A Transformer-based Approach for Semantic Segmentation

Xu Han

March 5, 2023

Abstract

In this essay I summarize my exploration for a new approach to do semantic segmentation on 3D medical images that is based purely on a Transformer architecture without relying on convolutions nor U-Net-style networks. The attempt turns out to be promising and achieves comparable results to the state-of-the-art Swin UNETR [HNT+22] on BraTS 2021 dataset.

1 Motivation

At the time of writing, segmentation networks for medical images usually utilise a U-shape network with skip connections proposed by U-Net [RFB15] (Figure 1) to combine features from different scales by down-sampling and up-sampling using different convolution kernels. However, inspired by the recently popular Vision Transformer [DBK+20], we want to find a new approach to perform medical image segmentation which

- does not use a U-shape structure but a simpler and more straightforward one, and
- does not rely[1] on convolutions.

2 Rationale

The effectiveness of the U-Net may due to its effective answers to the following questions

1. Q: How to encode features from different scales?

   A: By repeatedly applying convolutions to capture features from larger and larger scales.

2. Q: How to mix features from different scales?

   A: By skip connections to merge coarser features to finer ones.

Therefore, to replace the U-Net structure without suffering its effectiveness, we must provide another set of answers to these questions bearing the spirit of Transformer. My answers are:

1. Q: How to encode features from different scales?

   A: By tokenise/patchify the input using different patch sizes.

2. Q: How to mix features from different scales?

   A: By calculating attentions on the sequence mixed from patches of different sizes.

3 Network

In this section I provide an overview of the whole network structure without elaborating much details, for which please refer to the code. For the sake of convenience, in the article we call the proposed network **SegV3**, which simply results from the fact that it is the third structure I tried.

Without loss of generality, although the mathematics works for arbitrary dimensionality and heterogeneous dimensions, we assume the input image is a d-dimensional hypercubic with size D and all patches are hypercubics.

[1] I ended up using convolutions, but not in the core step. Therefore, I used word *rely* rather then *use*.

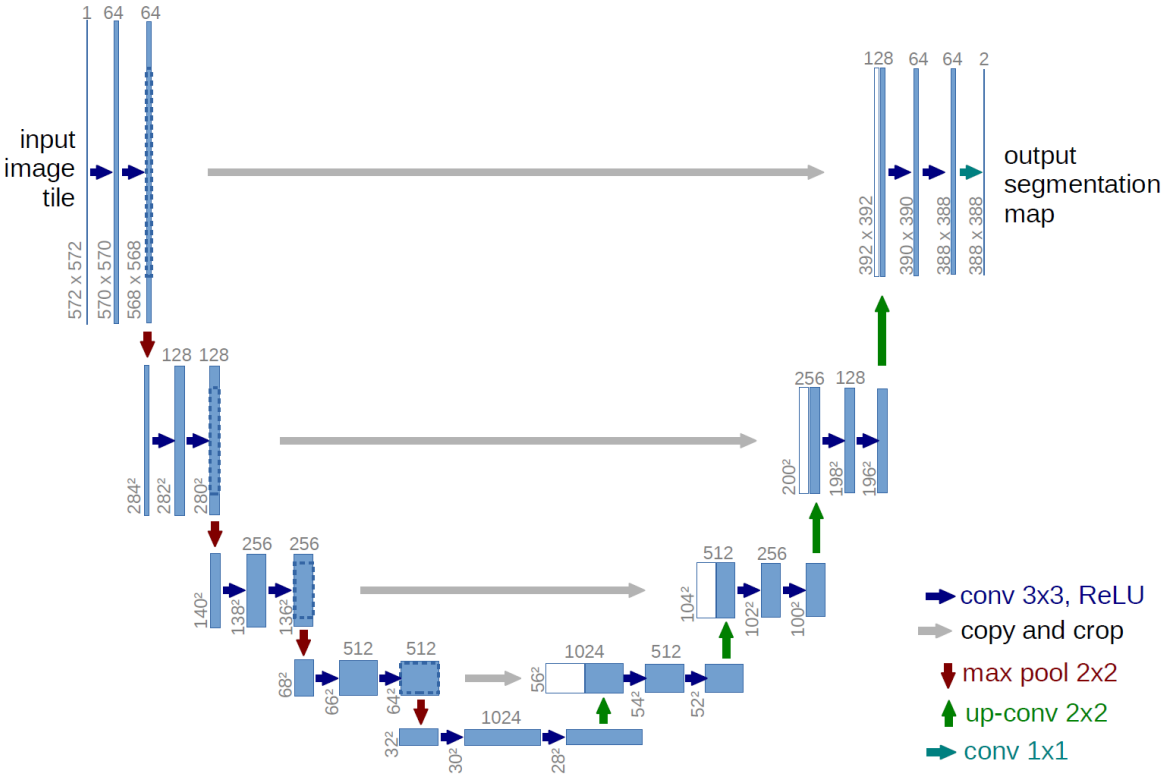

Figure 1: U-Net architecture.

3.1 Encoder

The encoder part of the network is shown in Figure 2, which consists of the following steps:

1. Let N be the number of scales we want to capture.
2. Patchify and encode[2] the input using different patch sizes p_i to get N sequences of length $n_i := (D/p_i)^d$ for $i = 1, 2, \cdots, N$.
3. Add to each sequence a **scale embedding**[3] e_i for $i = 1, 2, \cdots, N$.
4. Concatenate all N sequences into a single sequence of length $\sum_{i=1}^{N} n_i$ and encode it[4].
5. Split the concatenated sequence back into N shorter sequences of length n_i for $i = 1, 2, \cdots, N$ respecting the original partition.

3.2 Decoder

After encoding, we get N sequences of length n_i for $i = 1, 2, \cdots, N$, each of which is a tokenised representation of the original image from the scale defined by patch size p_i after self-attention within that scale and mixed-attention across all different scales. Now, we want to find for each i a decode map between:

- input: a tensor of shape[5] (n_i, C_e).
- output: a tensor of shape[6] $(\mathbf{D}, C_o)$.

[2] Follow the same encoding method of ViT.

[3] Use the same positional embedding formula from ViT with spatial dimension being one-dimensional (N). It turns out to be very crucial for the performance.

[4] Use the sample encoding method of ViT, as in the previous step.

[5] We skip the batch dimension which always lies at first slot.

[6] By bold font we mean a vector of d components.

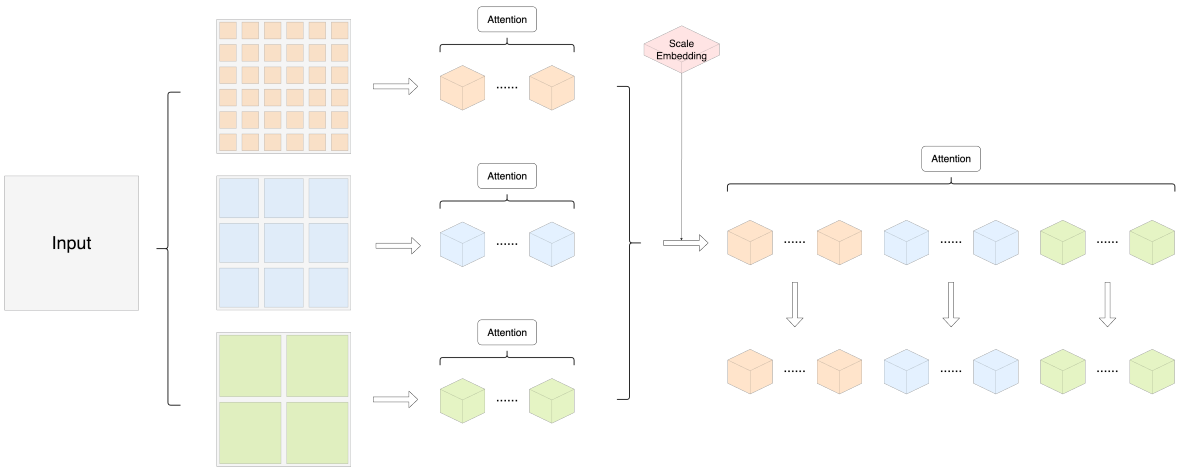

Figure 2: Encoder part of SegV3.

in which C_e is the number of hidden channels for encoders, and C_o the number of output channels. A reasonable approach is to achieve the mapping in two steps[7]:

$$(n_i, C_e) \xrightarrow{f} \left(n_i, p_i^d \cdot C_o\right) \xrightarrow{rearrange} (\mathbf{D}, C_o)$$

f can be a simple MLP from dimensions C_e to $p_i^d \cdot C_o$, but in that case the number of parameters will be huge, and the model may overfit. To rescue, the decoding is performed in three steps for each of the N encoded sequences:

1. Linearly map the sequence (n_i, C_e) to a sequence (n_i, C_d) where C_d is the number of decoder channels.
2. Rearrange (n_i, C_d) to $(n_i, \mathbf{q}, C_h)$ where $C_d = |\mathbf{q}| \cdot C_h$ and C_h is the number of hidden channels. Namely, we view the C_d channels as an aggregation of feature vectors of dimension C_h with spatial dimensions $\mathbf{q}$.
3. Upsample[8] $(n_i, \mathbf{q}, C_h)$ to $(n_i, \mathbf{p}_i, C_o)$.
4. Rearrange $(n_i, \mathbf{p}_i, C_o)$ to $(\mathbf{D}, C_o)$.

3.3 Merging

After decoding, we now have N feature maps. To get the final feature map from them, we apply the *Scale-Space Rendering* method from [YHS+21] (Figure 3).

3.4 Window

Assuming $D = 144$, and $p_0 = 12$, i.e. the input is an $144 \times 144 \times 144$ image and we partition it into patches of size 12, then there will be $(144/12)^3 = 1728$ patches, which is my empirical limit to be feasible for a 24GB-memory GPU. If the patch size is further decreased, the memory insufficiency will be inevitable. On the other hand, however, smaller patch sizes means finer representational units which are empirically crucial for good performance in segmentation tasks. Comparing to Swin UNETR's patch size, which is 2, 12 is too large. Therefore, we must find a way to decrease the patch size. As a result, we apply a simple and straightforward solution by introducing windows (Figure 4).

1. Set a window size W and partition the input image into $(D/W)^d$ windows.
2. Perform the aforementioned encode-decode flow independently on each window.
3. Collect feature maps from each window and union them according to their original positions.

[7] Recall $n_i = (D/p_i)^d$.

[8] It is where I have to use convolution, i.e. by re-interpreting part of the channel dimensions as spatial dimensions to transform using convolutions to significantly reduce number of parameters and the risk for overfitting.

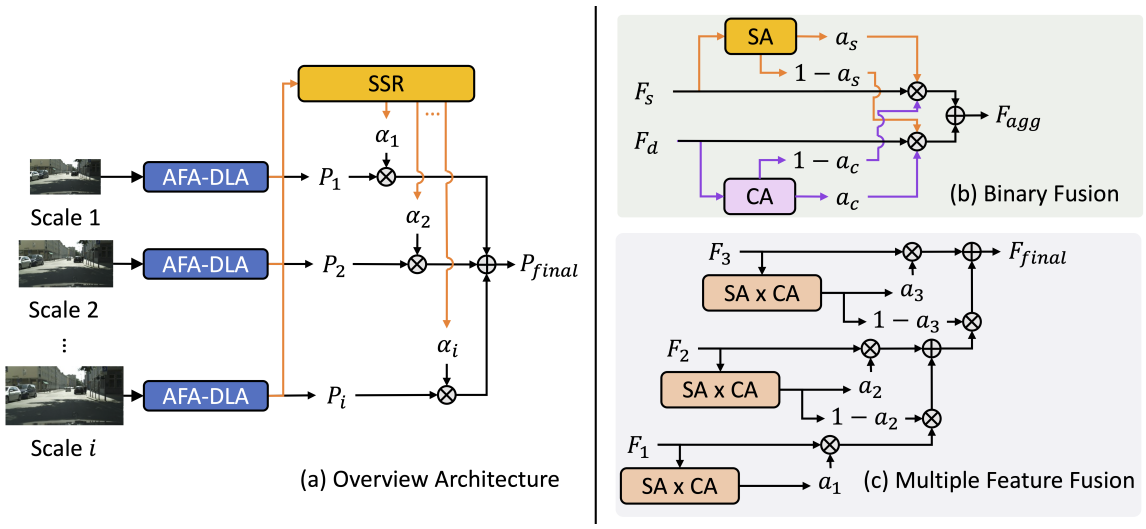

Figure 3: Scale-Space Rendering.

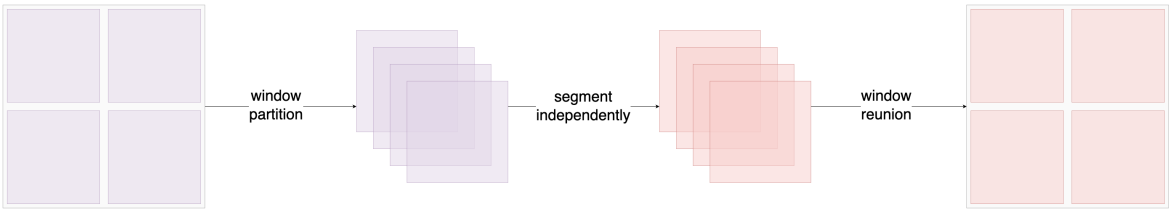

Figure 4: Introducing windows to allow smaller patchs.

4 Hyperparameters

The final base model is realised under following hyperparameters.

- Some hyperparameters are determined by the question.
  - **Dimentionality** $d = 3$ which is due to BraTS2021 is a 3D medical image dataset.
  - **Input channel** $C_i = 4$ because the BraTS2021 dataset has four input modalities: T1, T1ce, T2, and FLAIR.
  - **Output channel** $C_o = 3$ because the prediction consists of three labels: tumor core, whole tumor, and enhancing tumor.
- Some hyperparameters are set due to memory concerns.
  - **Window size** $W = 24$, a number which bears many factors including 3 which will give a reasonable $(24/3)^3 = 512$ patches.
  - **Spatial size** $D = 96$ to which all input images will be randomly cropped. There will be $(96/24)^3 = 64$ windows which is not too big.
  - **Patch sizes** $(p_0, p_1, p_2) = (3, 6, 12)$. Therefore in each window there will be three sequences of length $(24/3)^3 = 512, (24/6)^3 = 64, (24/12)^3 = 8$.
  - **Decoder channel** $C_d = 432$ which equals $3^3 \times 16$, a convenient factorisation for the final decoding step.
  - **Decoder hidden channel** $C_h = 16$ which requires a final upsampling from $(3, 3, 3, 16)$ to $(p_i, p_i, p_i, 3)$, where $p_i = 3, 6, 12$, which looks reasonable and feasible.
- Some hyperparameters are set by experiments.
  - **Encoder channel** $C_e = 432$ simply to be symmetric with the decoder.

– All other self-attention related hyperparameters are set to the same value for all encoder, decoder, and mixed sequences, e.g. number of blocks to be 2, and number of heads to be 8.

5 Results

The author of Swin UNETR partitioned the BraTS2021 dataset into 5 folds. The mean Dice score of the fold 0 is summarized in Table 1.

	Whole tumor(WT)	Tumor core(TC)	Enhancing tumor(ET)	#Params
(SwinUNETR/Paper)	(0.929)	(0.914)	(0.876)	
SwinUNETR/Xu	0.916	0.891	0.854	62.1M
SegV3	**0.922**	**0.896**	**0.863**	**35.5M**

Table 1: Evaluation result.

- SwinUNETR/Paper is the result reported in the paper [HNT+22] (in Table[9] 2) but I failed to reproduce it. Therefore I put it in a parentheses.
- SwinUNETR/Xu is the result evaluated by myself by downloading[10] the pretrained model from their official GitHub repository[11] and evaluating *using my own script*.

As we can see, we can achieve comparable, if not better, performance on all segmentation labels using nearly only half the number of parameters.

6 Future Works

If the effectiveness of a Transformer-style network is verified, I believe it is a good starting point to further explore the possibility for self-supervised learning, inspired by Masked Autoencoders [HCX+21]. In fact, I believe there is a general mechanism to turn *any* Transformer-style supervised training pipeline to a self-supervised one, as shown in Figure 5, which I call **Self-Supervisorisation**, which requires four components:

- A *Tokenizer* to turn input image into a sequence of tokens.
- A *Encoder* to encode the input token sequence.
- A *Reconstructor* to reconstruct the original image from the encoded sequence.
- A *Decoder* to turn encoded sequences into desired predictions.

The training consists of two steps:

1. Step 1: train the Tokenizer and Encoder.
   (a) Drop randomly some tokens after tokenizing.
   (b) Update parameters by differences between the original and reconstructed inputs which does not require annotations.
2. Step 2: train the Decoder.
   (a) Freeze the Tokenizer and Encoder.
   (b) Train the Decoder in the usual supervised way.

SegV3 can be easily adapted to this self-supervisorisation pipeline. If further experiments show promising result, it will be a good sign for reducing annotation requirements in medical image area in a systematical way.

[9] In the table the folds are named from 1 to 5 rather than 0 to 4, and I picked fold 1.
[10] I downloaded the fold 0 model, which presumably is the model for fold 1 in the paper.
[11] https://github.com/Project-MONAI/research-contributions/tree/main/SwinUNETR/BRATS21.

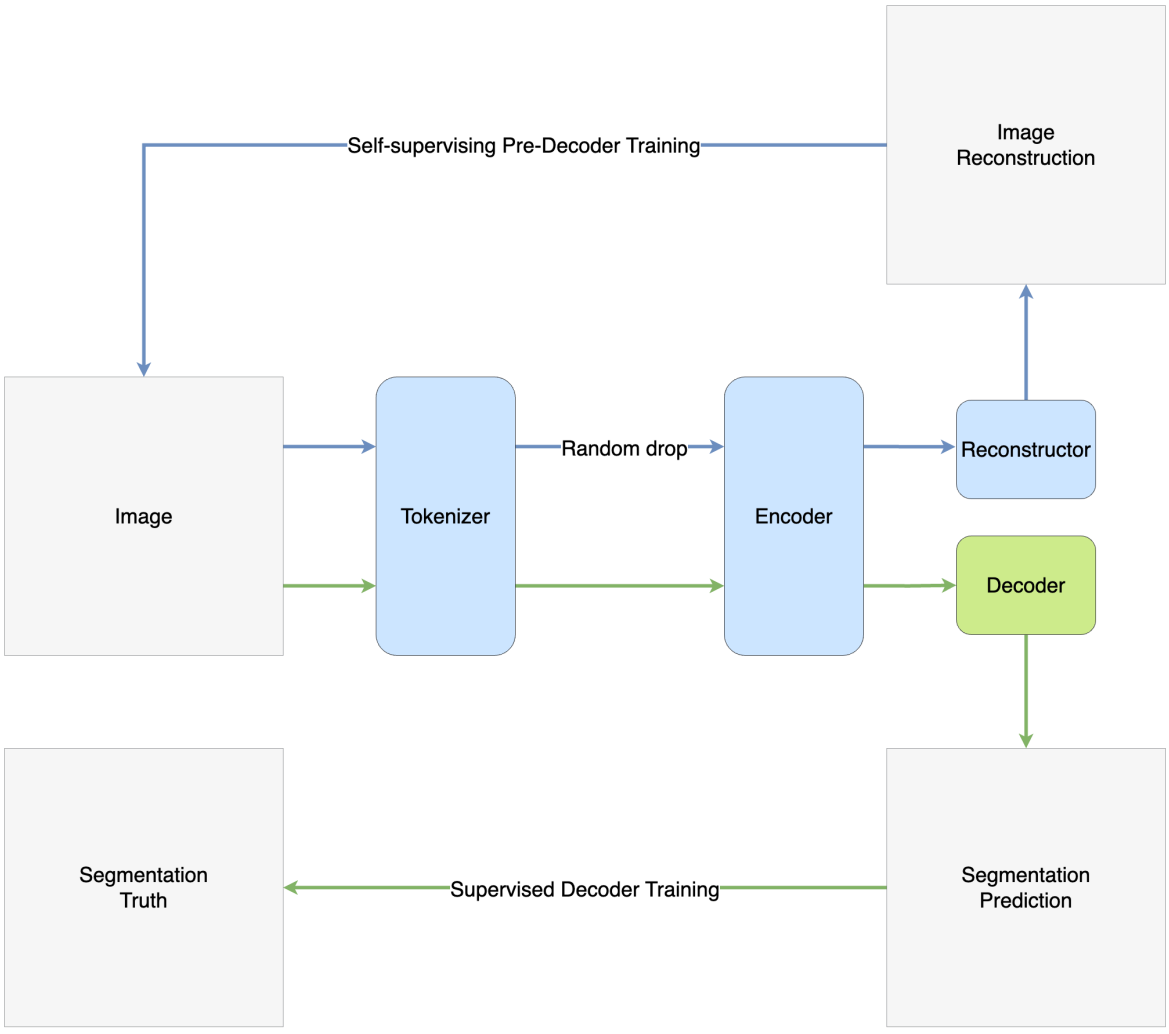

Figure 5: Self-supervisorisation.

References

[DBK+20] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2020.

[HCX+21] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners, 2021.

[HNT+22] Ali Hatamizadeh, Vishwesh Nath, Yucheng Tang, Dong Yang, Holger Roth, and Daguang Xu. Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images, 2022.

[RFB15] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[YHS+21] Yung-Hsu Yang, Thomas E. Huang, Min Sun, Samuel Rota Bulò, Peter Kontschieder, and Fisher Yu. Dense prediction with attentive feature aggregation, 2021.