UMadison GI Tract Image Segmentation Challenge

Kenny Wang **Simar Atwal** **Sruti Dondapati** **Srikar Chunduri**

Abstract

Cancers of the gastro-intestinal (GI) tract affect millions of people worldwide. In 2019 alone, an estimated five million individuals were diagnosed with a GI-related cancer, and nearly half of these patients require radiation therapy as part of their treatment. A major challenge in radiation oncology is getting high doses of radiation to tumors while protecting nearby organs at risk, particularly the stomach, small bowel, and large bowel. These organs can shift significantly from day to day due to normal physiological changes or any regular activity, making precise targeting essential for radiation oncology.

1 Introduction

Before radiation can be delivered to the tumors of cancaer patients, physicians must manually outline (segment) the stomach and intestines on each MRI scan to ensure accurate targeting and safely deliver a patients dose. This manual segmentation process is extremely time-consuming, often extending a typical 15-minute appointment to nearly an hour. For patients who must undergo treatment daily for several weeks, and this time can make therapy more difficult for patients to undergo. A segmentation model that could correctly segment out the important organs, would dramatically reduce the burden for oncologists. By quickly and accurately identifying the stomach and intestines on MRI scans, deep learning models could help radiation oncologists adapt treatment plans more efficiently, shorten patient sessions, and essentially make the patient more comfortable.

This, is the premise for our project and the rationale for training a reliable, and accurate model for segmentation.

2 Approach

2.1 Data Processing:

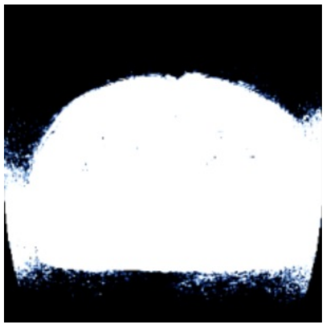

Figure 1: CT scan example

The dataset used for this project is UW Madison's GI Tract Segmentation Kaggle competition's dataset. The data's training examples have 2D CT scans of upper bodies, with up to 3 organ masks, namely: stomach, large bowel and small bowel.

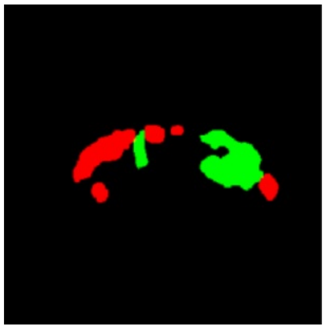
Figure 2: Valid segmentation mask example

Before preparing the data for model training, we examined the full dataset of 30,796 CT slices and discovered that a large portion of them did not contain any valid segmentation masks. Specifically, 17,507 slices (over half the dataset) had empty or missing masks for all three organs. These do not contribute meaningful information for supervised segmentation, mainly because the model would learn to predict empty masks. Therefore, we removed these cases and proceeded with the remaining 13,289 CT scans that contained at least one valid organ mask.

To prepare the dataset for training, we first did RLE Decoding of the RLE string for each organ in an image, decoding it into a binary mask. Then we stacked the 3 organ masks into an array, (H, W, 3). Then the next thing we did was resize all the CT scans (which all varied in image size) into 320x320 sized images. We then normalized all the 16-bit grayscale CT scans to values between 0 and 1, to help train the model more efficiently. We split the dataset into training sets and validation sets for the sake of performance evaluation (which we will get to later). Then for the final step of preprocessing, we converted our masks of the format (H, W, 3) into the PyTorch specific format, namely (3, H, W). This is to match PyTorch's (B, C, H, W) format.

Finally, these preprocessed tensors were then passed into our UNet and TransUNet models for training

2.2 U-Net:

U-Net is a standard convolutional neural-network architecture. It is designed specifically for biomedical image segmentation. Its primary advantage is its "U-shaped" structure: How U-Net works is - We use a contracting path (the encoder) to extract features from the input images. ReLU accomplishes this. But then we use MaxPooling to reduce the image sizes by half in this case, to support the encoder and help it learn features. The encoder in this case is working like a standard CNN classifier, but for segmentation.

The expanding path (the decoder) reconstructs spatial details of images to generate a segmentation mask. The decoder block here works, almost the exact opposite of the way the contracting path did. We use transposed convolution to double the image size (opposite of MaxPool), and at each stage the feature map is concatenated with the corresponding feature map from the encoder. These skip connections pass high resolution features directly from the e encoder to the decoder, and help the model generate sharper masks. For this project, we implemented U-Net from scratch using PyTorch: Loss function: BCEWithLogitsLoss, Optimizer: Adam, Evaluation metrics: Dice coefficient (per organ + mean), and visualizations

The U-Net model served as our baseline architecture, providing a reliable reference point before trying Trans U-Net architecture.

2.3 Trans U-Net:

Due to the nature of the dataset, where organs can span across large portions of the images, we hypothesized that a Trans U-Net, with its self attention mechanisms could outperform the U-Net architecture.

The Trans U-Net is a segmentation model that utilizes a CNN encoder, transformer blocks, and a CNN decoder. The CNN encoder allows the model to extract local features from the training data while the transformer blocks capture long-range dependencies through self-attention. Its architecture is based on the U-Net (as described above) but it combines it with transformers.

The encoder of the Trans U-Net is what sets it apart from a typical U-Net. While U-Net is pure convolution, Trans U-Net has a transformer bottleneck. The CNN is shallower because its job is to create a feature map that will be passed onto the transformers. A series of transformer encoder blocks, each with multi-head self-attention, feed-forward layers, and residual connections, model global dependencies across the image. Then it is reshaped to a 2D feature map to be passed to the decoder section– which mirrors the U-Net implementation.

For this model we used: Loss function: DiceLoss, Optimizer: AdamW, Evaluation metrics: Dice coefficient (per organ + mean), and visualizations

3 Results

3.1 Dice Coefficient:

The Dice coefficients were used for evaluating the model due to its ability to measure how accurately the model's predicted masks were to the actual masks. Each dice coefficient has a range from 0 to 1 where 1 indicates an accurate overlap and 0 is no overlap. The dice coefficient was chosen due to its accuracy for this specific project where organs are relatively small compared to the entire image, thus dice compares the areas of overlap ignoring the imbalance data.

3.2 Loss Curves:

To calculate the loss, a Dice Loss function was created to compute the Dice Loss = 1 - Dice coefficient. This measures the region overlap, being more stable for small organs without relying on pixel counting.

3.3 Visualizations:

To better understand the model performance beyond numerical metrics, the models were compared based on Dice Score distribution, IoU comparison and using an overall scatter plot.

Figure 3 represents the difference in performance between both the models by highlighting regions of similar performance and areas of distinct performance. Particularly, the red diagonal line represents areas where both models performed equally, while the points above the diagonal highlight a better performance for the Trans U-Net model, and points below the diagonal a better performance of U-Net.

Figure 4 highlights the number of images the models performed better on, which in this case, U-Net shows a distinctly higher performance on around 3000 images while Trans U-Net only performed well on around 300 images.

It was found that the Trans U-Net model had a tendency to overfit very quickly, leading to high test set accuracy but a significantly (10-20%) lower validation set accuracy. In order to fix the overfitting problem, we tried several approaches. First was using a pre-trained model and fine tuning on this dataset but we quickly realized that the training time was not feasible, it would go over the 9 hour benchmark. Then, we changed the loss function and altered albumentations. However, all of these pathways still led to a worse performance when compared to U-Net. Our first model was not only more efficient but it also trained well on this small dataset. We came to the conclusion that Trans U-Net would not be able to outperform U-Net without significantly greater computational power and time.

Overall, the U-Net model proved to have better performance resulting in a mean Dice score of around 0.13 while the Trans U-Net model had a mean Dice score of 0.092. Though scores for both the models are relatively low, the U-Net had a higher dice score, proving it is a better model for this given segmentation.

4 Analysis

The U-Net model provided a better overall segmentation performance for the given dataset due to various reasons. The main reason U-Net performed better was due to the dataset size and complexity, for which Trans U-Net requires a larger dataset for training to leverage its transformer model. This dataset was simply too limited for a Trans U-Net model to train itself off of, which allowed for a less complex and convolution based model like U-Net was able to generalize better. The fact that the dataset was halved and some due to a lack of valid segmentation masks in the dataset also contributes to the fact that the dataset was too limited to train the Trans U-Net model. Furthermore, U-Net has better spatial detail preservation due to its encoder decoder architecture which has skip connections, as explained earlier. These skip connections allow the model to preserve the fine grained details, in turn producing more efficient masks. The U-Net is also easier to train since it is less sensitive to hyperparameters and converges more consistently while Trans U-Net can become unstable during training.

4.1 Future Work

Some important things we would have to consider in the future when attempting another similar segmentation problem is possibly filtering the raw data more. We could preserve the aspect ratio instead of stretching the image into a square, sample some empty-mask slices instead of getting rid of all of them, or simply add neighboring slices as extra input channels to give the model context.

Another important point to note is that, next time we might consider trying to find a bigger dataset, if possible, so that the Trans U-Net model learns better.

5 Citations, figures, tables, references

5.1 Figures

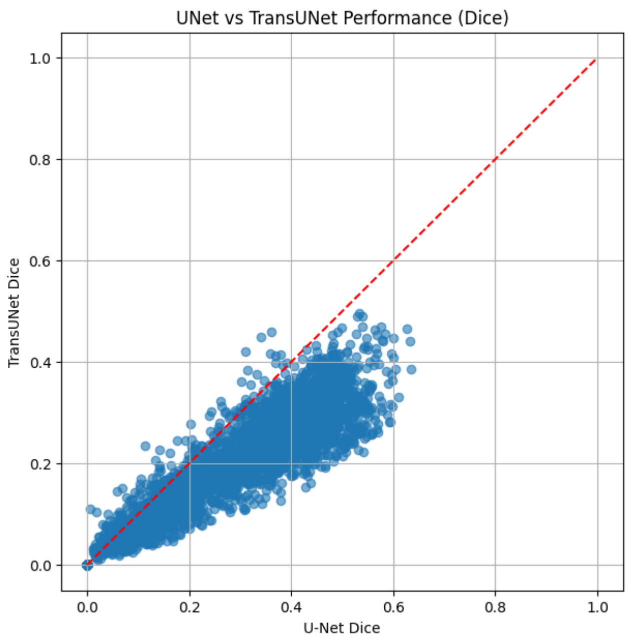

Figure 3: U-Net vs Trans U-Net Dice Performance

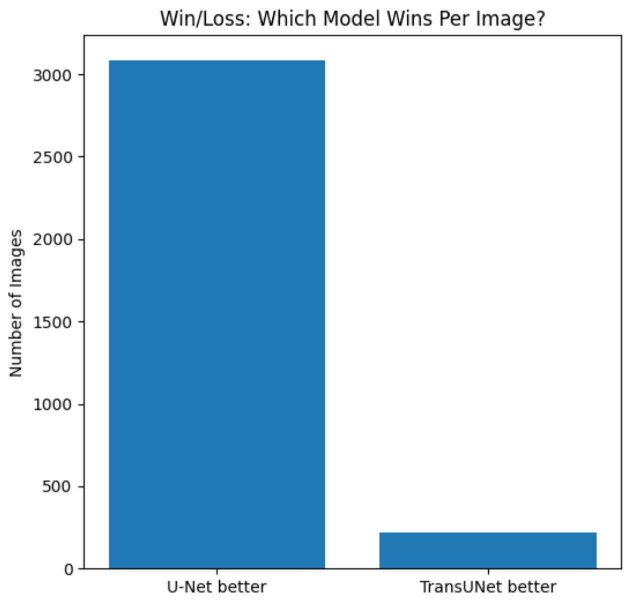

Figure 4: U-Net vs Trans U-Net Win/Loss

References

[1] UW–Madison, Kaggle. *UW-Madison GI Tract Image Segmentation*. 2022. `https://www.kaggle.com/competitions/uw-madison-gi-tract-image-segmentation`.

[2] Figure 3. Lee, S.L., Yadav, P., Li, Y., Meudt, J.J., Strang, J., Hebel, D., Alfson, A., Olson, S.J., Kruser, T.R., Smilowitz, J.B., Borchert, K., Loritz, B., Gharzai, L., Karimpour, S., Bayouth, J., & Bassetti, M.F. (2024). Dataset for gastrointestinal tract segmentation on serial MRIs for abdominal tumor radiotherapy. *Data in Brief, 57*, 111159. `https://doi.org/10.1016/j.dib.2024.111159`.

[3] Figure 4. Child, R., Gray, S., Radford, A., & Sutskever, I. (2019). Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*.

[4] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440.