GI Tract Image Segmentation with U-Net and Mask R-CNN

Alina Chou
Stanford University
alinac@stanford.edu

Wenqi Li
Stanford University
wenqili@stanford.edu

Edgar Roman
Stanford University
emroman@stanford.edu

Abstract

In this report, we present our approaches to the Kaggle UW-Madison GI Tract Image Segmentation challenge. Since radiation oncologists try to deliver high doses of radiation using X-ray beams pointed to tumors while avoiding the stomach and intestines, the goal of the challenge is to effectively segment the stomach and intestines in MRI scans in order to improve the cancer treatment, circumventing the need for the time-consuming and labor intensive process in which radiation oncologists must manually outline the position of the stomach and intestines. We apply U-Net and Mask R-CNN methods to segment the organ areas. Our best U-Net model achieves a Dice score of 0.51, and our best Mask R-CNN model achieves a Dice score of 0.73 on the validation set.

1. Introduction

Every year, there are approximately 5 million new cases of gastrointestinal (GI) cancer and 3.4 million related deaths around the world. Of these patients, about half are eligible for radiation therapy which significantly helps treat the cancer. However, the radiation therapy requires radiation oncologists to deliver high doses of radiation using X-ray beams pointed to tumors while avoiding the stomach and intestines. With newer technology such as integrated magnetic resonance imaging and linear accelerator systems (MR-Linacs), oncologists are able to visualize the daily position of the tumor and intestines. However, they have to manually outline the position of the stomach and intestines in order to adjust the direction of the x-ray beams to increase the dose delivery to the tumor and avoid the stomach and intestines. Unfortunately, this is both a time-consuming and a labor intensive process that can prolong treatments significantly every day, which can be difficult for patients to tolerate.

So the problem that we are investigating is how to effectively track healthy organs in medical scans in order to improve cancer treatment. It is an interesting and significant problem since cancer takes enough of a toll, and solving the problem will enable radiation oncologists to safely deliver higher doses of radiation to tumors while avoiding the stomach and intestines. This will make cancer patients' daily treatments faster and allow them to get more effective treatment with less side effects and better long-term cancer control.

For this project, we are building a model to accurately segment the stomach and intestines on MRI scans. The model takes in inputs of MRI scans from cancer patients who had 1-5 MRI scans on separate days during their radiation treatment provided by the UW-Madison Carbone Cancer Center. We then use U-Net and Mask R-CNN methods to obtain predicted segmented areas of patients' MRI scans with labels for "stomach", "large_bowel", and "small_bowel".

2. Related Work

2.1. U-Net Type Networks

Image segmentation is a classical computer vision task, and its application in biomedicine is particularly extensive. In 2015, Ronneberger *et al*. [10] devised a network architecture, named U-Net, and corresponding data augmentation techniques, in recognition of the fact that training data is usually not as abundant in biomedical applications. Initially, U-Net was used for the task of segmentation of neuronal structure in electoral microscopic images, but the authors showed that U-Net is a strong network architecture in many different biomedical image segmentation tasks.

The architecture of U-Net gives the model the ability of precise localization, meaning the model output a class label for each pixel, and therefore achieve image segmentation. U-Net employs a fully convolutional network structure, replacing traditional pooling layers by upsampling layers, and thereby increases the resolution of the output.

In 2016, Milletari *et al*. [7] introduced V-Net, a network architecture similar to U-Net, but with more specialized target application. In particular, V-Net aims to perform image segmentation on volumetric images, such as image slices obtained from MRIs. Furthermore, the authors introduced the Dice coefficient loss as the object loss function for opti-

mization.

Just like U-Net, the architecture of V-Net is fully convolutional, with a final Dice loss layer, which is differentiable. The convolutional layers of V-Net are 3D convolutions, and this learns the volumetric structure represented by stacks of image slices. To enrich training data, authors of V-Net applied non-linear transformations and histogram matching on the available data.

In 2018, Zhou *et al*. [11] built on top of U-Net and proposed a new architecture named U-Net++. U-Net++ used nested U-Net structures to form an encoder and a decoder, and redesigned connectivity so that the encoder feature map and the corresponding decoder feature map are semantically similar, thereby hoping to make optimization easier.

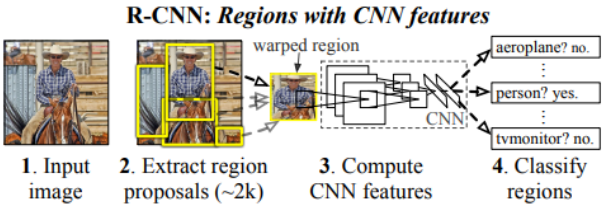
Another important model for medical image segmentation is context encoder network (CE-Net) proposed by Gu *et al*. in 2019 [5]. It is built upon the approach of U-Net but capable of preserving spatial information for 2D medical image segmentation that is previously omitted during the consecutive pooling and strided convolutional operations in U-Net.

The CE-Net contains three major components: a feature encoder module, a context extractor, and a feature decoder module. The authors use pretrained ResNet block as the fixed feature extractor. They applied the proposed CE-Net to different 2D medical image segmentation tasks and obtained in significantly better results compared to the original U-Net method and other state-of-the-art methods.

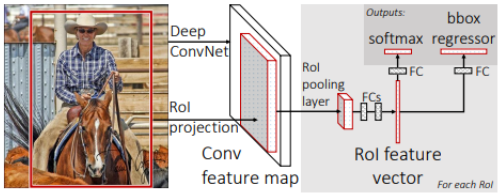
U-Net type of networks are applicable in a variety of biomedical vision tasks, such as Oktay *et al*.'s Attention U-Net for pancrea segmentation [8] and Falk *et al*.'s application to cell counting and detection [1]. This shows the ubiquity of the U-Net structure in biomedical tasks.

2.2. R-CNNs

Up until 2014, the performance of the state-of-the-art methods for object detection performance, as measured by the PASCAL VOC (Visual Object Challenge) dataset, seemed to have plateaued. However, in the paper *Rich feature hierarchies for accurate object detection and semantic segmentation* [4], Girshick *et al*. introduced an approach where they apply high-capacity CNNs to bottom-up region proposals to localize / segment objects and utilize domain-specific fine-tuning after performing supervised pre-training for an auxiliary task in cases where labeled training data is scarce. Girshick *et al*. called their novel proposed method, which combines region proposals with CNNs, Regions with CNN features, or R-CNN. R-CNN consists of thee modules. As described in the paper, the first generates region proposals that are category-independent, the second extracts a feature factor from each region using a large CNN, and the third utilizes a set of class-specific linear SVMs.

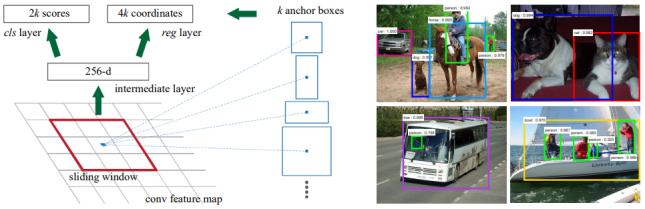


Ultimately, the original R-CNN outperformed the state-of-the-art methods on the PASCAL VOC 2010-12 (achieving a mAP of 53.3%) and ILSVRC2013 (achieving a mAP of 31.4%, compared to the second-best result of 24.3%) datasets. Nonetheless, R-CNN did have some notable drawbacks as pointed out by Girshick (2015) in his paper *Fast R-CNN* [3]. More specifically, Girshick noted that in terms of training, R-CNN utilizes a multi-stage pipeline and it is expensive in not only space but also time. Furthermore, at test-time, objection detection is quite slow. In fact, using the VGG16 CNN on a GPU, the model took about 47 seconds per image. The modified approach in the paper introduced several advantages: higher detection quality, single-stage training (using multi-task loss), an update to all network layers during training, and finally no need to utilize disk storage for feature caching. The Fast R-CNN architecture can be found in the figure below.



On a final note about *Fast R-CNN*, besides the model itself, the paper also contributed to the literature by exposing the region proposal computations as a bottleneck in the R-CNN models at test-time.

This is where Ren *et al*.'s *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* [9] comes into play. The researchers introduced a *Region Proposal Network* or RPN, which is a fully convolutional network that is capable of simultaneously predicting object bounds and objectness scores at each position. It enables region proposals which are quite nearly cost-free by sharing full-image convolutional features with the detection network. This is exactly what *Faster R-CNN* is: the combination of an RPN and Fast R-CNN into a single network. Below this section, we can find the Region Proposal Network on the left-hand side of the figure and example detections using RPN proposals (using PASCAL VOC 2007 dataset) on the right-hand side of the figure.



We finally arrive at He *et al.*'s paper, *Mask R-CNN* [6], which introduces a framework for object instance segmentation. The method extends Faster R-CNN by adding a branch for simultaneously predicting an object mask in addition to the bounding box recognition.

Due to Mask R-CNN's great success in instance segmentation, outperforming baseline variants of previous state-of-the-art models, we decided that it would be beneficial to incorporate this model to tackle the problem of identifying and locating the stomach and intestines in MRI scans.

3. Methods

3.1. Loss Function

We first define the Dice loss function, which will be used universally in all of our methods. Given two binary masks, denoted by M_1 and M_2 the set of pixels they mask on, respectively. The Dice coefficeint of M_1 and M_2 is

$$C = \frac{2|M_1 \cap M_2|}{|M_1| + |M_2|}$$

where the absolute value means the number of pixels. We immediately observe that $0 \leq C \leq 1$. If M_1 is the predicted mask and M_2 is the true mask, the Dice loss is then defined to be $L_{\text{Dice}} = 1 - C$, which we would like to minimize.

3.2. Baseline Method

For our baseline method, we utilize our preprocessed images and compute an average mask for all three segmentation areas – the small bowel, large bowel, and stomach – from the true masks of 70% of the data as the predicted masks. After averaging the masks, we proceed by applying a threshold on the average images and setting the remaining pixels to 1 to complete our final masks for the small bowel, the large bowel, and the stomach. We then calculate the average Dice Coefficient between the masks we produced and the true masks on the remaining 30% of our preprocessed data.

3.3. U-Net

We use the U-Net model on our dataset, and here we first present the network architecture in the original paper by Ronneberger *et al.* [10]. The network consists of two main parts, the down sampling part and the up sampling part (thus the name U-Net).

The down sampling part has 4 steps, and we will call each step a *step down* layer. A step down layer has the following architechture:

$$\text{MaxPool} \longrightarrow \boxed{\begin{array}{l}\text{Conv} \longrightarrow \text{ReLU} \longrightarrow \text{BatchNorm} \\ \downarrow \\ \text{Conv} \longrightarrow \text{ReLU} \longrightarrow \text{BatchNorm}\end{array}}$$

where the first convolutional layer has I input channels and O output channels, and the second convolutional layer has O input channels and O output channels. We will call the parts after `MaxPool` a *double convolutional layer* of type (I, O). The 4 step down layers get exponentially larger, *i.e.* the first one has a double convolutional layer of type $(64, 128)$, the second has a double convolutional layer of type $(128, 256)$, the third $(256, 512)$, and the fourth $(512, 1024)$. The `MaxPool` layer always has kernel size 2 and stride 2.

The up sampling part has 4 corresponding steps, and we will call each step a *step up* layer. A step up layer consists of a upsample layer and a double convolutional layer. The 4 step up layers get exponentially smaller, *i.e.* the first one has a double convolutional layer of type $(1024, 512)$, the second has a double convolutional layer of type $(512, 256)$, the third $(256, 128)$, and the fourth $(128, 64)$.

Finally, the whole network has the following architecture:

$$\text{DoubleConv} \longrightarrow \text{Down} \longrightarrow \text{Up} \longrightarrow \text{Conv}$$

The first double convolutional layer is of type $(3, 64)$ where 3 is the number of channels in the input image. The down and up parts are followed by a convolutional layer with 64 input channels and 2 output channels, where 2 is the number of classes since we are doing a binary segmentation.

We notice that the network is fully convolutional and has no regularization. This is because in the original application of U-Net, training data was lacking. For our application, since there is abundant training data, we add dropout layers in between the double convolutional layers. Due to memory constraints, we remove the largest step down layer and the largest step up layer in some experiments.

Using the Dice loss as the loss function did not allow the model to achieve a desirable performance, as our experiments show. Therefore in our best performing U-Net model, we use a loss function that is a sum of the cross entropy loss (between the predicted mask and the true mask) and the Dice loss.

3.4. Mask R-CNN

Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression. The model's framework

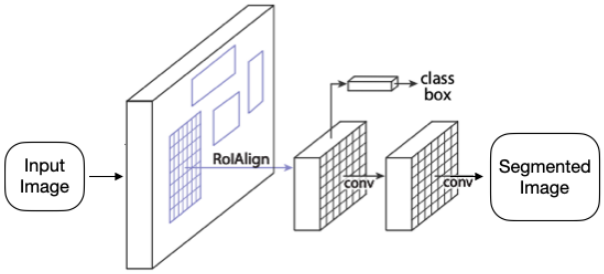


Figure 1. Mask R-CNN framework for instance segmentation

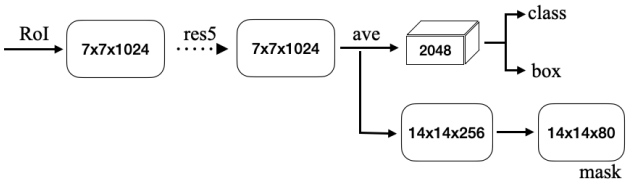


Figure 2. Mask R-CNN Head Architecture: Faster R-CNN with ResNet Backbone

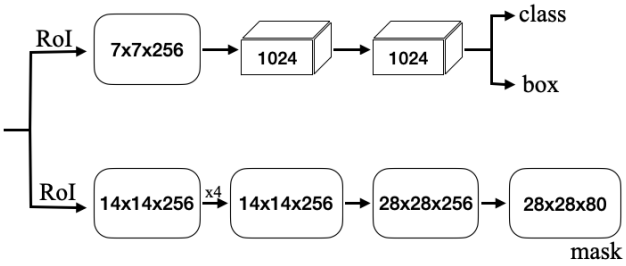


Figure 3. Mask R-CNN Head Architecture: Faster R-CNN with FPN Backbone

for instance segmentation is shown as Figure 1. It adopts the same two-stage procedure as Faster R-CNN which uses the Region Proposal Network as first stage to propose candidate object bounding boxes and the second stage being extracting features using RoIPool from each candidate box and performing classification and bounding-box regression. However, in the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. During training, a multi-task loss on each sampled RoI is defined as $L = L_{cls} + L_{box} + L_{mask}$ with L_{cls} denoting the classification loss L_{box} denoting the bounding-box loss, and L_{mask} denoting the average binary cross-entropy loss.

The network architecture for Mask R-CNN is differentiated into (1) the convolutional *backbone* architecture used for feature extraction over an entire image and (2) the network *head* for bounding-box recognition (classification and regression) and mask prediction that is applied separately to each RoI. The *backbone* architecture can be chosen based on specific models, though He *et al*. propose in their paper [6] that ResNet-FPN backbone for feature extraction with Mask R-CNN gives excellent gains in both accuracy and speed, which is the backbone architecture that this project choose to use as well. For the network *head*, Mask R-CNN extends two existing Faster R-CNN heads. The heads for Faster R-CNN with ResNet backbone is shown as Figure 2 and the heads for Faster R-CNN with FPN backbone is shown as Figure 3.

4. Dataset and Features

4.1. Dataset Details

The data we use is anonymized MRIs of patients treated with MRI guided radiotherapy provided by the UW-Madison Carbone Cancer Center. Specifically, the dataset contains 85 cases with 38496 scan slices of organs represented in 16-bit grayscale PNG format, and the annotations are provided in a csv format with the segmented areas represented as RLE-encoded masks. Sample csv annotations are shown in Table 1. An empty segmentation entry represents no mask presented for the class in the MRI scan slice. All the data is available as part of the Kaggle challenge [1].

id	class	segmentation
case123_day20_slice_0081	large_bowel	17746 6 18010 9 ...
case123_day20_slice_0081	small_bowel	
case123_day20_slice_0081	stomach	11055 4 11315 14 ...
case123_day20_slice_0082	large_bowel	17481 4 17746 7 ...
case123_day20_slice_0082	small_bowel	22236 2 22500 6 ...
case123_day20_slice_0082	stomach	11052 8 11314 15 ...

Table 1. MRI Scan Dataset CSV Annotation Sample

4.2. Preprocessing

Upon observing the dataset, we notice that it contains images which are ambiguous and show little visual information through the scan. So we first preprocess the dataset by obtaining only the images that have sufficient visual information. Further observing and processing the image sizes in the dataset, we notice that the most common size across all images (67.33% occurrence) in the dataset is 266×266,

[1] https://www.kaggle.com/competitions/uw-madison-gi-tract-image-segmentation/data

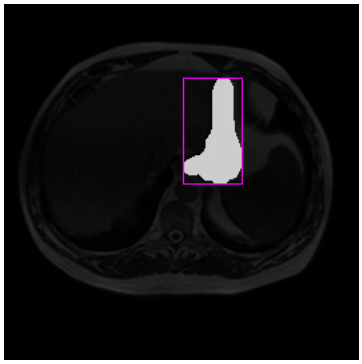


Figure 4. Image of Segmented Stomach with Mask & Bounding Box

and the rest are of sizes 310×360, 276×276, and 234×234 in frequency descending order. In order to create masks for segmented areas, we also have to ensure the sizes of images are the same, so we reshape all images to size 256×256 as it is a power-of-two number (required in evaluation calculation) that is closest to the current image sizes. After resizing the images, we then scale the RLE encoded segmentation masks accordingly for masks to cover same corresponding regions in each MRI scan as prior to reshaping.

Moreover, since the training labels are 16-bit RLE encoded masks that are less interpretable visually, we convert them into pixel values and visualize the labeled areas by highlighting the masks and drawing bounding boxes around the region. An example of the visualized labeled area is shown as Figure 4. We further overlay mask areas for all of the segmented organs – large bowel, stomach, and small bowel – with original MRI scan examples shown in Figure 5.

In order to feed the image, mask, and target into training models, we also process them as tensors and store them in separate files which will allow us to obtain the information simply by loading the proprocessed files. Finally, we split up the dataset to training, validation, and testing sets by 70%, 20%, and 10% respectively.

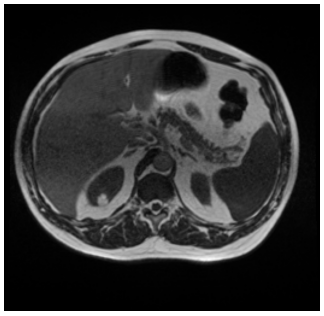
5. Experiments/Results/Discussion

5.1. Evaluation method

We use the Dice score to evaluate the mask predicted by our model against the true mask. As described above, the Dice score of a predicted mask is defined by

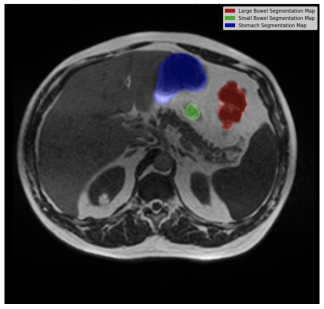
$$\frac{2|M_{\text{pred}} \cap M_{\text{true}}|}{|M_{\text{pred}}| + |M_{\text{true}}|}$$

where M_{pred} is the set of pixels masked in (*i.e.* predicted to class "1") by the model, M_{true} is the set of pixels masked in by ground truth mask, and the absolute value means the number of pixels.



(a) Original MRI Scan

(b) Large Bowel, Stomach, and Small Bowel Masks



(c) MRI with Mask Overlay

Figure 5. Example Preprocessing Mask Overlay Visualization

5.2. Baseline

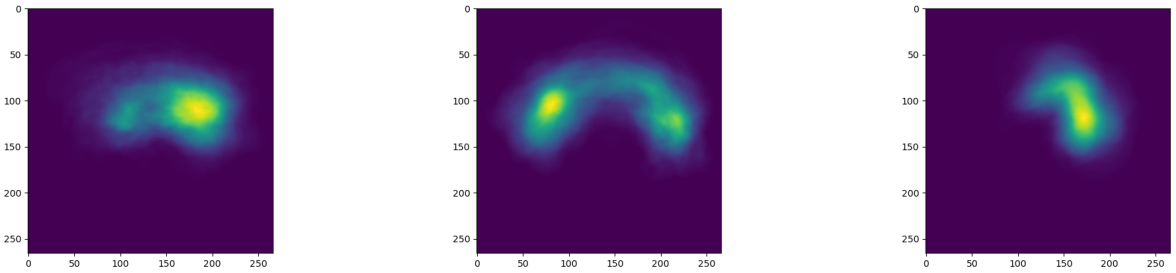
After calculating the average masks for small bowel, large bowel, and stomach based on preprocessed data, we visualize them as presented in Figure 6. Then, we calculate the Dice score for the predicted mask on the validation and testing set that we split off from the original dataset and average the scores to obtain a result of 0.33 Dice score.

5.3. U-Net

For the U-Net model, our implementation mainly follow the open source implementation by milesial [2]. We train the U-Net model on our dataset from scratch, and we experiment with multiple hyperparameter combinations as well as tune on the validation set.

In Table 2, the scaling parameter determines how much we scale the image in preprocessing. The scaling of 0.5 serves to reduce memory usage. The depth parameter is the number of step down and corresponding step up layers in the Down and Up part of the U-Net architecture. In particular, our device does not have enough memory to support training with a batch size of 64 and depth 4, so we choose

[2] https://github.com/milesial/Pytorch-UNet



(a) Average Small Bowel (b) Average Large Bowel (c) Average Stomach

Figure 6. Average masks for different segmentation regions.

Parameter Name	Default	Comb. 1	Comb. 2
batch size	1	32	32
learning rate	1e-5	1e-5	1e-4
max epochs	15	15	15
scaling	0.5	0.5	0.5
depth	4	4	4

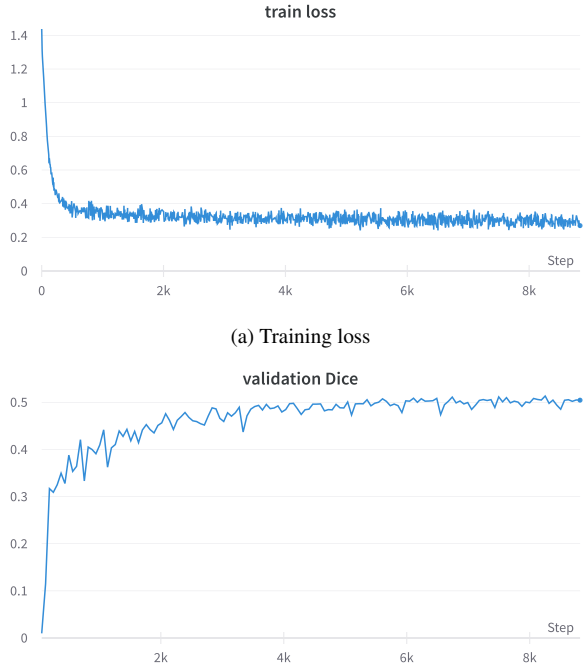
Table 2. Hyperparameter Combinations for U-Net

Parameter Name	**Comb 3.**	Comb. 4	Comb. 5
batch size	**32**	64	64
learning rate	**5e-5**	5e-5	1e-4
max epochs	**15**	15	15
scaling	**1**	0.5	1
depth	**3**	3	3

Table 2. Hyperparameter Combinations for U-Net

to remove the largest step down and step up layer in such experiments. It turns out that using a shallower U-Net gives comparable performance, and our best performing U-Net model has depth 3. For the optimizer, we use RMSprop and a ReduceLROnPlateau scheduler monitoring the validation Dice coefficient. Therefore, the learning rate exponentially decays by a factor of 10 in later epochs when the validation Dice score is not increasing. Although typically for a RMSprop optimizer the learning rate is 0.001, we observe that such a learning will immediately make the gradient explode. Upon tuning on the validation set, we find that learning rates $5e-5$ or $1e-4$ are the most desirable.

In Table 2, the parameter combination in the bolded column obtains the best performance, achieving a validation Dice score of 0.5134. We see that the training loss decrease in an expected manner, and the validation score increase accordingly. A final training Dice score check shows that the average Dice score on the training set is 0.54, which is not significantly higher than the validation score. Therefore, no significant overfitting occurred, which makes sense



(a) Training loss

(b) Validation Dice

Figure 7. Best Performing U-Net Results

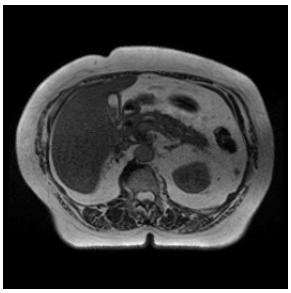
since we added dropout layers as a regularizer.

We next examine some incorrect masks predicted by the U-Net model. We notice that our model is almost always over predicting, *i.e.* the predicted mask would mask in approximately all of the true mask, together with other pixels. See Figure 8 and Figure 9 for two such examples.

This behavior might be due to the way which the Dice score (and thus the Dice loss) is computed. In the Dice score formula

$$\frac{2|M_{\text{pred}} \cap M_{\text{true}}|}{|M_{\text{pred}}| + |M_{\text{true}}|},$$

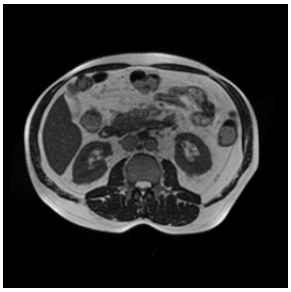
we see that the largest value that the denominator can take



(a) Original MRI Scan

(b) Predicted Mask (c) True Mask

Figure 8. Over-predicted Mask Example 1 Using U-Net



(a) Original MRI Scan

(b) Predicted Mask (c) True Mask

Figure 9. Over-predicted Mask Example 2 Using U-Net

on is $2|M_{\text{true}}|$, so the model tries hard to include all of the true mask. Although including more pixels than needed is punished in the denominator, we see that the constant $|M_{\text{true}}|$ on the denominator makes this punishment not as severe, while the factor 2 on the numerator encourages the model to be more bold (include all true pixels, together with others) than careful (including only true pixels, but perhaps

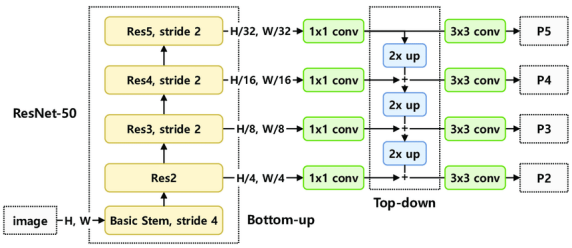


Figure 10. ResNet-50-FPN Architecture

not all of them). Therefore, we believe that adding a suitably weighted $|M_{\text{pred}}|$ term in the loss would help make the model less bold and result in fewer over-predicted masks.

5.4. Mask R-CNN

Our Mask R-CNN approach on MRI organ segmentation uses the following hyperparameters for training:

Parameter Name	Value
Batch Size	2
Learning Rate	1e-2
Max Epochs	7
Weight Decay	5e-4
Optimizer	SGD
Momentum	0.9

Table 3. Hyperparameters for Mask R-CNN Training

These hyperparameters are chosen after examining Gerber *et al.*'s work on hyperparameter tuning for Mask R-CNN [2] as they are able to achieve better training performance.

Furthermore, Table 4 shows our choice of hyperparameters that are specific to tuning the Mask R-CNN model.

Parameter Name	Value
Backbone	ResNet-50-FPN
Train RoIs Per Image	64
Max GT Instances	3
Detection Min Confidence	0.7

Table 4. Mask R-CNN Specific Hyperparameters

For our training, we decide to select ResNet-50-FPN for the Backbone (architecture shown in Figure 10). From our understanding, this allows for the model to train somewhat faster than if we had selected a Backbone like ResNet 101 or ResNeXt 101. However, this might have come as a trade-off between training time and accuracy. For this reason, if we had more time, we would have liked to experiment with different choices for the Backbone.

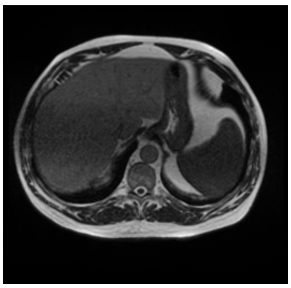
Now, in terms of Train RoIs Per Image, we decide to decrease this number to 64 from the default value due to the

fact that the sizes of the images are relatively smaller (image resolution typically 266 x 266, 360 x 310) than those found in the COCO dataset (image resolution of 640 x 480) as well as the fact that there simply are a fewer number of instances we attempt to segment within each of the images. After all, we hope to only segment a single stomach, small bowel, and large bowel. The result of using a smaller number of RoIs per image is a reduced training time.

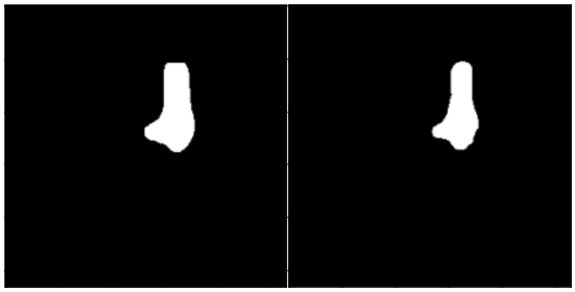
Similarly, we also reduce the Max GT Instances, or the maximum number of instances that could be detected in a single image since we only expect to find a single stomach, small bowel, and large bowel. Not only does this result in a reduced training time, but it also theoretically helps reduce the chance of any false positives.

For this same reason, we select to utilize a Detection Min Confidence value of 0.7 to minimize the number of false positives and ensure that we are confident in our predictions when we detect one of the three organs which we are attempting to segment.

After training the model using these parameters specified in Table 3 and Table 4, we evaluate the segmentation prediction result with the Dice loss and obtain a Dice score of 0.7732 on the training set and 0.7266 on the validation set. We further visualize the predicted mask compared with the true mask for different organ slices with example shown in Figure 11. As we can notice, the predicted mask is able to capture the area close to the true mask.



(a) Original MRI Scan



(b) Predicted Stomach Mask (c) True Stomach Mask

Figure 11. Mask R-CNN Mask Prediction Example Result

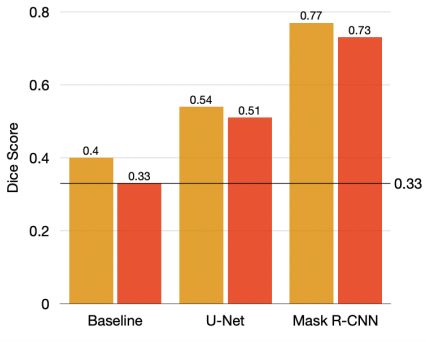


Figure 12. Dice Score Comparison Across Models

5.5. Discussion

Figure 12 demonstrates the comparison of results obtained using the baseline mask average method, U-Net model, and the Mask R-CNN model in GI image segmentation with the left column representing results on the training set and right column representing result on the validation set. As we can see, both U-Net and Mask R-CNN significantly improve the performance compared to the baseline, while Mask R-CNN achieves better result than U-Net. The reason might be that the loss function for the U-Net model is formulated in too naive a way compared to the Mask R-CNN model.

6. Conclusion/Future Work

In this project, we implemented and modified the U-Net model and the Mask R-CNN model, and applied them to the GI tract image segmentation problem. Our U-Net based model achieves a Dice score of 0.54 on the training set and 0.5134 on the validation set, while Mask R-CNN model achieves a Dice score of 0.7732 on the training set and 0.7266 on the validation set. Although both methods achieve significantly better results compared to the baseline model (0.33 Dice score on the validation set), we see that the Mask R-CNN model performs better than the U-Net model, and the reason might be that the loss function for the U-Net model is formulated in too naive a way compared to the Mask R-CNN model.

We hypothesize that several future directions may further improve our results and overcome the bottlenecks of the current network design:

1. Formulate a more sophisticated loss function for the U-Net model. Considering the current behavior of our U-Net model. including a term that punishes over-predicting might increase the performance.

2. Ensemble the U-Net architecture with the Mask R-CNN, potentially using U-Net as the backbone architecture of Mask R-CNN.

3. Data augmentation. We conjecture that the U-Net model not performing as well as expected might also be because the key data augmentation procedure in the original paper [10] was not performed, due to time and computational resources constraints. In particular, U-Net was not able to gain deformation-invariance in its learning from deformed training images.

7. Appendices

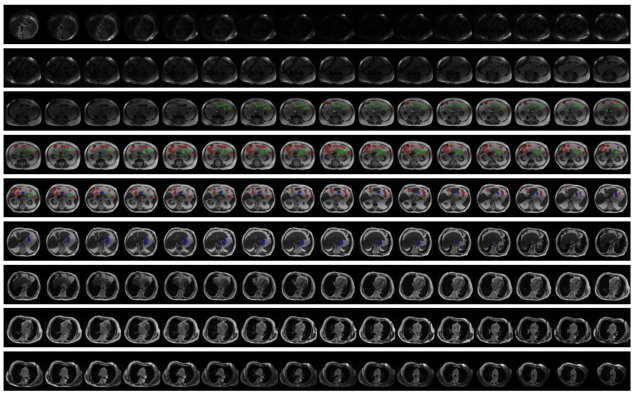


Figure 13. Dataset Preprocessing: Mask Overlay Example

8. Contributions and Acknowledgements

Alina Chou implemented and contributed to the baseline model, Mask R-CNN model, and evaluation metrics, and obtained training and evaluation results with visualization.

Wenqi Li implemented, modified, and trained the U-Net model, and wrote the sections on U-Net in this paper.

Edgar Roman implemented the initial pre-processing of the data, the baseline model, and contributed to the application of Mask R-CNN on the MRI data.

References

[1] Thorsten Falk, Dominic Mai, Robert Bensch, Özgün Çiçek, Ahmed Abdulkadir, Yassine Marrakchi, Anton Böhm, Jan Deubner, Zoe Jäckel, Katharina Seiwald, et al. U-net: deep learning for cell counting, detection, and morphometry. *Nature methods*, 16(1):67–70, 2019. 2

[2] Mia Gerber, Nelishia Pillay, Katerina Holan, Steven A. Whitham, and Dave K. Berger. Automated hyper-parameter tuning of a mask r-cnn for quantifying common rust severity in maize. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2021. 7

[3] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[4] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014. 2

[5] Zaiwang Gu, Jun Cheng, Huazhu Fu, Kang Zhou, Huaying Hao, Yitian Zhao, Tianyang Zhang, Shenghua Gao, and Jiang Liu. Ce-net: Context encoder network for 2d medical image segmentation. *CoRR*, abs/1903.02740, 2019. 2

[6] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. 3, 4

[7] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *2016 fourth international conference on 3D vision (3DV)*, pages 565–571. IEEE, 2016. 1

[8] Ozan Oktay, Jo Schlemper, Loic Le Folgoc, Matthew Lee, Mattias Heinrich, Kazunari Misawa, Kensaku Mori, Steven McDonagh, Nils Y Hammerla, Bernhard Kainz, et al. Attention u-net: Learning where to look for the pancreas. *arXiv preprint arXiv:1804.03999*, 2018. 2

[9] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1, 3, 9

[11] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*, pages 3–11. Springer, 2018. 2