Utilizing Unlabeled Data in Multi-Organ Segmentation for 3D CT Scans

1. Introduction

In the field of medical image segmentation, annotation is laborious and time-consuming and needs high professionalism, leading to a huge gap between the massive amount of data and limited number of labels. In this project, we explore effective ways to alleviate vision model's dependency on labels. In other words, we investigate different method including semi-supervised learning and other methods. Our main work is to perform semi-supervised learning on the dataset using π-Model strategy after preprocessing the abdominal organ dataset. For training of medical images we choose SwinUNetR deep learning model, SwinUNetR uses Swin Transformer for extracting global and local features and U-Net for image segmentation. For labeled data, we train the image to generate pseudo labels, and then with the ground truth label of the image using Dice and Cross-Entropy as the loss function. For training on unlabeled data, we first perform perturbation on the image by rotating, flipping, and offset, and then feed the transformed image into the model to generate the label, and then compute the result with the pseudo label for consistency loss. Our goal is to reduce the loss for both types of training.

We conducted research and read other authors' articles on semi-supervised learning for multi-organ segmentation at the beginning of our project. Zhou[1] et al. proposed Deep Multi-Planar Co-Training (DMPCT) to accomplish the task of multi-organ segmentation of abdominal CT scans by incorporating multi-planar information of sagittal, coronal, and axial planes of unlabeled data during the training process, which combined with the use of Teacher Model, Multi-Planar Fusion Module and Student Model so that the application of Multi-Planar Fusion generates more reliable pseudo labels and reduces the errors occurring in the pseudo-labeling, which helps in the training of better segregation networks. Ultimately, the proposed model outperforms the fully supervised approach by more than 4% on 100 unlabeled and 210 labeled data. Huang[2] et al. cooperatively trained two networks to allow the coupled networks to teach each other on unlabeled organ data, and to mitigate the noise of teaching supervisions between the networks, weighted-average models were used to generate more reliable soft labels, in addition to utilizing a region mask that selectively applies consistency constraint to unlabeled organ regions that require collaborative teaching, thus further improving performance. A final DSC of 83.60% was achieved on the MOBA dataset.

As a standard semi-supervised learning paradigm, we use π-Model to leverage unlabeled data in datasets via consistency training. As universal model: Segment Anything came out, we plan to make use of its strong Generalization and fine tune it to a dedicated medical images model. We expect the fine-tuned SAM performs well on the unlabeled image relying on its generality, but the initial fine-tuning model segmentation quality are not good enough to impede us from taking as pseudo-labels, more details shown in result.

2. Methods

2.1. Dataset Preparation

BTCV is a dataset that includes 50 abdominal 3D CT scans, each labeled with seven organ annotations: duodenum, gallbladder, kidney, liver, pancreas, spleen, and stomach. The 50 scans were captured during the portal venous contrast phase with variable volume sizes ($512 \times 512 \times 85$ to $512 \times 512 \times 198$) and fields of view (approximately $280 \times 280 \times 280$ mm^3 to $500 \times 500 \times 650$ mm^3). The in-plane resolution varies from 0.54×0.54 mm^2 to 0.98×0.98 mm^2, while the slice thickness ranges from 2.5 mm to 5.0 mm. For the purpose of our analysis, we divided the dataset into two distinct subsets: a training set consisting of 30 scans and a test set comprising 17 scans. Considering the specialized format of medical series/exams and the subsequent preprocessing, a dataloader embedded with a certain transformer is prepared.

2.2. Architecture

Compared to supervised learning, semi-supervised learning requires less labeled data and can leverage abundant unlabeled data for improved model performance. In our project, we assumed that a model should output similar predictions for an unlabeled input, even when it is slightly perturbed. Based on that, we used the π-model. The main idea of the π-model is implemented by applying different transformations to the original images, which are assumed to cause few differences in the results, thus underlying the model's consistency robustness.

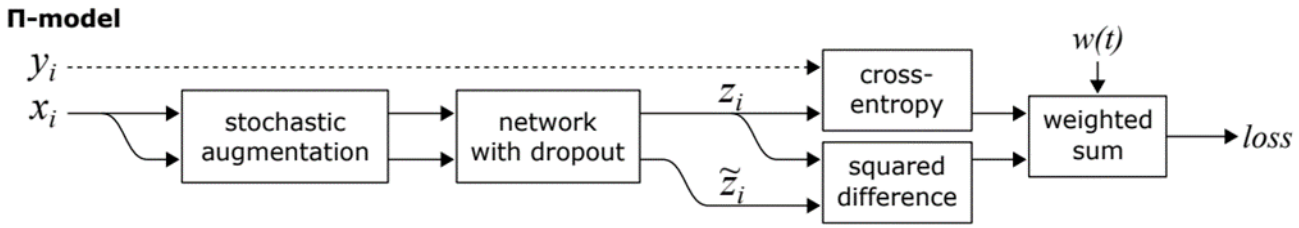

Figure 1 Schematic Diagram of the Pi-Model for Semi-Supervised Learning

For the segmentation task, we implemented SwinUNETR model for medical image segmentation. SwinUNETR is a 3D image segmentation model that combines the Swin Transformer with the U-Net structure. The Swin Transformer is a deep learning model adapted for computer vision tasks. It can handle various scales of images, and it is very efficient in processing high-resolution images. U-Net is a CNN architecture that's widely used for image segmentation tasks. It can capture context and provide precise localization. Specifically designed for 3D medical image segmentation tasks, SwinUNETR utilizes the Swin Transformer to capture complex features of an image and improves the accuracy of segmentation through the U-Net structure.

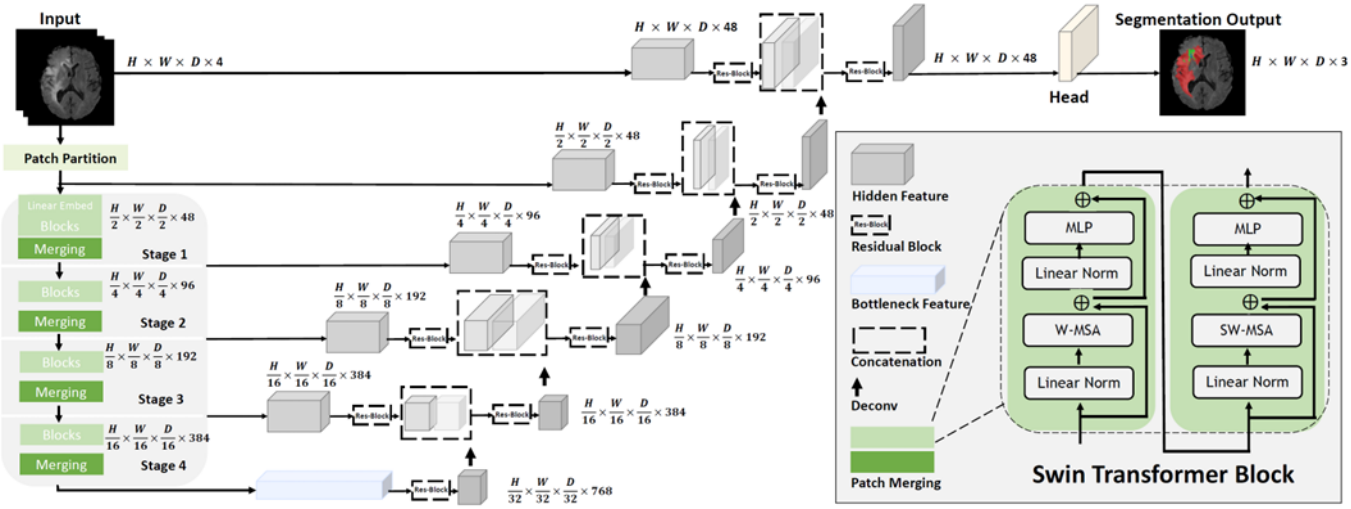

Figure 2 Schematic Diagram of the SwinUNETR Model

2.3. Pipeline

Here, we will introduce the pipeline of our π-Model. Our method is based on one assumption that the model's prediction should be consistent under small perturbations. Therefore, we can transform a CT image and expect the output of the model to be the same as the output of the model without the transformation. In other words, pseudo-labels can be used to supervise the training of unlabeled data. So, the whole workflow is as follows.

The process takes as input CT scans, which are medically important images used for diagnosis and treatment planning. Images are pre-processed using different methods like normalization and changing the brightness. The input images are subjected to data perturbation processes including rotation, flip, and offset. These techniques help the model to generalize better as they simulate the variations that the image may encounter in the real world. The output of the segmentation model is a pseudo label of the CT image that identifies regions of the image with different organizations or structure. This is a way of assessing the model's ability to generalize, ensuring that in real-world applications, the model can make accurate predictions about new and unseen data[7].

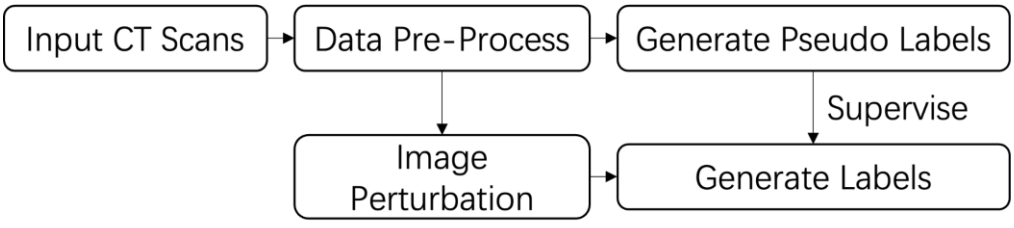

Figure 3 A brief diagram of the Pipeline

2.4. Image Pre-Process

The image pre-processing pipeline involves a series of steps for preparing and augmenting data for deep learning model training. Initially, images and their corresponding labels are loaded and an additional channel dimension is added to accommodate the requirements of deep learning frameworks. Orientation and spatial resolution are standardized across all images to maintain consistency. Intensity normalization is applied to the images, and background regions are cropped out. Spatial padding ensures uniformity in image sizes. For data augmentation, techniques like random cropping, rotational transformations, and intensity shifts are used to enhance the dataset's diversity and robustness. Finally, the processed images and labels are converted into tensor format, making them suitable for training in PyTorch. We aim to standardize the dataset and introduce variety and complexity to the dataset by performing image pre-processing.

```
def get_loader(args):
    train_transforms = Compose(
        [
            LoadImaged(keys=["image","label"]), #0
            AddChanneld(keys=["image", "label"]),
            Orientationd(keys=["image", "label"], axcodes="RAS"),
            Spacingd(
                keys=["image", "label"],
                pixdim=(args.space_x, args.space_y, args.space_z),
                mode=("bilinear", "nearest"),
            ), # process h5 to here
            ScaleIntensityRanged(
                keys=["image"],
                a_min=args.a_min,
                a_max=args.a_max,
                b_min=args.b_min,
                b_max=args.b_max,
                clip=True,
            ),
            CropForegroundd(keys=["image", "label"], source_key="image"),
            SpatialPadd(keys=["image", "label"], spatial_size=(args.roi_x, args.roi_y, args.roi_z), mode='constant'),
            # RandZoomd_select(keys=["image", "label"], prob=0.3, min_zoom=1.3, max_zoom=1.5, mode=['area', 'nearest']), # 7
            RandCropByPosNegLabeld(
                keys=["image", "label"],
                label_key="label",
                spatial_size=(args.roi_x, args.roi_y, args.roi_z), #192, 192, 64
                pos=4,
                neg=1,
                num_samples=args.num_samples,
                image_key="image",
                image_threshold=0,
            ), # 8
```

Figure 4 Preprocessing part of the code schematic

2.5. Training Process

For the training process, as shown in the figure. Our process is as follows. For training with labeled data, we feed the preprocessed images into the SwinUNetR model, and the output of the model is used as the pseudo label. then, it is compared with the ground truth, and the loss function is the sum of Dice and Cross-Entropy, and we use the CTs of 9 patients as the training data. For the training of unlabeled data, we first perturbation the CT, including rotation 90 degrees, flip, offset 5 units, then, input the transformed CT into the SwinUNetR model, and feed the untransformed CT into the model as well, and calculate the consistency loss of the outputs of these two parts. We used CTs from 21 patients as training data. In this way, we could utilize the unlabeled data. We trained 2000 Epochs and the learning rate was set to 0.0001.

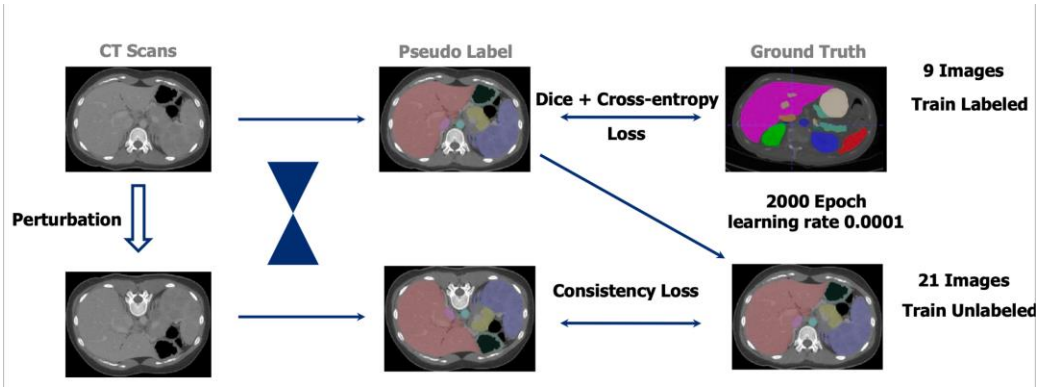

Figure 5 Schematic diagram of the training process

The following is a pseudo-code for applying the Perturbation section to an image

```
Function: generate_consistency_pair
Input:
    - self: object reference
    - roi: region of interest dimensions (width, height, depth)
    - transform_type: the type of transformation to apply; options include
    'random', 'flip', 'rotate', 'offset'

Procedure:
1. If transform_type is 'random':
   a. Randomly select a transform_type from 'flip', 'rotate', 'offset'
   b. Crop the input image to the size of the region of interest (roi)

2. If transform_type is 'flip':
   a. Randomly select a spatial axis (0, 1, or 2)
   b. Apply a flip transformation along the selected axis
   c. For each batch in the input:
      i. Apply the flip transformation to the batch
      ii. Append the transformed batch to a list
   d. Stack the list of transformed batches along the batch dimension

3. If transform_type is 'rotate':
   a. Randomly select two spatial axes without replacement
   b. Apply a 90-degree rotation along the selected axes
   c. Define the inverse rotation (rotation by -90 degrees along the same
   axes)
   d. For each batch in the input:
      i. Apply the rotation to the batch
      ii. Append the rotated batch to a list
   e. Stack the list of rotated batches along the batch dimension

4. If transform_type is 'offset':
   a. Randomly select a spatial axis from 2, 3, or 4
   b. Define an offset value (e.g., 5 pixels)
   c. If the selected axis is 2 (width):
      i. Apply the offset to the input image
      ii. Define the inverse transformation as an array with the
      negative offset

Return:
   - A pair of images: the original and the transformed image
   - The inverse transformation for the applied transformation
```

Figure 6 Pseudo-code for applying the Perturbation part to the image

In our research, we utilize a deep learning framework for the semantic segmentation of CT scans. The training process begins with loading batches of labeled CT images, where each batch includes the image data, segmentation labels, and associated filenames. The segmentation labels are encoded into one-hot vectors for multi-class segmentation tasks. The model then predicts the segmentation map for each image, which is compared to the ground truth using a combined loss function that incorporates both Dice loss and Binary Cross-Entropy loss to ensure accurate boundary delineation and class imbalance handling. Backpropagation is performed to minimize this combined loss, updating the model parameters via gradient descent. To monitor training progress, we log the average Dice and BCE losses after each epoch, ensuring consistent improvement over 2000 epochs with a learning rate of 0.0001. Additionally, we introduce perturbations in the training set to test the robustness of our model against variations in the input data. This is complemented by a consistency loss mechanism that encourages the model to predict similar segmentations for perturbed versions of the same image, enhancing its generalization capabilities. We train our model using both labeled and unlabeled data to leverage the unlabeled data through a semi-supervised learning approach, further improving the model's performance.

```python
def train_iter_labeled(args, train_loader, model, optimizer, loss_seg_DICE, loss_seg_CE,step,loss_bce_ave,loss_dice_ave):
    batch = next(train_loader)
    x, lbl, name = batch["image"].to(args.device), batch["label"].float(), batch['name']
    B, C, W, H, D = lbl.shape
    y = torch.zeros(B,NUM_CLASS,W,H,D)
    for b in range(B):
        for src,tgt in enumerate(TEMPLATE['01']):
            y[b][src][lbl[b][0]==tgt] = 1
    y = y.to(args.device)
    logit_map = model(x)
    term_seg_Dice = loss_seg_DICE.forward(logit_map, y, name, TEMPLATE)
    term_seg_BCE = loss_seg_CE.forward(logit_map, y, name, TEMPLATE)
    loss = term_seg_BCE + term_seg_Dice
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
    if args.local_rank == 0:
        tqdm.write(
            "Epoch=%d: Training (%d / %d Steps) (dice_loss=%2.5f, bce_loss=%2.5f)" % (
                args.epoch, step, len(train_loader), term_seg_Dice.item(), term_seg_BCE.item())
        )
    loss_bce_ave += term_seg_BCE.item()
    loss_dice_ave += term_seg_Dice.item()

    return loss_bce_ave, loss_dice_ave
```

Figure 7 Schematic code for training labeled data

```python
def train_iter_unlabeled(args, train_loader_unlabeled, model, optimizer,loss_consistency,step,loss_consis_avg):
    batch = next(train_loader_unlabeled)
    x,name = batch["image"].to(args.device), batch['name']
    x, x_t, transform = loss_consistency.generate_consistency_pair(x, (args.roi_x, args.roi_y, args.roi_z))
    with torch.no_grad():
        logit_map = model(x)
    logit_map_t = model(x_t)
    term_consis = loss_consistency.forward(logit_map, logit_map_t, transform)
    loss = term_consis * args.alpha
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()
    if args.local_rank == 0:
        tqdm.write(
            "Epoch=%d: Training (%d / %d Steps) (consistency_loss=%2.5f)" % (
                args.epoch, step, len(train_loader_unlabeled), term_consis.item())
        )
    loss_consis_avg += term_consis.item()

    return  loss_consis_avg
```

Figure 8 Schematic code for training unlabeled data

2.6. Loss Function

As for the loss function, basically we want the model to optimize both segmentation quality and pixel-level classification performance. We use dice loss plus multi-label binary cross-entropy loss to calculate the loss of training labeled image process. Combining these two loss functions allows the model to optimize both overall segmentation quality and pixel level classification performance. And we use MSE loss in the training unlabeled image process.

$$Dice\ Loss\ = 1 - \frac{1}{N}\sum_{i=1}^{N} \frac{2 \times |X_i \cap Y_i|}{|X_i| + |Y_i|}$$

$$Multi\ Label\ Binary\ Cross\ Entropy\ \ Loss\ = \frac{1}{N}\sum_{i=1}^{N} \sum_{j=1}^{M} [y_{ij} \cdot log(p_{ij}) + (1 - y_{ij}) \cdot log(1 - p_{ij})]$$

$$Mean\ Squared\ Error\ = \frac{1}{n}\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2$$

2.7. Other Training Techniques

For the optimizer we use Adam Weight Decay(AdamW)[8] to separate the weight decay part, hoping for better training

stability and generalization ability. This approach helps to reduce overfitting problem and may provide better generalization and parameter stability in not very large dataset. In addition, we set up a learning rate scheduler LinearWarmupCosineAnnealingLR, which contains Linear Warmup and Cosine Annealing. During the warmup phase, the learning rate gradually increases. We prefer to conduct a rapid exploration of the parameter space in the early stages of training, aiming to escape local minima and enhance learning efficiency. After the warmup phase, the learning rate decreases gradually in the form of a cosine function. It is hoped that the better solution space can be approached quickly in the early stages of training, while in the later stages, it can be finely tuned to find a better solution and improve stability.

2.8. Software

Pytorch with some other Python libraries like monai, model.SwinUNETR partial, dataset.dataloader, utils etc. Also, for training we use multiple gpu's for distributed training.

3. Results

Two metrics are used during the validation to select the best epoch and to evaluate the result. From the left figure below, the DICE metric measures the overlapped region by each slice between the prediction and the ground truth. Horizontally, by different organs, Kidney, Liver and Spleen showed high averages over 90% overlapped with ground truth with a small deviation across the validation dataset. However, Duodenum and Gallbladder exhibited relatively poor results with only 46% and 68% respectively with a significant deviation as shown by the boxplot. Vertically, separated by perturbation method, Rotate and Offset demonstrated superior results over Flip and Benchmark. A supervised training with only 30% labeled data in the training dataset was chosen as the Benchmark. Between the Rotate and the Benchmark, a two-sample t-test was performed to test the difference between the mean values and the result showed that the mean of Rotate statistically increase from the Benchmark by nearly 10%.

From the right figure below, the Hausdorff distance metric measures the maximum distance between two contours. Each slice was evaluated and the average of each CT exam was calculated as one data point as shown in the boxplot. Instead of the maximum Hausdorff distance, a 95 percentile was implemented on account of its nature to be affected by noises. Horizontally, Kidney, Liver, and Spleen showed highly consistent result with DICE over the other organs. Pancreas showed better result compared to stomach, which is different from that of DICE. The reason is that the lower abdominal part of stomach sometimes failed to segment as shown in the figure below. By different perturbations, the Rotate still has statically improvement from the Benchmark.

In comparison with the initial goals, the SAM was not integrated into our training because so far it cannot provide good enough pseudo-label for semi-supervised training. From the figure below, it can be shown that the predicted segmentation of SAM on liver cannot correctly capture the complicated boundary and showed a very rough boundary with noise in comparison with the ground truth, which contributed to an as high as 27.6 of Hausdorff distance compared to an average of 5.76 of our Benchmark. The reason might be that medical image often contain subtle features and patterns that are crucial for accurate diagnosis but might be missed by a generalized segmentation model not specifically trained on medical data.

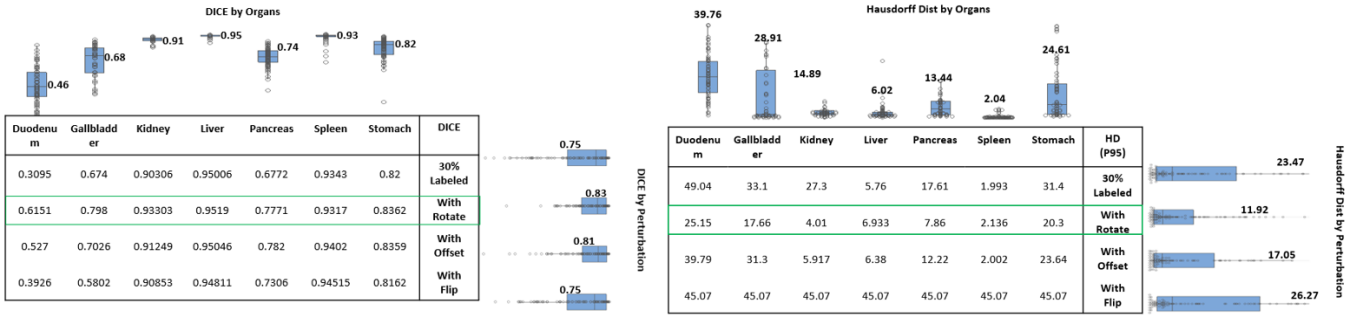

Figure 9 Dice result and distribution by different organs and perturbations (left); Hausdorff distance result and distribution by different organs and perturbations (right)

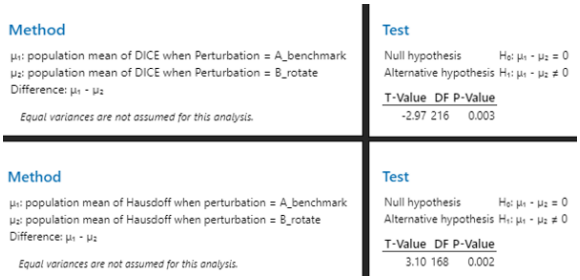

Figure 10 Two-Sample T-Test for DICE (above) and Hausdorff Distance (below)

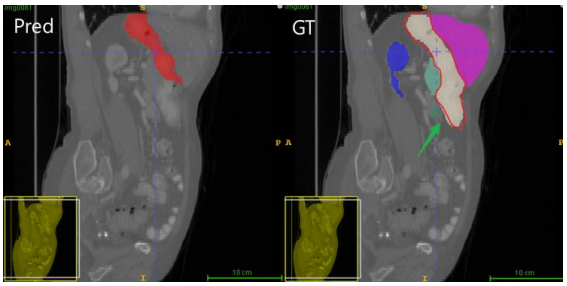

Figure 11 Stomach segmentation prediction (left); Stomach segmentation ground truth (right)

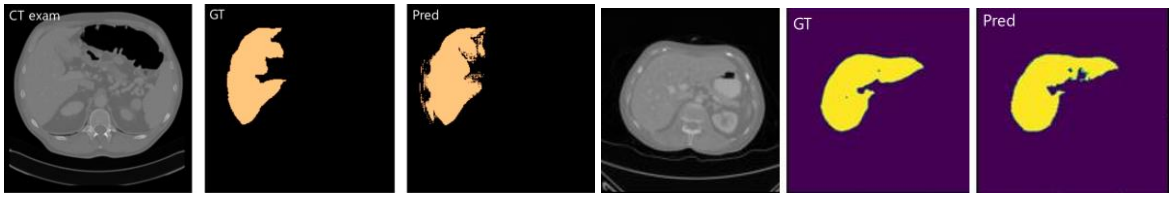

Figure 12 Fine-tuned Segment Anything Model (SAM) result comparison between prediction and ground truth (GT) (Left); Benchmark result comparison between prediction and ground truth (Right)

4. Discussion

From the figure below of the two problematic organs, duodenum is discontinuous across slices. Its position and shape also change dramatically across slices and even more obvious when compared to liver. The reason is two-folded. There is a natural variation in the anatomy of the duodenum among different individuals. This variability can be quite significant, leading to differences in the shape and position of the duodenum as seen on CT scans. Another aspect is that the position of the patient during the CT scan (Supine, Prone, Lateral Recumbent etc.) can affect the appearance of the duodenum. For example, lying down versus standing up can change the position of abdominal organs due to gravity. Hence, it might need more data by different kinds of CT scan duodenum to train the model. Another problematic organ is the gallbladder, and it is the smallest organ in our prediction. Its average size is 5cm and only 5-7 slices to describe it in the dataset, which is much less than that of liver and spleen (50, 20 respectively). The lack of adequate information might be the same making gallbladder hard to predict. To avoid abuse of radiation, the slice thickness ranges from 2.5 mm to 5.0 mm in BTCV dataset. However, the modern CT can scan down to a 0.4mm thickness. For more information, we can train the model specific for gallbladder with more data and specifically smaller slice thickness.

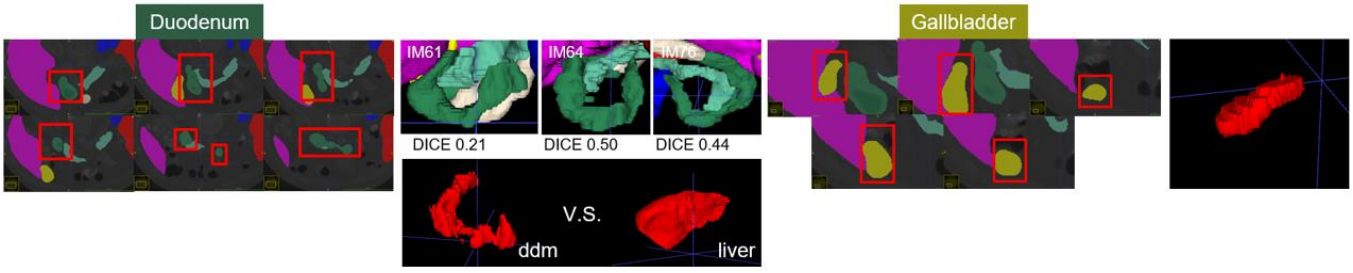

Figure 13 Problematic organ duodenum (left); Problematic organ gallbladder (right)

Another intriguing finding is that all models showed less accurate segmentation on IMG69 as shown in the figure below. The reason is necessary information is cropped out by the pre-processing composer. The histogram depicted the intensity distribution across the validation dataset and IMG006 was apparently more discursive, so the intensity selection bounded box as drawn in red rectangle left its area out of the selection, which leads to missing necessary information. The cut-off values hard-cored in the intensity selection function were based on the common abdominal organ intensity distribution [5].

As for comparison with others's work, Rotation as an input perturbation improves the performance of multi-organ segmentation by 1.32% compared to the benchmark. Currently, the highest organ nsd is 0.9574, still a gap.

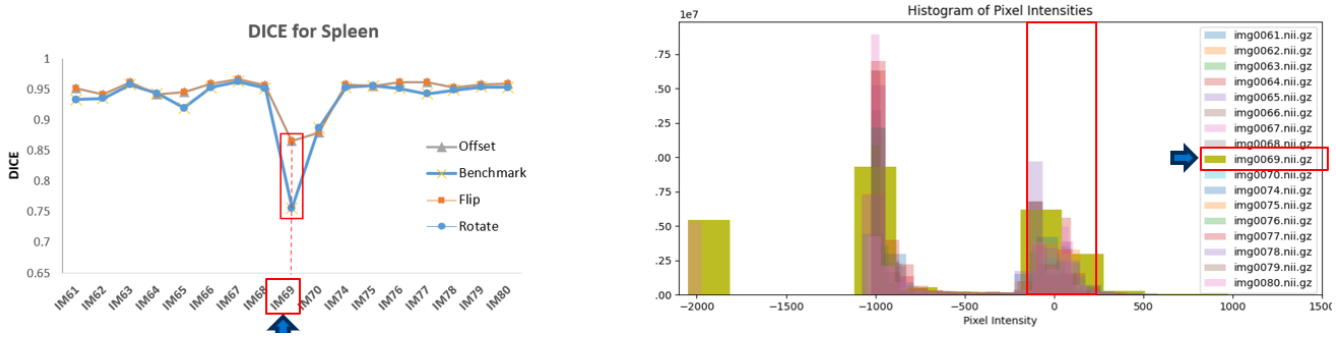

Figure 14 Validation DICE score by different perturbations on IMG0069 (left); Intensity distribution of validation dataset (right)

5. Conclusion and Future Improvements

In conclusion, our semi-supervised training model demonstrated some advancement especially on Kidney, Liver, and Spleen. In the meantime, at the current stage it also showed limitations on Duodenum and Gallbladder. In the next stage, to compete with a top-torch result (claim 0.9164 organ DSC on 08/08/2023), we can try training with base SwinUneTr model [6] and which also means more data is needed. If the dataset can be provided with the metainformation (most of the time, it is erased considering personal privacy) e.g., slice thickness, radiation dosage, position of patient, more useful information can be embedded during training for a more generalized and adaptive model.

Another task to do is to make an adaptive intensity selection function in the pre-processing composer. Instead of hardcore upper and lower limits, it can change the values of limit adapting different dataset, which we remise can significantly avoid information loss in the pre-processing stage.

We also want to refine the quality of the pseudo label. For SAM prediction pseudo label, the major issue is with rough boundary. A mask can be generated based on the uncertainty map. The noise surrounding the boundary is obviously high and with higher uncertainty. Or we can implement morphological structuring algorithms like binary erosion to refine the boundary.

6. Reference

1. Y. Y. Zhou, Y. Wang, P. Tang, et al. (2019). Semi-supervised 3D abdominal multi-organ segmentation via deep multi-planar Co-Training. In Proceedings of IEEE Winter Conference on Applications of Computer Vision, 121-140.
2. Hatamizadeh, A., Nath, V., Tang, Y., Yang, D., Roth, H. R., & Xu, D. (2021, September). Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images. In International MICCAI Brainlesion Workshop (pp. 272-284). Cham: Springer International Publishing.
3. R. Huang, Y. J. Zheng, Z. Q. Hu, et al. (2020). Multi-organ segmentation via Co-training weight-averaged models from few-organ datasets. In Proceedings of the 23rd International Conference on Medical Image Computing and Computer Assisted Intervention, 146-155.
4. Mazurowski, M. A., Dong, H., Gu, H., Yang, J., Konz, N., & Zhang, Y. (2023). Segment anything model for medical image analysis: an experimental study. Medical Image Analysis, 89, 102918.
5. https://radiopaedia.org/articles/windowing-ct

6. darragh/swinunetr-btcv-tiny · Hugging Face
7. Y Tang, D Yang, W Li, et al. Self-Supervised Pre-Training of Swin Transformers for 3D Medical Image Analysis.arXiv preprint arXiv:2111.14791v2. 2022.
8. Loshchilov, I., & Hutter, F. Decoupled Weight Decay Regularization. arXiv preprint arXiv:1711.05101. 2017.