SYNOPSIS OF

SWIN FUSION UNETR FOR PULMONARY ARTERY SEGMENTATION

A Project Report

submitted by

AWIK DHAR

for the thesis final review for the degree of

MASTER OF TECHNOLOGY

Data Science

Under the guidance of

Dr Ganapathy Krishnamurthi

Associate Professor

Department of Engineering Design

INDIAN INSTITUTE OF TECHNOLOGY MADRAS.

MAY 2023

Contents

1 Abstract

Semantic segmentation of MRI and CT scans using deep learning techniques has been of growing interest and currently vision transformers deliver the state of the art performance. Swin UNETR is a transformer encoder, CNN decoder U-shaped net and is the current SOTA in 3D medical segmentation tasks. 3D vision transformers tasks usually need to work with patches of small sized input for computational reasons. The U shaped network has skip connections to preserve spatial and semantic information at various resolutions. However, classical direct one-to-one skip connections are used in most architectures. We propose Swin Fusion UNETR that uses "fused" skip connections for more multi scale incorporation of spatial information at each stage of the U-shaped network, implementing configurations ranging from simplest and efficient to parameter or GPU memory heavy, that can be scaled further. We propose Swin Fusion Overdrive that delivers the best performance at the lowest parameter count. We also experiment with MAE based pre-training, which can lead to further improved performance.

2 Introduction and Related Work

The ability to isolate pulmonary arteries(PA) is of clinical significance for an array of applications. These include investigation for pulmonary hypertension where the PA are enlarged, chronic obstructive pulmonary disease(COPD), pulmonary embolism, pulmonary vascular disease and so forth. The isolation of PA is therefore of interest and segmentation is a way to achieve the same. Accurate segmentation is necessary to combat challenges like separation of arteries, veins, airways; fine tracking of artery bifurcations, inter-patient variability and imaging noise.

Although the task is of pulmonary artery segmentation, the project is developed with the lens of general 3D segmentation and the architectures and techniques that improve performance. The prevalent approach recently has been deep learning methods in the fields of end-to-end CNN architectures and vision transformers. The choice of models for ensemble and training choices will be governed by the nature of our task.

A directly relevant technique is DeepVesselNet which reduces computational and memory complexity by efficient use of "fast cross hair filters", extreme class balancing for vessels that account small fraction of vessel voxels, and pretraining on synthetic generated data with proxy tasks like vessel bifurcation detection and so forth.

Some general and efficient approaches to medical segmentation include UNETR, 3D Unet and nnUnet. 3D Unet is a 3D analogue of the standard 2D Unet, and can handle larger input volumes than the transformer counterparts. UNEt TRansformer(UNETR) employs a transformer encoder that deals with flattened patches of volume as tokens, and a convolutional decoder, yielding excellent results. nnUnet(no new net) is a task robust ensemble learning method that combines a 2D Unet, 3D UNet and cascade UNet- leveraging differing strengths of the component networks. 2D UNet is able to handle larger input sizes and is more robust to anisotropic data(prostate cancer), but doesn't leverage third spatial dimension. 3D UNet has access to all spatial dimensions but can only take smaller slices of input due to GPU memory limits. UNet cascade involves a 3D UNet which is trained on downsampled images and another 3D UNet that further segments the upsampled segmentaions from the first stage. Such an ensemble is able to capture different aspects of data through unique architectural strengths.

For U-shaped networks, use of multi scale information and other treatments of skip connections has been of interest. SA-UNet has a spatial attention module that uses MaxAvgPooling and sigmoid activation to calculate attention weights in its bottleneck-layer to get more efficient bottleneck features. MISSFormer is a fully transformer based U-shaped network that presents a computationally efficient attention method and custom MLP heads in the transformer blocks. More importantly, it introduces a "context bridge" that combines feature maps from all resolutions, by flattening and concatenating, instead of just direct one to one connections like in most other U-shaped architectures. The paper also discusses ablation studies on transformer block depths that can be of interest for the PARSE data as well, since there are only 90 CT scans in the training data. Interestingly, in the ablation study, the authors study the impact of successive removal of skip connections from each stage top to bottom. The successive stages from top to bottom were yielding progressively half the tokens for the context bridge input, since the resolution was getting quartered at each stage while channels doubled, leading to half the map overall. Removing these large skip connections from higher(larger feature map) stages had a more detrimental impact on the segmentation performance, since their ablation removes a lot of feature map "pixels" and therefore, a lot of information that could have been passed to the decoders.

UNETR, Swin Unet and Swin UNETR are transformer-encoder CNN-decoder U-shaped networks with skip connections. Swin Unet is primarily is used for 2D image segmentation and implements computationally efficient self-attention mechanism, with shifted windows as shown in Fig 1. UNETR uses 3D input with encoder blocks that output maps at same resolution, and skip connections that use res-blocks with maxpooling to downsample these outputs to decoder. Swin UNETR adapts the Swin Transformer blocks for 3D feature map encoders, with the same CNN decoders and skip connection res-blocks as UNETR, and achieves state of the art results in 3D segmentation tasks. The computationally efficient attention is especially important for 3D inputs.

UNet 3+ uses adaptive maxpooling to upscale/downscale feature maps from all stages to the current stage resolution and combine them for multi scale information at each stage. It leads to significant improvements in Dice scores(2-3%) at reduced number of parameters compared to UNet. UNet 3+ is not adapted to 3D segmentation since the computational complexity would be infeasible. MISSFormer flattens out the maps and doing so on 3D maps would result in a far too long sequence. UNet 3+ on the other hand uses all stages including upscaling of lower stages as well which we hypothesize

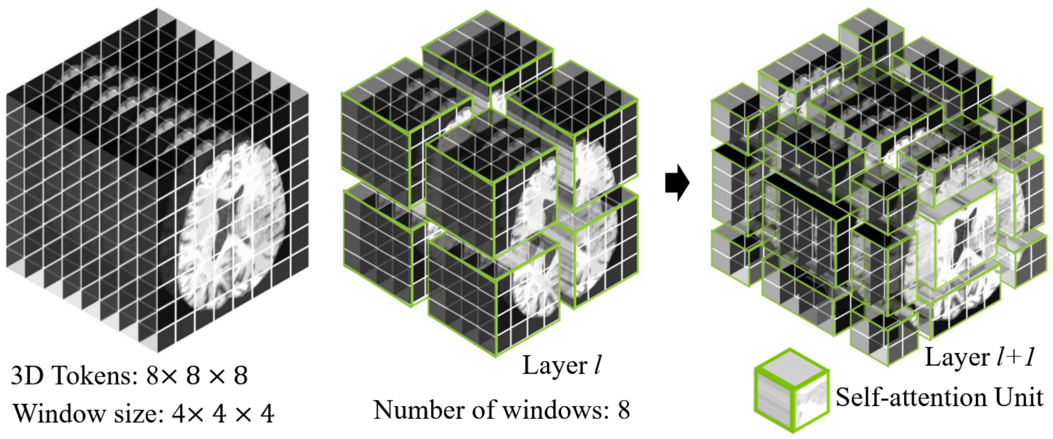

Figure 1: Shifted window mechanism in Swin UNETR

doesn't add significant information to justify the parameter increase including it would cause. It also uses a Classification Guided Module(CGM) to add an additional classification task to combat over-segmentation due to noisy information accumulated at the shallower stages. Adapting from UNet 3+ and MISSFormer for 3D segmentation, we introduce Swin Fusion UNETR in its various configurations, delivering compute and memory efficient yet superior segmentation performance compared to vanilla Swin UNETR.

3 Formulation and Objective

The PARSE data available with us consists of 100 3D volumes, split into 90 for training and 10 for test. To perform effective segmentation of pulmonary arteries, the following objectives were laid down:

1. Training the vanilla Swin UNETR from scratch to get baseline segmentation Dice score.
2. Testing MAE based reconstruction masking settings
3. Evaluation of different skip connection enhancement techniques and picking and efficient one for 3D inputs.
4. Testing various extents and configurations of skip connection fusion to balance complexity vs performance.

4 Methods

4.1 MAE based Pre-training

The actual method to be executed involves using randomly masking 75% of the patches in a 2D image and feeding only the non-masked tokens with added positional embedding, to a large transformer encoder with light transformer decoder. This allows for a heavy encoder to be trained on smaller inputs and saves compute and memory. But such a routine is more involved as the volume split into patches happens internally within the model and original patch position encoding is to be preserved. Using blacked out tokens deteriorates the performance relative to non-masked only training, but outperforms training from scratch. Reconstruction was the proxy task experimented with by trying different levels of masking the input. The settings were 10%, 30%, 50% masking. At 60% the reconstruction was too distorted so the setting was discarded.

4.2 Swin UNETR

To get a reference baseline, Swin UNETR was trained from scratch with input size 96×96×96. The actual image size is 170×170×170 from which the 96 sized patches are sampled randomly during training. Feature size and other model arguments are kept at default for all experiments.

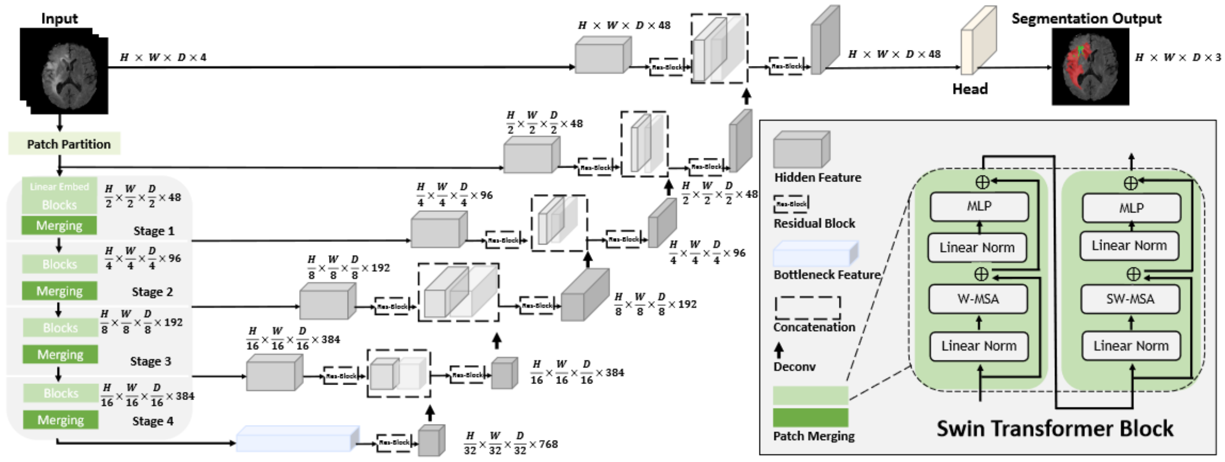

Figure 2: Swin UNETR with one-to-one skip connections

4.3 Swin Fusion UNETR

Swin UNETR uses direct one-to-one skip connections between the encoder and decoder blocks. Especially for sparse/fine segmentation tasks and small input sizes, the fine details within the feature maps get lost along the stages of the enocoder as the maps get smaller in resolution. To combat this, MISSFormer uses "context bridge" which fuses multi-scale encoder information/feature maps and splits and passes enriched skip connections to the decoder blocks. For 3D segmentation however this is computationally inefficient as the flattened input to the context bridge transformer would be too big. UNet3+ for each stage downsamples the maps from above stages and upscales maps from below, concatenates and passes this skip connection to the corresponding decoder block after convolution.

Using both downsampling and upsampling feature maps for each stage may yield better results, but the upsampled skip connections may not provide significant information benefit like the higher downsampled maps do. They would also be much more expensive due to large number of channels in the lower stages. Considering this marginal

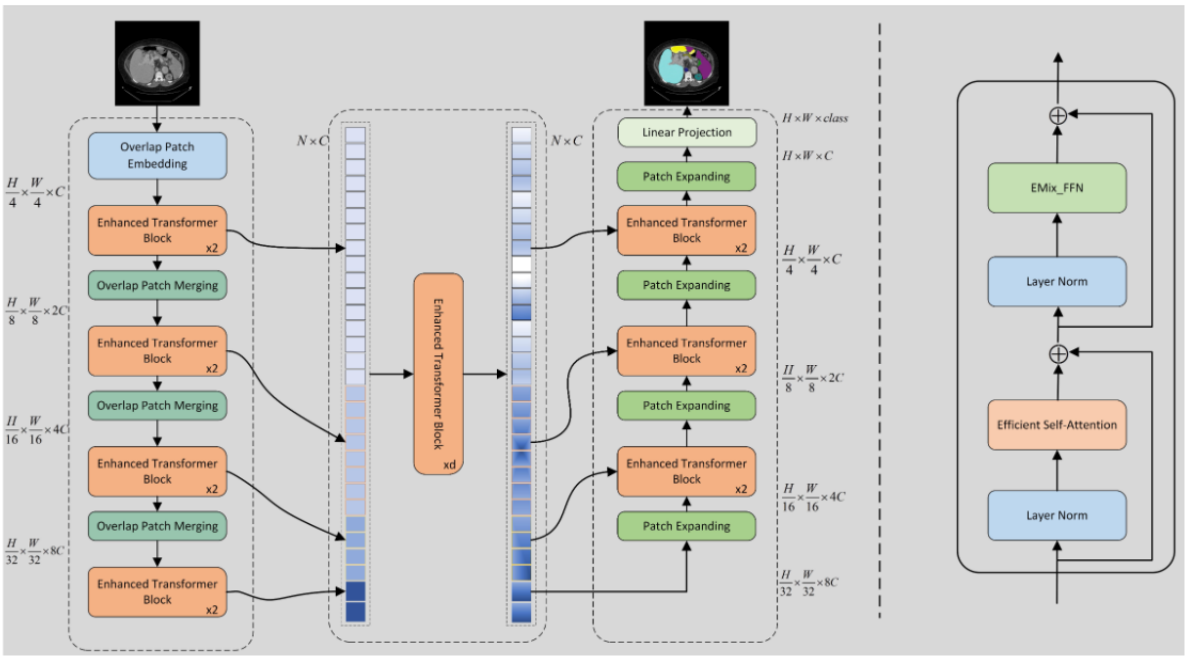

Figure 3: MISSFormer with context bridge. Skip connections are flattened, concatenated, passed through transformer blocks and split into skip connections again.

performance uplift vs parameter count trade-off, only downsampling of higher feature maps has been done, with good results. The higher maps have been "fused" with the maps at every stage, through concatenation along the channel axis, after which a residual block is used to produce the final skip connection.

Experiments have been done on the various configurations that can arise from this. Number of skip connections/maps above the current stage, type of pooling to bring them to the same resolution before concatenation(max, avg, maxavg), type of convolution after fusion of connections.

4.3.1 Pooling

Maxpooling and MaxAvgPooling were tried for the downsampling operations. MaxPooling seemed to plateau at Swin UNETR's performance, and was discarded. Whether or not the feature maps were obtained directly from the residual blocks outputs in the higher stages(see Swin UNETR Fig. 2), which were the previous skip connections to the decoder, or from the Swin transformer encoder's hidden features/outputs after convolution, two variants were obtained. The "v1" models directly used maps from above that are also fed to the corresponding decoders, while "v2" models didn't use those but

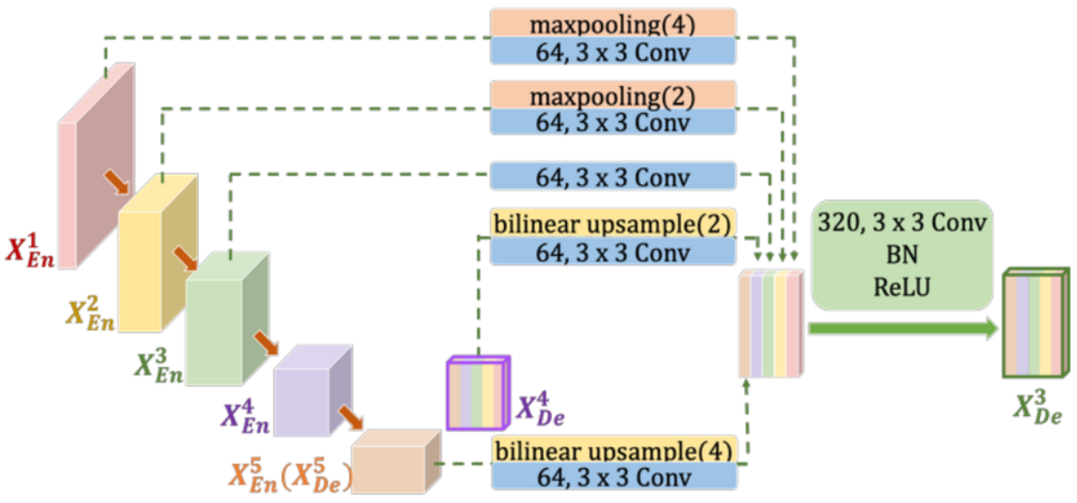

Figure 4: UNet3+ with full-scale skip connections. Swin fusion only fuses the skip connections above, downsampled to current stage's resolution, the same way UNet3+ does as shown here. The final convolution step may be done using filters of size 3 or 1 as discussed in 3.6.

instead used convolved hidden features/outputs of the transformer encoder.

4.3.2 Number of stages fused

The extent of fusion of maps was also studied. Three variants were tried; one fusing(pooling and concatenating) one skip connection directly above to the given stage, one fusing two skip connections above, and one where all the skip connections above are being fused.

4.3.3 Convolution kernel size

The number of parameters that the Conv3D layers introduce scales cubically with the kernel size. Especially when fusing all stages above, the number of convolution filter parameters to be learned increases dramatically, and isn't worth it since they're not being used within the encoders or decoders themselves but merely the skip connections. Kernel sizes implemented are 1 and 3, with wide gap in the number of parameters (for example 45 mil. vs 89 mil. in Swin fusion overdrive) but no observed difference in performance(Swin_fusion_v2). Larger number of parameters makes tasks prone to overfitting and also increases the training time due to slower convergence. Hence, given similar performance, lower parameter count is ideal.

4.4 Model versions

1. **Swin_fusion_1_v1**: 1 skip connection(res-block output) fused
2. **Swin_fusion_2_v1**: 2 skip connections(res-block output) fused
3. **Swin_fusion_2_v2**: 2 skip connections(encoder output passed through separate res-block) fused. 1 skip connection fused for stage 2.
4. **Swin_fusion_od**: All skip connections above a given stage(encoder output passed through separate res-block) fused. "od" here refers to overdrive.
5. **Swin_fusion_od(large)**: Same as overdrive, but boosting channels in maps before pooling and fusion. Increases performance at the cost of VRAM.

For **Swin_fusion_2_v2**, MaxAvgPool was also tried as the pooling operation before concatenation/fusion. The res-block mentioned is the one found in the vanilla Swin UNETR as seen in fig. 1, used as is in all these models. The other res-blocks mentioned in v2 models(3,4) are the ones we introduce to transform the skip connections and make them more suitable to the stage below they are going to be fused with.

5 Experiments and Results

5.1 Implementation Details

The training was done on Google Colab with 12.7GB RAM and Tesla T4 GPU(15GB VRAM). This posed some constraints on batch size and number of iterations each model was allowed. Batch size was 1 for the dataloader but effectively 2 due to 2 random samples per CT scan for each iteration. Number of iterations ranged from 20,000 to 50,000 depending on whenever it reached the plateau. Plateau is defined here as less than 0.4% of improvement in two successive 5000 iterations, and it's point of occurrence differed among the models.

Data augmentation has been done extensively due to the lack of data. Random patch sampling does help expose the model to get more different views of the data. Random flipping along the three axis and random shifts in intensity with a probability of 0.5, and rotation with a probability of 0.1 have been done. Voxel spacing has been kept uniform along all axes unlike previous efforts, to not distort the input and keep it uniform.

Note: Check the last page for optimized implementation details.

5.2 Loss functions

For reconstruction pre-training, **SSIM**(Structural Similarity Index Measure) was chosen as it would provide more appropriate error rates to make the model learn. MSE on the other hand, while classically used, is notorious for giving large errors for slight shifts in pixel positions, which is a misleading learning signal to the model. MSE's limitations even led to the ESRGAN paper for more effective super-resolution over SRGAN using perceptual loss. The test metric is SSIM metric.

$$SSIM(x,y)=\frac{2\mu_x\mu_y}{\mu_x^2+\mu_y^2}\frac{2\sigma_{xy}}{\sigma_x^2+\sigma_y^2}$$

Figure 5: SSIM loss function

where *x* and *y* are windows being compared, of common size. μ refers to the window mean and σ refers to the standard deviation within the window. $\sigma_{x,y}$ is the covariance in x and y.

DiceCELoss was used to combine Dice loss and Cross Entropy loss for accurate segmentation results. The test metric is Dice Metric.

$$\mathcal{L}(G,Y)=1-\frac{2}{J}\sum_{j=1}^{J}\frac{\sum_{i=1}^{I}G_{i,j}Y_{i,j}}{\sum_{i=1}^{I}G_{i,j}^2+\sum_{i=1}^{I}Y_{i,j}^2}-\frac{1}{I}\sum_{i=1}^{I}\sum_{j=1}^{J}G_{i,j}\log Y_{i,j}.$$

Figure 6: Dice Loss function

where I denotes voxels numbers; J is classes number; Y$_{i,j}$ and G$_{i,j}$ denote the probability of output and one-hot encoded ground truth for class *j* at voxel *i*, respectively

5.3 Reconstruction Pre-training

When 10% of the volume was blocked, the SSIM score obtained was 93.25%. With 50% of volume patched, the SSIM score obtained was 83.26%

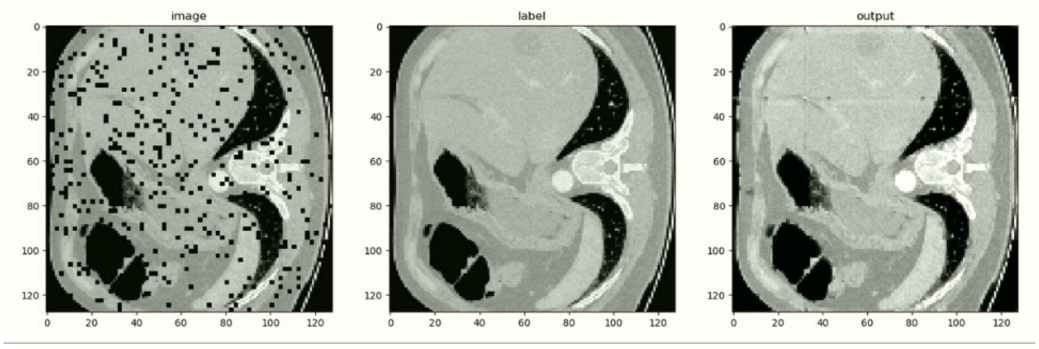

Figure 7: Reconstruction with 10% masking

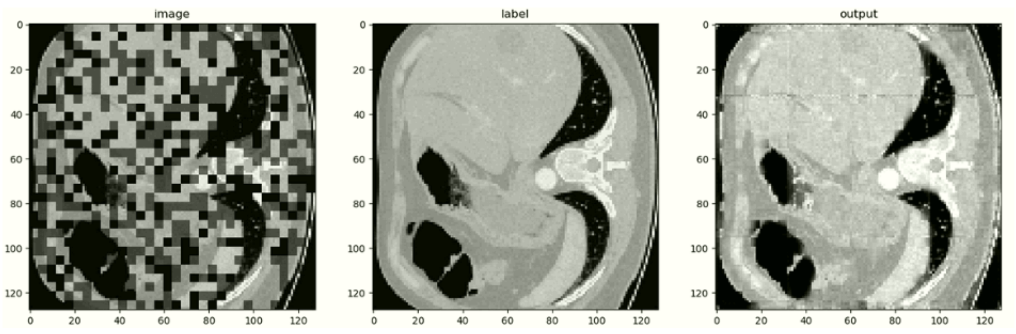

Figure 8: Reconstruction with 50% masking

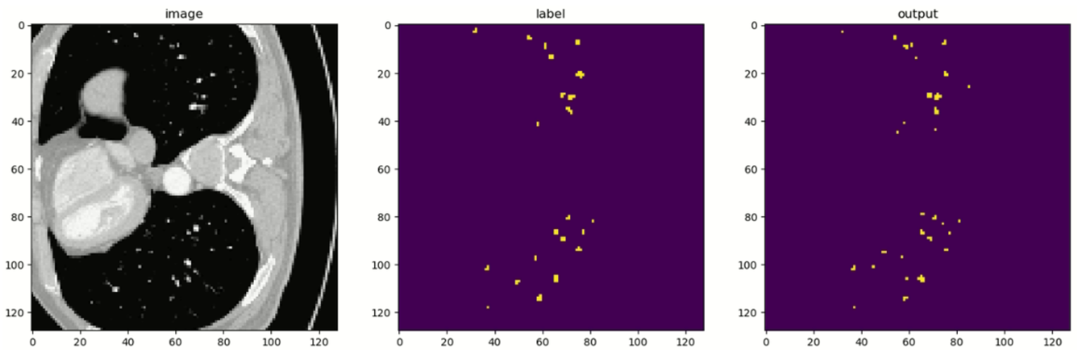

Figure 9: Segmentation with pre-training(50% mask)

The performance of the reconstruction task seems heavily dependent on the availability of low level details. Unlike tumor segmentation task where relatively large blobs of tumor are to be located, PA vessels are fine and sparse. For 10% patching, SSIM score went up from 81 to 93.25 by merely not re-scaling the volumes to 96×96×96 and using sliding window of that size.

Method	SSIM(%)↑	DSC(%)↑
Scratch	-	81.55
10% mask	93.25	81.56
50%mask	83.26	81.85

Table 1: Comparison of results for pre-training

The segmentation Dice score saw no increase over scratch for 10% mask due to weak stimulus, and a 0.3% increase when using 50% mask. Using 75% mask gets maximal improvements in MAE due to a strong stimulus to the model to understand structure of the image to be able to generate a plausible reconstruction, and the same is expected here. Additional lung CT data from another task is being made compatible(scaled) with this data, which combined with the larger volume size at the same 96×96×96 input size(same computational cost), and 75% masking will lead to pre-trained weights yielding significant improvements in Dice score.

In our experiments, we did not pre-train our highest performing model - Swin Fusion UNETR, due to already too many training cycles for the Swin fusion models, since it would take two more, one for getting pre-trained weights and one for using these weights. It would also help to get more lung CT scans from other tasks(tumour segmentation) and get even stronger pre-training weights, which can be part of future work.

5.4 Swin fusion UNETR

All models apart from **Swin_fusion_od** have skip connection fusion until the second last(5th) stage, with the last stage(6th) untouched. Since the number of parameters introduced by convolution filters scales quadratically with the number of channels(assuming equal in and out channels like in our case), the last stage would blow up the parameter count if the number of channels were further increased by our skip connection fusion. The number of channels doubles as along the stages until the bottleneck layer(6th stage) where using a kernel size of 3 is expensive. Therefore, we replaced the vanilla Swin UNETR's 3×3×3 filters with 1×1×1 filters dramatically reducing total parameter count from 62 million to 45 million, despite the increase in number of channels.

Swin_fusion_1_v1 and **Swin_fusion_2_v1** are both "v1" models where the res-block outputs in Swin UNETR(Fig.2) are directly maxpooled and passed to the below stage, unlike "v2" models which use encoder outputs for the same. The prior "1" and "2" in their names indicate the number of skip connections above being fused with current stage, at each stage. This leads to extremely lightweight method of fusing skip connections, but for "v1" models saturates quickly in performance(82.49% and 82.59% respectively) as during training the model may get confused as to whether this skip connection, being shared by multiple stages, is to learn maps for the given stage or the stage it is being passed to. The maps farther away may also be less relevant as they contain more differing information. The overall uplift using these models is 1% with comparable number of parameters(62 vs 62-67 million).

Swin_fusion_2_v2 solves this by having separate dedicated res-blocks that transform the feature maps directly from transformers encoder(2 stages from above) before being passed down. Both kernel sizes of 3 and 1 were tried for the final res-block before passing the skip connection to the decoder block, but using kernel size 3 didn't see any significant improvement over using kernel size 1 instead(83.19% vs 83.29%). This suggests that the bottleneck in performance improvement through skip connections optimization is primarily in availability of the information from different semantic levels/resolution as opposed to getting further features out of the connections. The uplift obtained is higher than the v1 models at **1.7**%.

Fusion of multi-scale information incorporates semantic and spatial information that can get lost by the bottleneck stage. This is especially important in our case as the artery segmentations are fine and sparse, and access to information from larger resolution feature maps can preserve the knowledge of presence and precise location of these arteries, more so when the input size is small to begin with. Hence, **Swin_fusion_od**(od meaning overdrive) was developed where *all* stages above the given stage are fused. It achieves the highest increase in segmentation performance, **improving 1.8% over baseline**, from 81.55% to 83.39%, at 45 million parameters, and slightly reduced RAM usage. VRAM consumption optimization can be very helpful especially for 3D inputs where feature maps take up lot of memory and batch size is extremely low. Swin_fusion_od(large) is the same same as the regular "od" model except some of the higher stages feature maps(especially the first stage) have increased channel size before maxpooling, so that they provide more information when fused below. The result is a significant **2.67% improvement over baseline at 84.21%**, although this is the only model that uses significantly more VRAM due to more channels and larger maps stored in memory.

Using MaxAvgPool seemed to plateau to the baseline performance of 81.5%, but it might have just needed steady and continued training due to possibly slower but consistent convergence. MaxAvgPool performs both Max and Avg pool operations and concatenates the results along the channel axis.

Model	**#Parameters(mil.)↓**	**DSC(%)↑**
Swin UNETR	62.3	81.55
Swin_fusion_1_v1	62.7	82.49
Swin_fusion_2_v1	63	82.59
Swin_fusion_2_v2	66	83.19
Swin_fusion_2_v2*	80	83.29
Swin_fusion_od	**45.4**	*83.39*
Swin_fusion_od(large)	*47.4*	**84.21**

Table 2: Comparison of results. Best results in bold. Second best in italics. Swin fusion UNETR (overdrive version) performs significantly better than vanilla Swin UNETR at reduced parameter count. * - Using kernel size 3 for final res-blocks

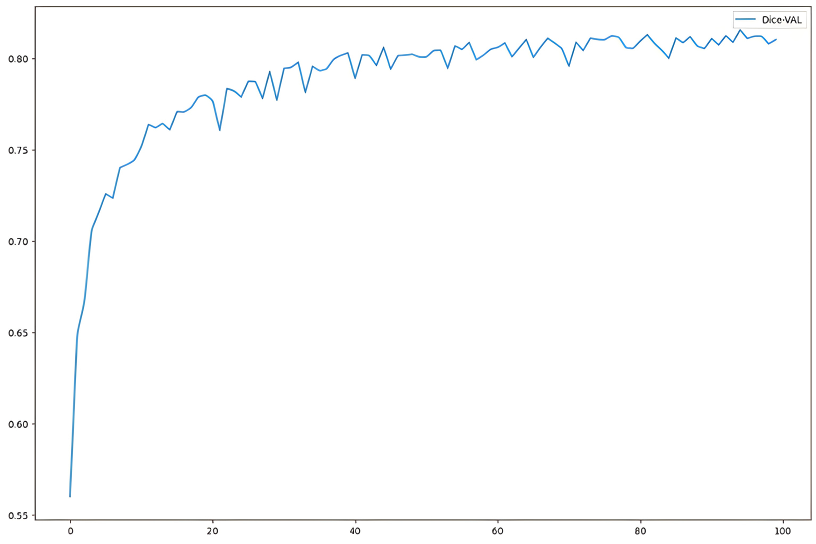

Figure 10: Swin UNETR training from scratch

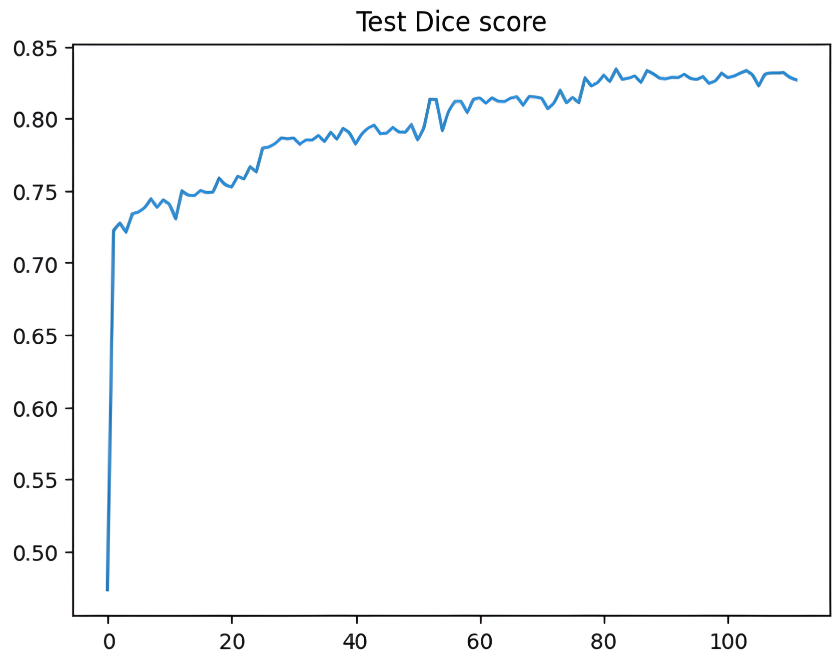

Figure 11: Swin Fusion UENETR (overdrive) converges more gradually and steadily, due to higher skip connection complexity over Swin UNETR

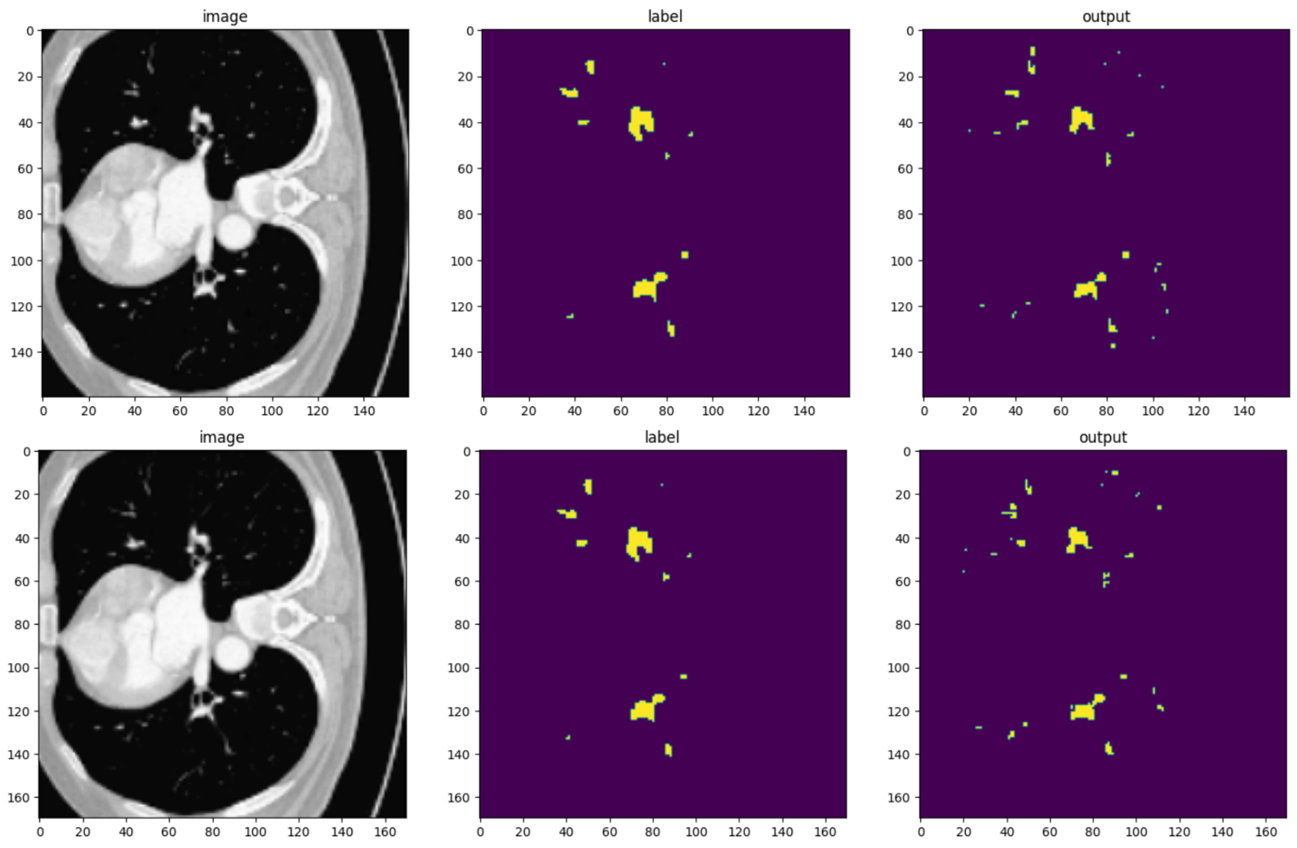

Figure 12: Qualitative comparison of Swin UNETR vs Swin Fusion Overdrive. Swin Fusion Overdrive over-segments slightly less.

6 Future Scope

In our experiments the number of channels in the stages being fused were unaltered. This can make the stimulus from the higher maps weaker, for example stage 2 has 48 channels whereas stage 6 has 768. Meanwhile, stage 1 has only 1 channel. When reduced to the same spatial resolutions with such low number of channels, the information gained by such a skip connection from above is progressively negligible as the gap between stages being fused increases. In UNet3+, all stages are reset to 64 channels before concatenation, which makes all stages have similar weightage. Swin Fusion Overdrive(large) moderately increases the number of channels in the maps from the higher stages, which significantly improves performance, but channel distribution is still not uniform to the extent of UNet3+. This was done so as to conserve GPU memory, and memory efficient skip connection fusion can be an area of further investigation.

Incorporation of lower maps by upscaling like UNet3+ could also be experimented with although we hypothesize the gains in performance might be lower and not worth the increase in parameter count.

7 Conclusions

In this thesis, we propose Swin Fusion UNETR, built on top of Swin UNETR network, for pulmonary artery segmentation although the network's improvements would generalize to other tasks as well. PARSE dataset was used to validate the effectiveness of the method. The fusion of multi-scale information in skip connections is a generalizable idea that can be applied to other networks as well to obtain compute efficient improvements in segmentation performance, and this thesis proposes a simpler and effective way of implementing the same. Combined with MAE based reconstruction pre-training, data efficient segmentation performance can be attained.

REFERENCES

[1] Hatamizadeh, Ali, et al(2022), Swin UNETR: Swin Transformers for Semantic Segmentation of Brain Tumors in MRI Images. http://arxiv.org/abs/2201.01266

[2] H. Huang, L. Lin, R. Tong, et al(2020), UNET 3+: A full-scale connected U-Net for medical image segmentation http://arxiv.org/abs/2201.01266

[3] Tang, Yucheng, et al.(2021) Self-Supervised Pre-Training of Swin Transformers for 3D Medical Image Analysis.https://doi.org/10.48550/ARXIV.2111.14791.

[4] F. Isensee, P. F. Jaeger, S. AA Kohl, J. Petersen, and K. H. Maier-Hein nnu-net: a self-configuring method for deep learning-based biomedical image segmentation *Journal title.*, 18(2): 203–211(2021)

[5] X. Huang, Z. Deng, D. Li, X. Yuan(2021), MISSFormer: An Effective Medical Image Segmentation Transformer https://arxiv.org/pdf/2109.07162v1.pdf

[6] Y. Xie, J. Zhang, C. Shen, Y. Xia(2021), CoTr: Efficiently Bridging CNN and Transformer for 3D Medical Image Segmentation https://arxiv.org/pdf/2103.03024.pdf

[7] J. Chen, Y. Lu, Q. Yu, X. Luo, et al.(2021), TransUNet: Transformers Make Strong Encoders for Medical Image Segmentation https://arxiv.org/pdf/2102.04306v1.pdf

[8] G. Tetteh, V. Efremov, N. D. Forkert, M. Schneider et al.(2020) DeepVesselNet: Vessel Segmentation, Centerline Prediction, and Bifurcation in 3-D Angio https://arxiv.org/ftp/arxiv/papers/1803/1803.09340.pdf

[9] O. Cicek, A. Abdulkadir et al.(2016), 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation https://arxiv.org/pdf/1606.06650.pdf

[10] K. He, X. Chen, S. Xie, Y. Li, et al.(2021) Masked Autoencoders Are Scalable Vision Learners https://arxiv.org/pdf/2111.06377.pdf

[11] R. Bachmann, D. Mizrahi, A. Zamir(2022), MultiMAE: Multi-modal Multi-task Masked Autoencoders https://arxiv.org/pdf/2204.01678.pdf

Resources

The following configuration was used to double the training speed from 1.87s/it to 1.10 it/s. We also lay suggestions to fit the training routine in memory with reasonable batch size.

1. Use ThreadDataLoader instead of DataLoader with num_wokrers set to 0 to use multithreading instead of multiprocessing.
2. If possible, disable tracking of meta data in the meta tensors by executing

   ```
   set_track_meta(False)
   ```

   prior to starting training. For us it wasn't possible unless torch compiled model was used.
3. If VRAM is available disable gradient checkpointing by setting argument

   ```
   use_checkpoint=False
   ```

   during model initialization. This may provide 20% increase in training speed(it/s)
4. Ensure that CacheDataset is used, at max cache_rate you RAM allows(may get constrained on Colab) and set the copy_cache argument to False. The 1.87s/it figure only had this feature used among this list, however we only set the copy_cache to False later on which provided some speed improvement.
5. Obtain an optimized model by executing(requires Pytorch 2.0+)

   ```
   model = torch.compile(model)
   ```

   prior to training, although this might consume some extra and growing memory(RAM). Attempt to use

   ```
   model = torch.compile(model, mode = 'max-autotune')
   ```

   although it did not work for us. It is supposed to yield the most optimised model but may make training initiation even longer than the default method.

Use mixed precision training to halve the VRAM usage but ensure that the input to the model isn't so small in magnitude that the reduced precision deteriorates performance. Either scale the pixel values or appropriately or just normalize them to between 0 to 1 like we did.

For a comprehensive tutorial on optimal training using MONAI library, refer to the official MONAI guide on fast training and this notebook for demo, or google Fast Model Training Guide monai if the link has expired at the time of reading.