MULTI-ORGAN IMAGE SEGMENTATION

DATS 6303: Deep Learning

Team 3: Liang Gao & Kanishk Goel

2024-5-3

Contents

1 Introduction **4**

2 Dataset **4**

3 Model: U-Net **6**

4 Model: FCN **9**

5 Conclusion **11**

List of Figures

1 MRI scans . . . 4
2 MRI scans with Masks . . . 4
3 Number of Masks in each image . . . 5
4 Percentage: Image with Masks . . . 5
5 U-net architecture . . . 7
6 Results: BCE loss as criterion . . . 7
7 Results: Dice loss as criterion . . . 7
8 Results: Combination of Dice and BCE loss as criterion . . . 8
9 Ground Truth segmentation . . . 8
10 Prediction segmentation . . . 8
11 FCN outline . . . 9
12 FCN Structures . . . 9
13 Transpose Convolution . . . 10
14 IoU Loss . . . 11
15 Target . . . 11
16 Prediction . . . 11

List of Tables

1 Dice coe for three classes in test dataset . . . 6

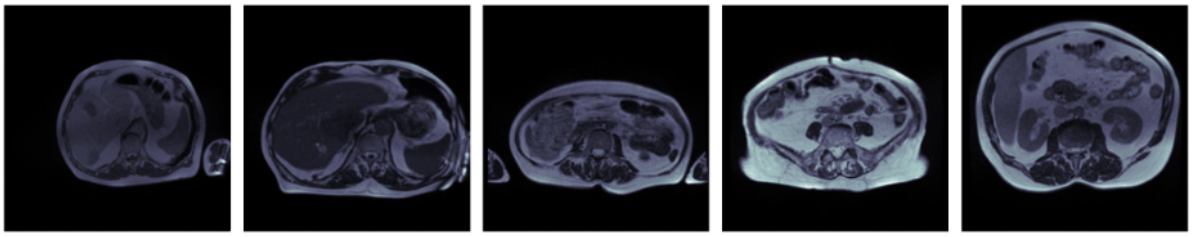

Figure 1: MRI scans

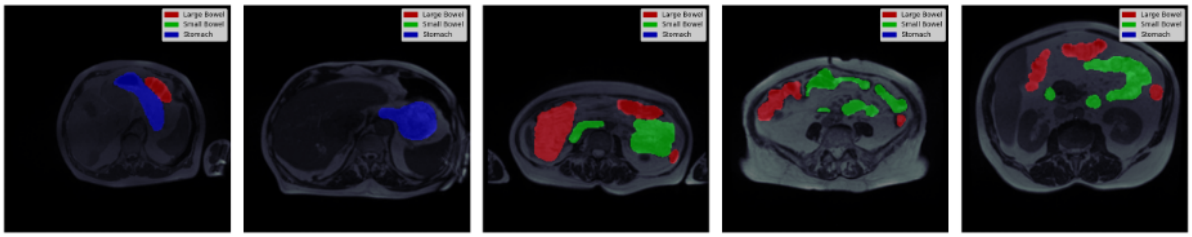

Figure 2: MRI scans with Masks

1 Introduction

Image segmentation is currently a highly popular topic in computer vision. Medical image segmentation represents a dynamic intersection of computer vision and medicine, bringing together advanced imaging techniques and analytical methodologies to tackle some of the most pressing challenges in healthcare.

The primary objective of this team project is to gain a comprehensive understanding of medical image segmentation and to delve into the functionality of convolutional neural networks (CNN) in addressing complex image-related challenges.

2 Dataset

The data are MRI scans from actual cancer patients on separate days during radiation treatment. The data is from the UW-Madison Carbone Cancer Center, a pioneer in MR-Linac-based radiotherapy [1]. The raw data has 115,488 observations and three columns:

Id: unique identifier for the object

class: the predicted class for the object (large bowel, small bowel, and stomach)

segmentation: RLE-encoded pixels for the identified object (28094 3 28358 7...)

RLE – Run length encoding([2]) is a basic form of data compression where sequences of the same data value are stored as a pair of data values and count. For example, 28094 is the value, and 3 is the count of this value. This method is particularly efficient for images with large areas of uniform pixels.

Figure 1 are examples of visualized MRI scans. Figure 2 are MRI scans with masks.

In the raw dataset, each 'id' represents a unique image slice. Each image slice has separate rows that classify segments of the image as 'large_bowel,' 'small_bowel,' and 'stomach.' The dataset structure has been reorganized such that each image slice is represented by a single row with three distinct columns dedicated to the three organ classes. Each column contains RLE-encoded pixel values initially listed in the segmentation columns of the dataset. We counted the number of masks(Figure 3) in each slice and deleted the images without any masks. Figure 4 illustrates the distribution of images containing segmentation masks across three anatomical regions. It presents the percentages of the images that feature segmentation masks for each specified organ, indicating that 36.59% of the images have masks for the large bowel, 29.1% for the small bowel, and 22.41% for the stomach.

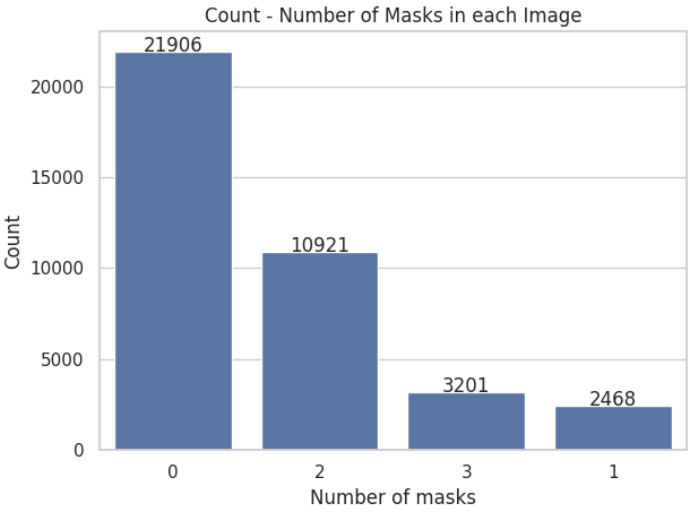

Figure 3: Number of Masks in each image

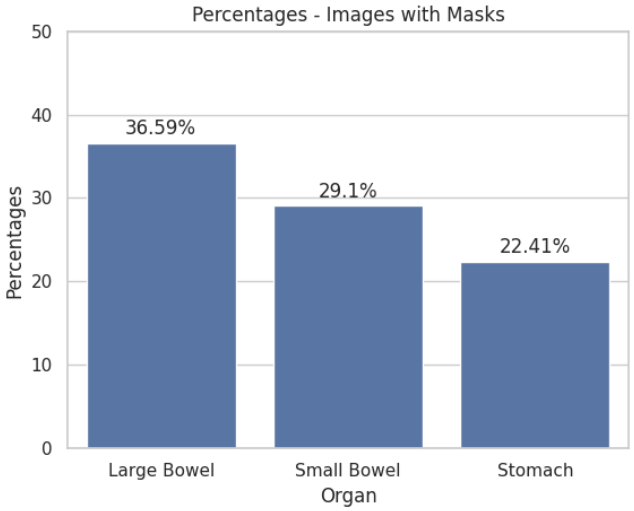

Figure 4: Percentage: Image with Masks

3 Model: U-Net

U-Net is a convolutional neural network(CNN) specifically designed for medical image segmentation. Initially developed for biomedical image segmentation, the architecture of U-Net is structured to work effectively with fewer training samples and to yield more precise segmentation. It is characterized by its symmetric shape, which includes a contracting path to capture context and a symmetric expanding path that enables precise localization. This unique architecture allows U-Net to learn from a substantial amount of spatial context from input images, making it highly effective in detecting the boundaries of objects within various medical scans. Due to these attributes, U-Net has become a standard tool in medical image analysis, particularly in tasks requiring detailed and accurate anatomical structure segmentation [3].

The network architecture, depicted in Figure 5, is a fully convolutional network that exclusively utilizes convolutional layers without any dense layers. The architecture is divided into two main paths: a contracting(downsample) path on the left and an expansive(upsample) path on the right. The contracting path uses a series of two 3x3 unpadded convolutions followed by a ReLU activation and a 2x2 max pooling with a stride of 2 for downsampling, during which the number of feature channels doubles at each step. The expansive path involves upsampling of the feature map, followed by a 2x2 convolution that reduces the feature channels by half, concatenation with the corresponding cropped feature map from the contracting path, and two subsequent 3x3 convolutions each followed by a ReLU. The final layer employs a 1x1 convolution to map the features to the desired number of output classes.

The model architecture code is based on the Pytorch-UNet repository by Milesial (2020)[4], and the code for the training part references Saraki, (2023)[5]. We used the DICE coefficient as our metric. It is a statistical tool used to measure the similarity between two sets of data. Compare the pixel-wise agreement between a ground truth segmentation and a predicted segmentation. The equation to calculate the DICE coefficient is:

$$\text{Dice} = \frac{2 \times |X \cap Y|}{|X| + |Y|} \tag{1}$$

The corresponding loss function is the DICE loss, which directly considers the overlap between the predicted and true segmentation masks.

$$\text{Dice loss} = 1 - \text{Dice Coefficient} \tag{2}$$

For the training part, most techniques I used were the same as exams, such as adding data augmentation for training data and changing image size and number of channels. I tried both 1(grey) and 3(RGB) as my input channel. There was not much difference in the performance, so I finally set my input channel as 1. We have three classes/masks, so the output channel is three. Both input and output are images in the torch tensor format. For the training criterion, I want to compare the results of two different loss functions: BCEWithLogitsLoss, which I used during the exam, and DICE loss, which is new to me. I plot the training loss versus validation loss against the number of epochs and the Dice coefficient of the validation dataset. Figure 6 is the result with BCEWithLogitsLoss as the criterion, figure 7 is the result of using Dice los, and figure 8 is the result with a combination of both loss functions. For the loss plot, the curve with BCEWithLogitsLoss is slightly smoother than the other two. There is not much difference in the results plot. The result of the test dataset used the model with Dice loss as the criterion.

Test Result I calculated the Dice coefficient separately for three classes. The results are in table 1.

Classes	Large bowel	Small bowel	Stomach
Dice coe	0.90	0.88	0.93

Table 1: Dice coe for three classes in test dataset

The ground truth segmentation is in figure 9, and the predicted segmentation is in figure 10.

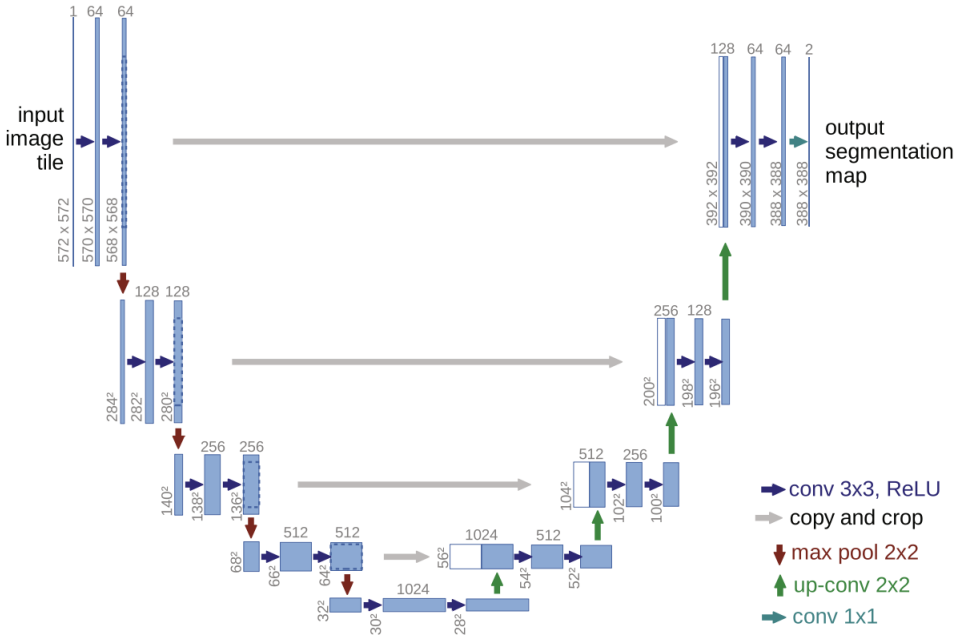

Figure 5: U-net architecture

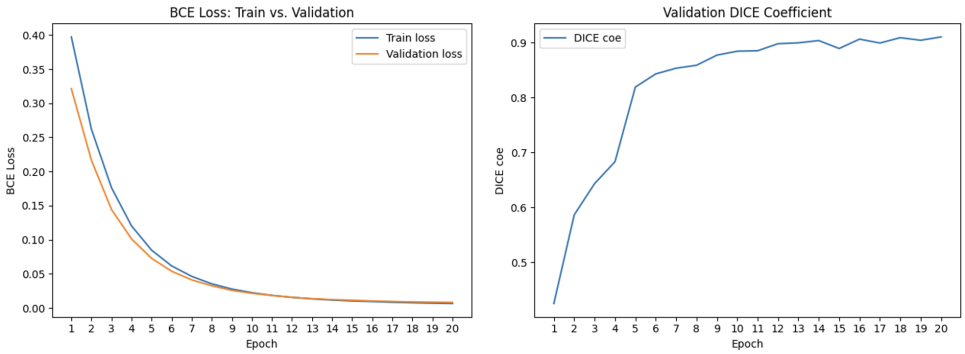

Figure 6: Results: BCE loss as criterion

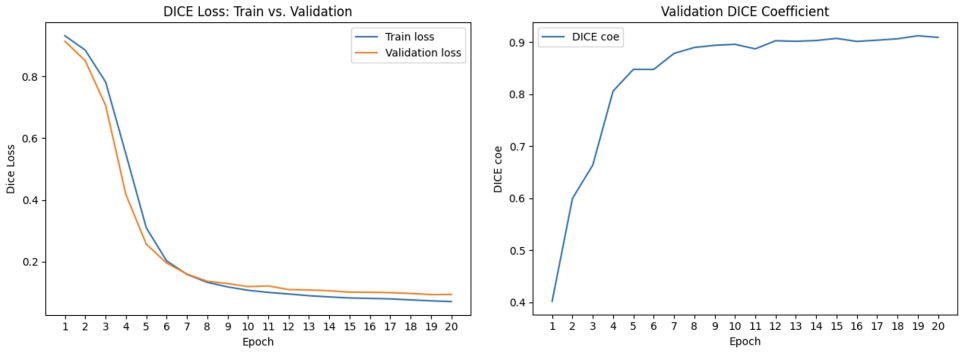

Figure 7: Results: Dice loss as criterion

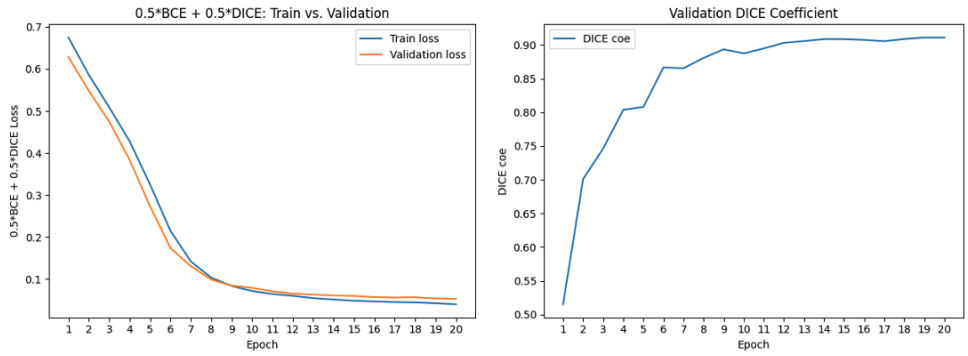

Figure 8: Results: Combination of Dice and BCE loss as criterion

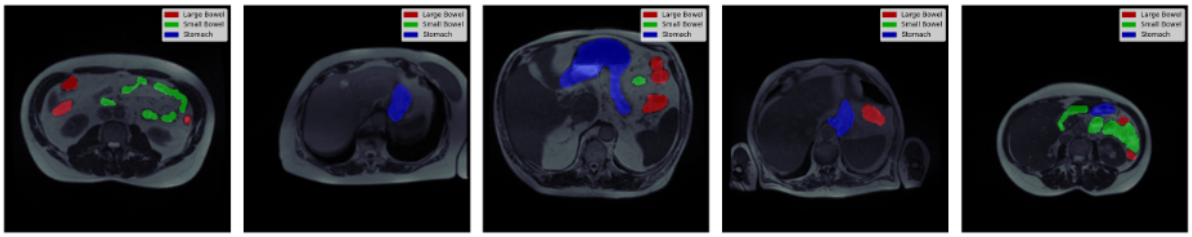

Figure 9: Ground Truth segmentation

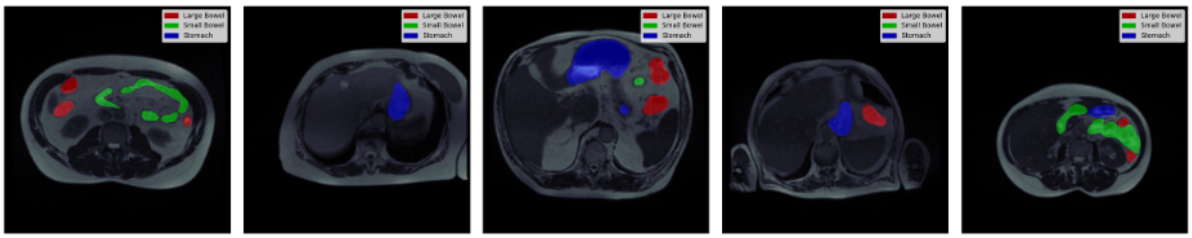

Figure 10: Prediction segmentation

The result shows this model did a good job in this task. Three reasons are important(based on my learning and understanding). First, the Double Conv layer enables it to build on the extracted features to capture more detailed information. Second, there are no dense layers in this model architecture, which reduces parameters, making U-Net less prone to over-fitting (no over-fitting in my training) and computationally efficient. Additionally, the spatial relationships and the location information within the image are preserved, which is crucial for accurate segmentation where the precise localization of objects within an image is necessary. Third, concatenation with the contracting path in the expansive path allows it to use a higher-resolution feature map and capture more information.

4 Model: FCN

The model architecture consists of an encoder network and a decoder network.

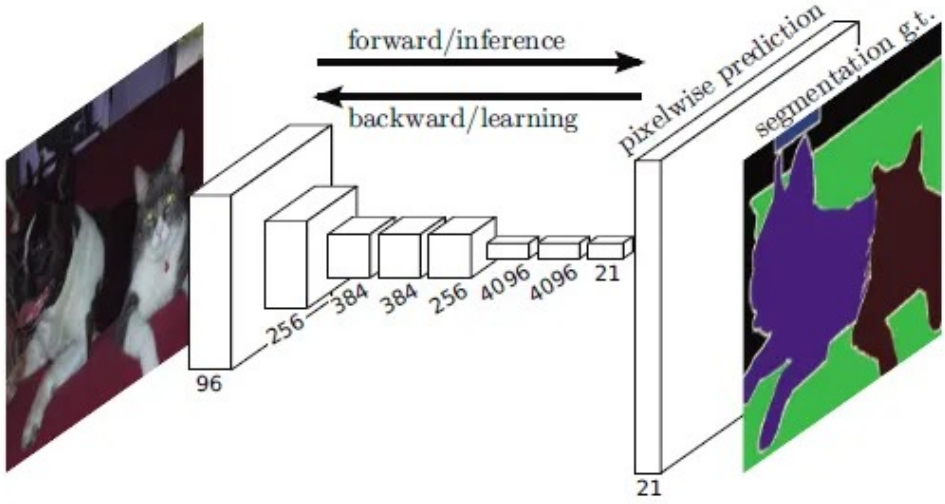

Figure 11: FCN outline

The encoder network is based on a pre-trained or a custom convolutional neural network (CNN) modified to output multiple-scale feature maps. The decoder network is used for upsampling these feature maps using transposed convolutions and combining them using skip connections, which helps preserve spatial resolution and capture fine-grained details in the segmentation map. Convolution is a process of getting the output size smaller. Thus, the name deconvolution comes from when we want to have upsampling to get the output size larger. It is also called convolution and transposed convolution.

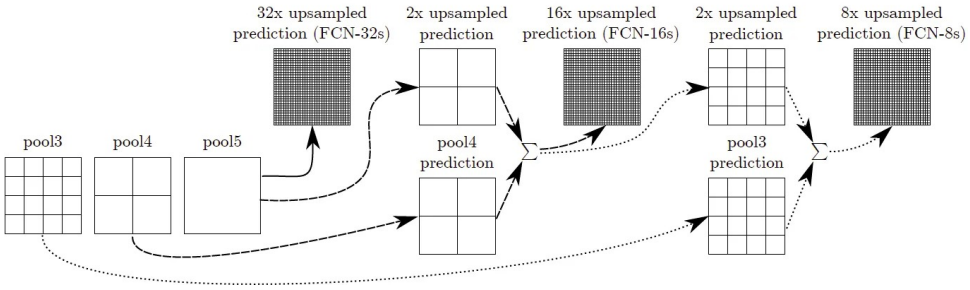

Figure 12: FCN Structures

After going through pool 5, 32× upsampling is done to make the output the same size as the input image. But it also makes the output label map rough. And it is called FCN-32s. This is because deep features can be obtained when going deeper; spatial location information is also lost. That means output from shallower layers has more location information. If we combine both, we can enhance the result. This fusing operation is just like the boosting/ensemble technique used in AlexNet, VGGNet, and GoogLeNet, where the results are added by multiple models to make the prediction more accurate. But in this case, it is done for each pixel, and they are added from the results of different layers within a model.

Transpose Convulation[6]: A transpose convolution is not the same as deconvolution. A deconvolution layer reverses the standard convolution layer to sound similar to transpose. They are alike in one sense, generating the same spatial information. However, only the reverse dimension is transposed, rather than the values of standard convolution. To explain a standard convolution, for a given size of the input (i), kernel (k), padding (p), and stride (s), the size of the output feature map (o) generated is given by

$$\mathrm{o} = \frac{i + 2p - k}{s} + 1 \tag{3}$$

On the other hand, a transposed convolutional layer is usually carried out for upsampling, i.e., to generate an output feature map with a spatial dimension similar to that of the input feature map. For a given size of the input (i), kernel (k), padding (p), and stride (s), the size of the output feature map (o) generated is given by:

$$\mathrm{o} = (i - 1)s + k - 2p \tag{4}$$

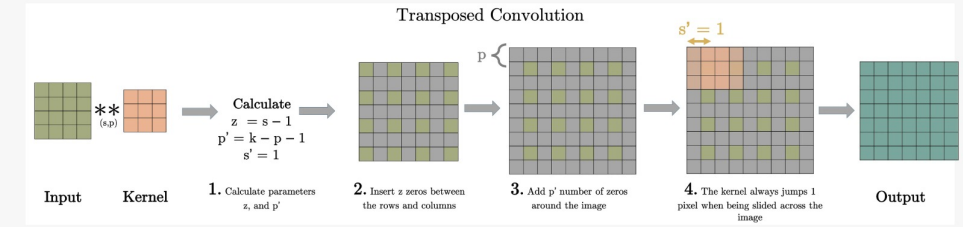

Figure 13: Transpose Convolution

Model Architecture: I first used an FCN8s [7] model taken from GitHub to learn more about the architecture and how it works. The model used has 4 feature sequential layers consisting of convolution and pooling layers. However, an FCN can be implemented using a pretrained CNN combined with upsampling ie transpose layers for image segmentation.

Custom Model: I built a custom model using a pretrained VGG16 coupled with upsampling layers. This enabled me to use transfer learning. This model starts off with a VGG16 layer, then a convolution layer, followed by 4 transpose convolution layers, ending with a convolution layer to make the size the same as the input.

Criterion: To calculate the metric performance I used a combination of two losses namely DICE loss and IOU loss. Both the losses were equally weighed to achieve the desired results.

IoU Loss:[8] The equation for IoU loss is:

$$\mathrm{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{5}$$

The area is calculated as follows. The red box is the target mask given in the dataset, whereas the green mask is the predicted region mask by the model.

Results: These are two plots for the FCN image segmentation. The first one is the actual targeted image, and the second one is masked with the predicted layer by the model.

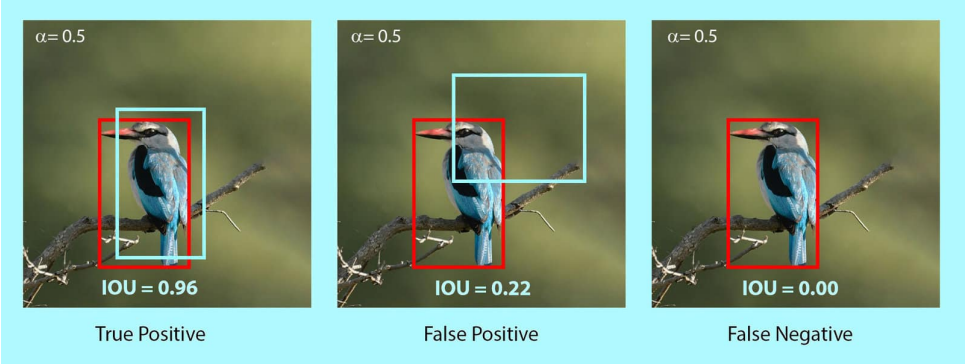

Figure 14: IoU Loss

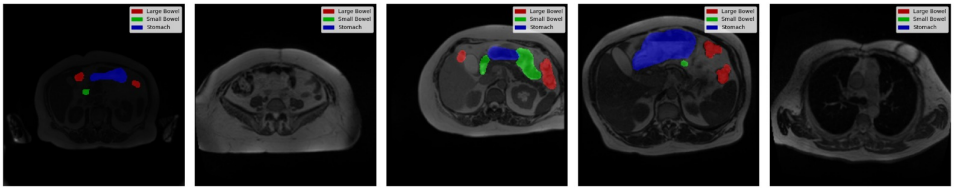

Figure 15: Target

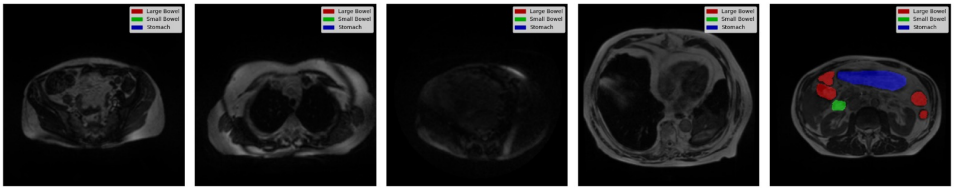

Figure 16: Prediction

5 Conclusion

In this project, we explored two popular deep learning architectures, U-Net and Fully Convolutional Networks (FCNs), for multi-organ image segmentation in medical imaging. Through analyzing these models, we gained valuable insights into their architectures, functionalities, and performance characteristics in the context of biomedical image analysis.

U-Net, with its symmetric encoder-decoder architecture and skip connections, demonstrated impressive capabilities in accurately segmenting anatomical structures within medical images. Its ability to capture context and spatial information, combined with the absence of dense layers, contributed to its effectiveness in localizing objects with high precision while avoiding overfitting.

On the other hand, the FCN model, with its encoder-decoder structure and transposed convolutional layers, provided an alternative approach to image segmentation. While similar in concept to U-Net, FCNs utilize upsampling techniques to recover spatial resolution and capture fine-grained details in segmentation maps. By combining features from multiple layers through skip connections, FCNs preserve spatial information and produce high-quality segmentations.

References

[1] P. C. P. Y. S. L. L. happyharrycn, Maggie, "Uw-madison gi tract image segmentation," 2022. [Online]. Available: https://kaggle.com/competitions/uw-madison-gi-tract-image-segmentation

[2] P. Pinto, "Run-length encode and decode," https://www.kaggle.com/code/paulorzp/run-length-encode-and-decode, 2017, accessed: 2023-05-03.

[3] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*. Springer, 2015, pp. 234–241.

[4] Milesial, "Pytorch-unet," https://github.com/milesial/Pytorch-UNet, 2020.

[5] S. E. Saraki, "UW-Madison GI Tract Image Segmentation Train EDA," 2023, accessed: 2024-05-03. [Online]. Available: https://www.kaggle.com/code/sabahesaraki/uwmgi-train-eda

[6] A. Anwar, "Transpose convolution," https://towardsdatascience.com/what-is-transposed-convolutional-layer-40e5e6e31c11/, 2020.

[7] Zeng, "Fcn8s," https://github.com/SJTUzhanglj/FCN/blob/master/model.py/, 2017.

[8] Prakash, "Iou loss," https://learnopencv.com/iou-loss-functions-object-detection/, 2023.