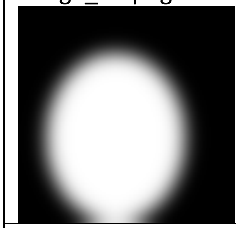
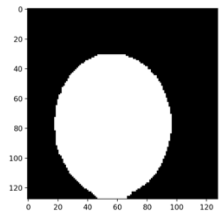
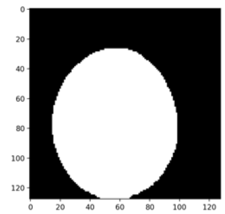
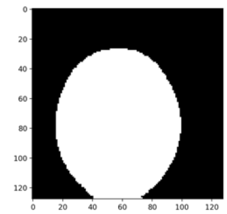
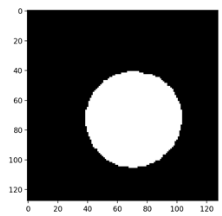
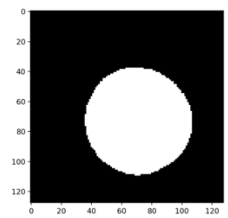
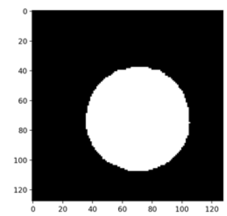
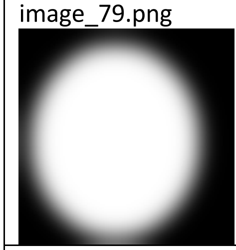
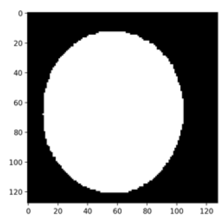
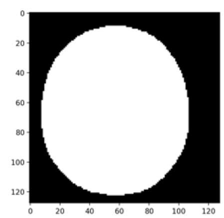
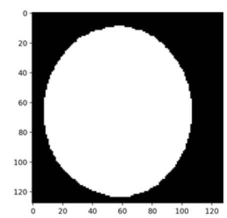
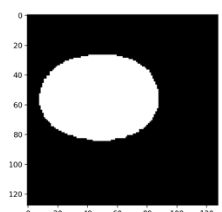
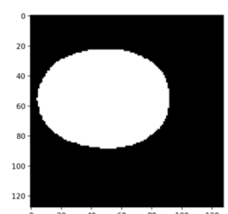
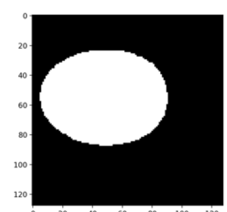
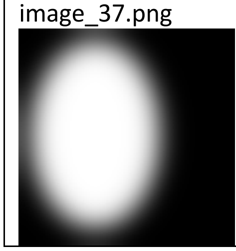
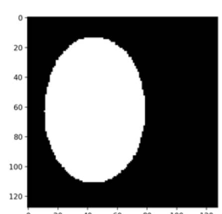
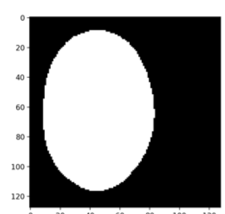
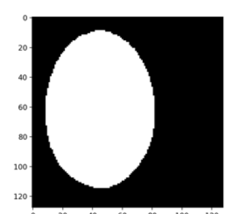
image_0.png 			
image_25.png 			
image_36.png 			
image_75.png 			
image_39.png 			
image_90.png 			

image_12.png 			
image_78.png 			
image_79.png 			
image_2.png 			
image_37.png 			

Speculation:

1. Cross Entropy loss: This loss function is sentitive to class imbalance, thus the circles produced are smaller than the other two loss function. The images also have a bit more of a fuzzy edge since the model may assign some pixels to the wrong class.
2. IOU loss: This loss function produces a slightly better result with clearer boundaries than cross entropy. It is more robust to class imbalance.
3. Dice loss: Dice loss is often preferred for imbalanced datasets, and it is more sensitive to small differences in overlap than IOU loss. The result circles can be slightly bigger than IOU loss, but in generally these two produces quiet similar segmentations.

9. Quantitative evaluation

Metrics/ Loss function	Cross entropy loss function	IOU loss function	DICE loss function
F1 score	0.9670	0.6337	0.9698
Precision score	0.9534	0.6290	0.9657
Recall score	0.8790	0.6313	0.9639

Speculations

1. Cross Entropy Loss function:

    Inference: This loss function is commonly used in image segmentation tasks and works by penalizing the model for misclassifying pixels. High F1 score and precision score suggest that the model was able to accurately identify the white pixels that belong to the white circle. However, the lower recall score suggests that the black background may have made it harder for the model to distinguish between the circle and the background, leading to some false negatives.

2. IOU Loss functions:

    Inference: IOU loss function works by computing the intersection over the union between the predicted segmentation mask and the ground truth mast. It aims to maximize the overlap between the predicted and true masks. The lower F1, precision, and recall score suggest that the model may have struggled to accurately segment the white circle. A possible explanation is that the IOU loss function may not take into the account the shape or size of the circle?

3. DICE Loss function:

    Inference: This loss function is similar to IOU function, but it computers the harmonic mean between the precision and recall scores. The high F1 score, precision, and recall score obtained with the DICE loss function suggest that the model was able to accurately segment the white circle. One possible explanation for this is that the DICE loss function is more sensitive to small differences in overlap between the predicted and true masks.