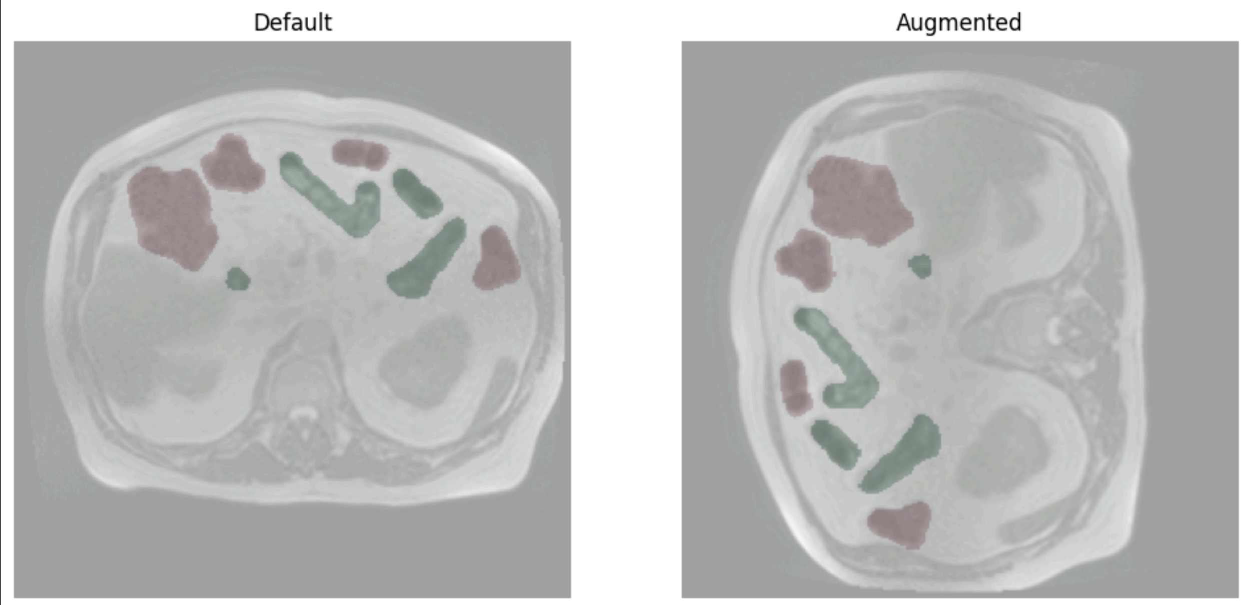
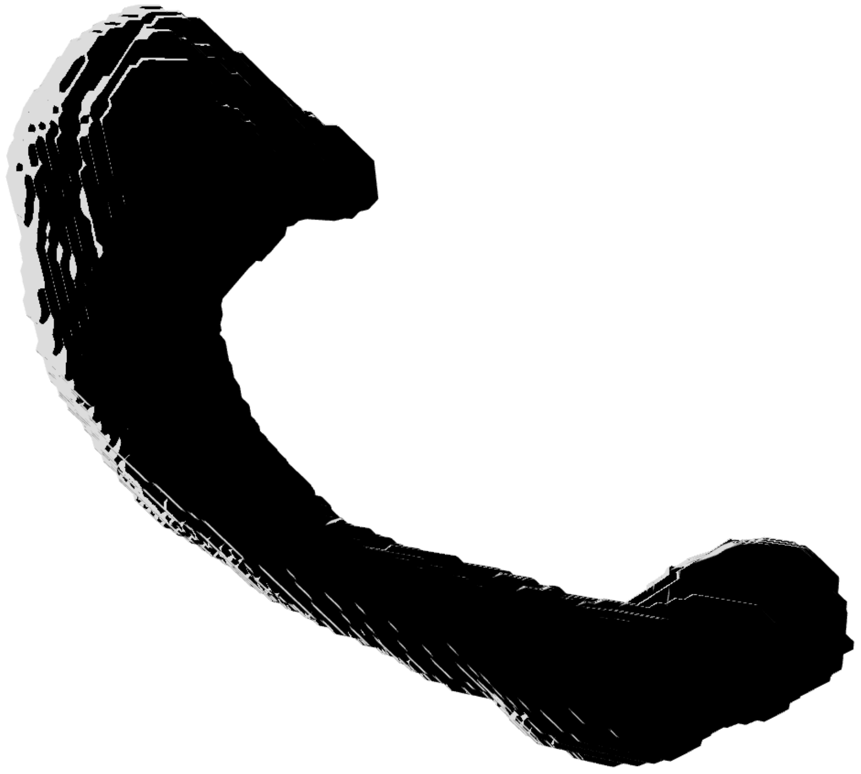
AI HW3 CV Report

Data understanding and preprocessing

I started with loading data. Examining how much data I have. Obviously, I counted number of images, which is 144 per patient, totaling in almost 6000 training samples. Interesting fact, is that there are 12437 empty masks. Dimensions of images are 266×266, which is strange, as UNet usually requires dimensions divisible by 16. I decided do simple preprocessing — scaling colors of image from 0 to 1. As for augmentations, you could see them in a notebook, they were greatly inspired by this notebook https://www.kaggle.com/code/awsaf49/uwmgi-unet-train-pytorch#%F0%9F%94%8D-Optimizer , though I want to emphasize, that horizontal/vertical flips and $90 \cdot k, k \in \mathbb{Z}$ are essential for good model's performance. This way model learns not the positional features, but more "qualitative", in the sense that it learns how organs look, not where they usually located.



I also did 3D visualization of masks, to see whether they look adequate:



Training

For training I used combined loss of DiceLoss and BCE with logits loss, because it helps handling class imbalance and maximizing the overlap of the predicted and ground truth masks, while the BCE loss focuses on pixel-wise accuracy, leading to better overall convergence and segmentation quality. As metrics for evaluation I used Dice Metric (because it's the main metric) and IoU coefficient because it is similar in its definition to Dice coefficient.

I used 4 folded, patient wise split, for OOF validation (5 would train too long, 3 would result in too weak models). My code is structured in such way, so that I can easily run experiments. To try new model all I need is to pass list of models, losses and optimizators to run_training function, and it will train them.

For inference I used soft voting of logits of models.

Also, for training I used 2.5D images (stacked 5 adjacent slices, prediction is done only for central one), because we can safely assume if one slice has organs on it, neighboring slices will have it too.

Inference

I used test time augmentation.

Experiments

Initially I started with the simplest approach: ResNet50 + UNet. It gave good enough performance. By recommendation of Yaroslav Prytula I tried EfficientNet + UNet++.

It indeed gave better performance, becuse EfficientNet provided optimized scaling of depth/width/resolution as the encoder, and **U-Net++'s** nested, dense skip connections to reduce the semantic gap and enhance multi-scale feature fusion.

I also assumed, that attention mightbe helpful for extracting 2.5D information, thus I tried MANet, Mix Transformer + UNet and SegFormer. Unfortunately, I think the amount of data we had is not enough to effectively train transformers, thus they did not yielded good enough results.

Final sumbission

For final submission I took 3 model: ResNet+UNet, EfficientNet + UNet++, MiT+UNet, and did soft weighted voting for logits they provided slightly better results, but I assume improvement is neglectable (smaller than 0.01).

Weights are: EffNet + UNet++ 0.45, ResNet + UNet 0.35, MiT + UNet 0.2. Weights are choosen by assigning the biggest weight to the best performing model.

MedDINOv3

Setup on apple silicon macbooks is impossible. Setting it up on kaggle or colab's notebook is also imposible, because to setup MedDINOv3 you need access to terminal.