3rd Palce Solution of SenNet + HOA - Hacking the Human Vasculature in 3D

1 About me

1.1 Cometition & Score

- Competition Name: **SenNet + HOA - Hacking the Human Vasculature in 3D**
- Team Name: **ForcewithMe**
- Private Leaderboard Score: **0.727912**
- Private Leaderboard Rank: **3/1149**

1.2 Team Members (Solo)

- I am Hongjian Song (Kaggle ID: forcewithme), a Kaggle GrandMaster from Guangdong Province of China, whose email address is vadaforce@163.com. I am interested in computer vision and image segmentation. I like competing on Kaggle, and I have won 6 gold medals, including 2 solo gold medals on Kaggle until now.

2 Summary

My approach was strikingly straightforward, relying solely on 2D models and only utilizing smp (segmentation models pytorch) and timm (pytorch image models) in the whole training and inference pipeline. Here are the three highest-priority key points in my solution:

1) Refining labels from sparse to dense.
2) Emulating the magnification factor of the test set.
3) Maintaining an appropriate resolution.

3 Solution

3.1 Training pipeline: From Sparse to Dense

The training code is released at a kaggle dataset. Given that half of the training set has dense labels (kidney 1, kidney 3 dense), and the other half was sparse, utilizing dense labels to refine sparse ones was a crucial step. The overall process entailed:

1) Training UNet(maxViT512) and UNet(EfficientNetv2s) using kidney 1 and kidney 3 dense.
2) Generating supplemental labels for kidney 3 sparse using the trained UNet maxViT512 and UNet EfficientNetv2s models.
3) Resuming the training of UNet maxViT512 and UNet EfficientNetv2s for a few epochs with kidney 1, kidney 3 (dense, sparse plus supplemental labels).
4) Repeating the step2 on kidney 2.
5) Training three UNet models (with EfficientNetv2s, SeResNext101, MaxViT512) and one UNet++ using all real labels from all kidney plus pseudo labels.

Note: As the organizers disclosed the proportion of annotations within kidney 3 and kidney 2, I endeavored to select thresholds based on pixel quantity as close as possible to the official proportion when choosing threshold values for pseudo label.

3.2 Inference pipeline

The reproduction of the 2 final submissions is at link1 and link2, with all model weights published. My final submissions were a single model of MaxViT and a ensemble of the four models. Surprisingly, both submissions scored same at 0.727. I did not use any form of weighting and MaxViT only constituted a quarter of the ensemble submission, but their scores were totally the same on private LB.

Tabel 3.2.1 details of the 4 models included in the 2 final submissions

Model	**Backbone**	**Resolution**	**public**	**private**
UNet	MaxViT-Large 512	512	0.846	0.727(submission1)
UNet	SeResNext	512	0.819	0.753
UNet	Efficiennet_v2_s	448, 832	0.799	0.703
UNet++	Efficiennet_v2_l	512	0.817	0.692
ensemble	-	-	0.846	0.727(submission2)

Even more astonishing was that the single-model score of SeResNext on private LB turned out to be the highest. Probing with different resolution and threshold several times, it can achieve 0.78 on private LB. These results can be reproduced with the weights and code in any of the 2 notebooks introduced above.

Tabel 3.2.1 UNet-SeResNext101 Perform the best on Private LB

Resolution	**Threshold**	**public**	**private**
512	0.2	0.820	0.753

512	0.15	0.816	0.757
512	0.1	0.811	0.766
384+512	0.1	0.807	0.778
384+512	0.12	0.811	0.775
384+512	0.08	0.801	0.780

3.3 Emulating the Magnification of the Private Test Set

A pivotal reason for my decision to participate in this competition was the disclosure of the magnification factors for the training and test sets by the hosts. The training set had a magnification of 50um/voxel, the public test set was the same at 50um/voxel, while the private test set was at 63um/voxel. A larger magnification factor implies a lower resolution. For instance, a 600um object would occupy 12 pixels in both the training and public sets, but only 10 pixels in the validation set. Hence, during training, I set the scaling center to 0.8, rather than 1, with a scaling range of 0.55 to 1.05, to simulate the private test set.

3.4 Maintaining an Appropriate Resolution

During the early stages of the competition, whether validating on kidney 2 or kidney 3, I observed that if I trained for 20 epochs, after the initial few epochs, the dice coefficient (not surface dice) variation on the validation set was very minimal, with the MaxVit512 large model exhibiting the least fluctuation. Considering that we only had three kidneys, I decided to train on all kidneys directly after completing the pseudo labeling process, given the stability in convergence.

3.5 Train on all data if convergence is Stable

During the early stages of the competition, whether validating on kidney 2 or kidney 3, I observed that if I trained for 20 epochs, after the initial few epochs, the dice coefficient (not surface dice) variation on the validation set was very minimal, with the MaxVit512 large model exhibiting the least fluctuation. Considering that we only had three kidneys, I decided to train on all kidneys directly after completing the pseudo labeling process, given the stability in convergence.

3.6 Minimizing the Impact of Threshold Values

I search the best threshold with scripts at the path ‘src/valid_2stage’ in my training code.

The metrics calculation code uses the implementation published by junkoda at kaggle notebooks. My most stable single model was able to maintain very minor fluctuations in the surface dice score (less than 1) within a threshold range of 0.2. After model fusion, the stable threshold range could be potentially in 0.3~ 0.4. A stable threshold is extremely crucial in segmentation competitions. In this competition, as my final model lacked a validation set, I had to utilize thresholds searched with earlier trained models that included a validation set and apply them to the final version of the model. Fortunately, the models trained on the full dataset appeared to possess threshold values very close to those from the earlier models trained with k1+k2 (sparse), and validate on k3. At the same time, the fluctuation of threshold values across kidney 3 dense, public, and private was very small.

3.7 Heavy Augmentation on Intensity.

Difference kidneys has large variance on intensity. So I used a heavy intensity augmentation. The probability of RandomBrightnessContrast augment is 1.0, and the probability of RandomGamma augment is 0.8.

3.8 Details on Pseudo Labels

My pseudo label process contains 3 steps:

1) Inference on the sparse kidney, save the mask in float format. Don't use any threshold to binarize them now!
2) Calculated the positive pixels. It's a simple np.sum operation.
3) Search the threshold, get TP, FP and FN, and find the threshold that meets the hosts description (85% for kidney 3 sparse) most. Suppose the model can find all the target perfectly, all the FP on sparse label can be treated as pseudo labels

The reason that I start with kidney 3 instead of kidney 2, is that half of kidney 3 has dense annotation. According to the host's paper, if a model is trained and test on the same kidney, the dice is super high. So I think the models can predict very well on kidney 3.

When making the pseudo label on kidney 2, I can't find a appropriate threshold that perfectly meets 65% sparsity. Hence, I trained difference models on 0.1, 0.15, 0.2 for better diversity in ensemble, respectively. But according to my final few submissions, I think the pseudo threshold doesn't matter too much.