Utrecht University

ViTs vs. CNNs for 3D Medical Image Segmentation: Are Transformers All You Need?

Minor Research Project
MSc Medical Imaging

Pablo Xabier Arregui García

Examination Committee:

Dr. Matteo Maspero
Assistant Professor, UMC Utrecht

Dr. Leticia Gallardo Estrella
Lead Deep Learning Engineer, Thirona B.v.

Daily Supervisors:

Michail Tsiaousis
Deep Learning Engineer, Thirona B.v.

Laurens Weitkamp
Deep Learning Engineer, Thirona B.v.

Utrecht, 10th July 2023

ViTs vs. CNNs for 3D Medical Image Segmentation: Are Transformers All You Need?

P.X. Arregui García
MSc Medical Imaging, Utrecht University

***Abstract*—The influence of Vision Transformers (ViTs) is increasing in the field of medical image segmentation. In recent years, several papers have presented ViT-based architectures that outperform the previously state-of-the-art CNNs (such as nnU-Net). An example of such is the Swin UNet Transformer (or Swin UNETR), which, in combination with a self-supervised pre-training scheme, has outperformed other CNN and ViT-based architectures in multiple segmentation tasks. However, there are certain design and configuration choices that may aid the ViT to achieve this performance. In this paper, we perform an objective comparison between Swin UNETR and U-Net, comparing both networks in an equal resource setting. We explore two downscaling approaches to balance the parameter count of Swin UNETR, making it closer to the U-Net in this aspect. We measure the ViT's performance loss due to downscaling, as well as the gain obtained from using pre-trained weights for the encoder. Additionally, we assess whether residual blocks aid the Swin UNETR or U-Net to obtain superior performance. Our results show that in the framework used in this study, both U-Net and Swin UNETR show comparable results, with the CNN-based network achieving a slightly superior (1%) DSC. The downscaled ViT models show a decrease of 1.4% in DSC, while pre-training improves the outcome of the original Swin UNETR by 1.6%. Residual functionality proves to aid the pre-trained Swin UNETR with an increase of 3.6% in DSC, while only improving U-Net's DSC by 0.8%. In the constrained resource setting of this study, the U-Net proves to obtain similar performance to Swin UNETR, while employing fewer GPU resources and improving inference speed.**

***Index Terms*—ViTs, CNNs, 3D U-Net, SwinUNETR, self-supervised pretraining, model comparison**

1. INTRODUCTION

Over the course of several years, convolutional neural networks (CNNs) have consistently emerged as the predominant choice for medical image segmentation. In 2015, Ronnenberg et al. [1] presented the U-Net architecture, a CNN used to segment medical images. This architecture achieved high performance, worked well with a limited amount of data, and was computationally efficient. Since then, the U-Net has dominated the field of medical image segmentation and has been the object of multiple architectural modifications aiming to boost its performance.

In 2021, Isensee et al. [2] presented the nnU-Net, a self-configuring medical image segmentation algorithm that used a standard U-Net as its architecture. In their work, the authors showed that a simple and self-configured (referring to the capacity of adapting its own configuration and parameters automatically) U-Net was capable of achieving state-of-the-art performance. Hence, all signs indicated that, as it happened in general for computer vision, CNNs would perpetuate their dominant role in the domain of medical image segmentation.

However, in 2020, Dosovitskiy et al. [3] presented a new architecture for computer vision, i.e., the vision transformer (ViT). This architecture was based on the Transformer presented by Vaswani et al. [4], which was adapted to solve computer vision tasks by converting images into sequences of patches. The main novelty behind this computer vision architecture was that it no longer relied on convolutions to extract features, but was mainly based on self-attention mechanisms. The reason for using self-attention rather than convolutions is that the first can better capture global features, which aids when learning long-range dependencies in images. Additionally, although transformers require larger amounts of data, they excel when being pre-trained in a self-supervised manner [5, 6], which further boosts their performance without the burden of extra annotations. In this way, when pre-trained on large-scale datasets, the ViT showed excellent performance on image classification tasks.

Although the first ViTs were exclusively designed for image classification, in 2021 Chen et al. presented the TransUNet [7], a CNN-ViT-hybrid network that showed promising capabilities for medical image segmentation. Subsequently, a multitude of ViT-based segmentation networks have been introduced in the field [8–13]. Among the various approaches that were presented, the UNet TRansformer (UNETR) [10] demonstrated state-of-the-art performance, surpassing the competing networks (including nnU-Net) in the BTCV dataset [14].

Despite this, the UNETR has two limitations. First, it only supports patches (or tokens) of a fixed scale, which might be unsuitable for dealing with variable-scale features. Secondly, the computational complexity of its self-attention mechanism is quadratic to image size, which greatly affects its efficiency when dealing with high-resolution images. To deal with these shortcomings, Hatamizadeh et al. presented the Swin UNet TRansformer (Swin UNETR) [15], an architecture that used the novel Swin ViT [16] as its backbone. Unlike conventional ViTs, Swin ViT's computational complexity is linear to image size, and it works with patches at multiple scales. By employing this cutting-edge Swin ViT, and using a self-supervised pre-training scheme on a large cohort of CT scans [17], SwinUNETR surpassed the performance achieved by its predecessor (UNETR) in the BTCV dataset.

In spite of the superior performance shown by Swin UNETR, certain observations can be made concerning its

configuration and design. Firstly, it is noteworthy that the number of parameters in Swin UNETR is considerably greater than its CNN-based counterparts, having three times as many parameters as nnU-Net for the BTCV challenge. Secondly, it is surprising that more than 80% of Swin UNETR's total parameters belong to its CNN-based decoder, which is considerably larger than the decoder employed by a standard U-Net. Consequently, Swin UNETR has a diminutive ViT encoder and an extensive CNN decoder, which contradicts the notion of the ViT encoder providing the network with an advantage over CNNs. Thirdly, residual blocks have been added in the skip connections, filtering the features that pass from encoder to decoder, which is a design choice that sets this network apart from both U-Net and UNETR. Finally, training Swin UNETR as reported in [17] requires many more computational resources than training a CNN like nnU-Net [2]. Thus, these factors raise the question whether the greater performance of Swin UNETR is genuinely a consequence of its Swin ViT encoder's superiority.

The purpose of this study is to perform an objective (equal computational resources) comparison between a standard U-Net and the novel Swin UNETR, to determine the comparative advantage of Vision Transformers over Convolutional Neural Networks in the context of medical image segmentation. Additionally, we aim to compare these architectures in a similar parameter number scenario, mainly focusing on reducing Swin UNETR's parameter count. Furthermore, we want to assess the impact of downscaling on Swin UNETR's performance, as well as the effect of using pre-trained weights to initialize its ViT encoder. Finally, we intend to understand the effect of the decoder-related design choices that set Swin UNETR apart from the U-Net. Thus, the main contributions of this work are:

1) We present a method for downscaling Swin UNETR, which halves its parameter count, does not affect its ViT encoder and enables the re-use of pre-trained encoder weights. This downscaled model can potentially benefit from the performance gain provided by a pre-trained encoder without the need to repeat the time-consuming pre-training process.
2) We perform a comparison between U-Net and the original-sized Swin UNETR in a constrained resource setting that is equal for both networks. Additionally, we quantify the impact of downscaling Swin UNETR, as well as provide a comparison between U-Net and our downscaled Swin UNETR with approximately equal parameter counts.
3) We quantify the performance provided by residual blocks in the skip connections and decoder in Swin UNETR. Furthermore, we also study the effect of implementing such features in the U-Net.
4) We provide a comparison between U-Net and Swin UNETR in terms of computational burden and efficiency, determining which network is more cost-effective when hardware resources are limited.

All experiments are conducted on the Pancreas Tumor Dataset, taken from the Medical Segmentation Decathlon (MSD) [18].

2. RELATED WORK

2.1. 3D self-supervised pre-training

Self-supervised representation learning consists of using proxy tasks to facilitate neural network feature learning from unlabelled data. When applied to medical images, this allows neural networks to encode region-of-interest-aware information, which increases their performance on downstream tasks (such as segmentation). In [19], Taleb et al. presented the first 3D self-supervised pre-training scheme for medical image segmentation, which consisted of the solution of five proxy tasks. Using this method, the authors pre-trained a 3D U-Net-like encoder, which was posteriorly plugged into a U-Net decoder in the fine-tuning stage.

In [17], Tang et al. used a similar method to pre-train Swin UNETR's Swin ViT encoder. In this case, the authors used three proxy tasks (masked volume inpainting, image rotation, and contrastive coding), and formulated the problem with a multi-objective loss function. Again, for the solution of the downstream segmentation tasks, a U-Net decoder was attached to the pre-trained Swin ViT encoder.

2.2. Residual blocks in skip connections and decoder

In a U-Net, skip connections enable propagating the spatial information that gets lost during the pooling (or downsampling) operations. However, some works [20, 21] argue that there is a large semantic gap between encoder and decoder features, which causes these long-range skip connections to be sub-optimal. To alleviate such a semantic gap, in [21], the authors modified the skip connections of a U-Net by adding a stack of residual convolutional blocks that processed the features coming from the encoder before being concatenated to the features at the decoder. These modified skip connections, which were named as *Respaths*, enabled the U-Net to obtain better results, as well as to converge faster.

Although it is not mentioned by the authors in [15], we assume that the choice of adding the residual blocks in Swin UNETR's skip connections follows the same purpose. Additionally, Swin UNETR also contains residual blocks in the decoder (*Resblocks*). Such blocks have been introduced into several deep learning models [22–24] to solve gradient vanishing and explosion problems, and are considered to smooth the loss surface and ease the training of deep neural networks [25].

3. METHODS

3.1. Dataset and data splits

The dataset used in this work is the Pancreas Tumor Dataset from the Medical Segmentation Decathlon (MSD) [18, 26]. This dataset contains 420 portal venous phase CT scans of patients undergoing resection of pancreatic masses. The images have a variable number of slices, with a slice thickness of 2.5 mm and an in-plane resolution of 512×512. For each scan, expert delineations of the pancreas and the tumor are

provided, which are encoded in multi-label images. Thus, this task consists of a 2-class segmentation problem (pancreas and tumor).

In the MSD challenge, this dataset is divided into 281 images for training and 139 for testing. However, the labels of the 139 images belonging to the test set are not publicly available, which renders it impossible to compare models statistically. In light of this, an internal split of the dataset was created, dividing the 281 images that originally constituted the training set into 160 images for training, 40 for validation, and 81 for testing. This way, the resulting 81-image test set was used to perform quantitative evaluations among the models utilized in this study.

3.2. Data preprocessing

Initially, images were cropped based on the patient's body contour. To achieve this, the images were binarized using Otsu thresholding [27] considering two classes: foreground (body contour) and background (air, scanner bed ...). Then, pixels wrongly classified as foreground were removed using connected component analysis and keeping the largest connected component (which was assumed to be the body contour). Next, a bounding box was applied to the borders, and the images were cropped.

After cropping, all images were resampled to a target spacing, corresponding to the median spacing of the images on the training set. As images in the pancreas tumor dataset are highly anisotropic (out-of-plane spacing is more than three times greater than in-plane spacing), resampling was done in two steps: first, it was applied in-plane with third-order spline interpolation, and then out-of-plane using nearest neighbor interpolation. By treating the out-of-plane axis separately, resampling artifacts were suppressed [28].

Once resampled, images were normalized to a range of [-1, 1] before being fed to the network. Given that CT intensity values are quantitative, and therefore reflect tissue-related physical properties, it is advantageous to retain their information. To achieve this, a global normalization scheme was applied to all images, using the global (referring to the whole training set) foreground mean and standard deviation. Then, images were normalized by subtracting the mean and dividing by the standard deviation. In addition, clipping was applied with the 0.5 and 99.5 percentiles of the global foreground voxels. The described process of resampling and normalization follows the scheme proposed in [2, 28].

Furthermore, due to memory limitations of fitting an entire 3D image from the pancreas tumor dataset in the GPU employed in this study, all networks were trained using fixed-size image patches (cropped sub-volumes of the entire 3D image). The choice for the patch sampling strategy, the number of patches per image, and the selected patch size are discussed in Section 3.4.2.

3.3. Network architectures

This section presents the two network architectures employed in this study, i.e., U-Net and Swin UNETR, as well as the strategies used to downscale Swin UNETR.

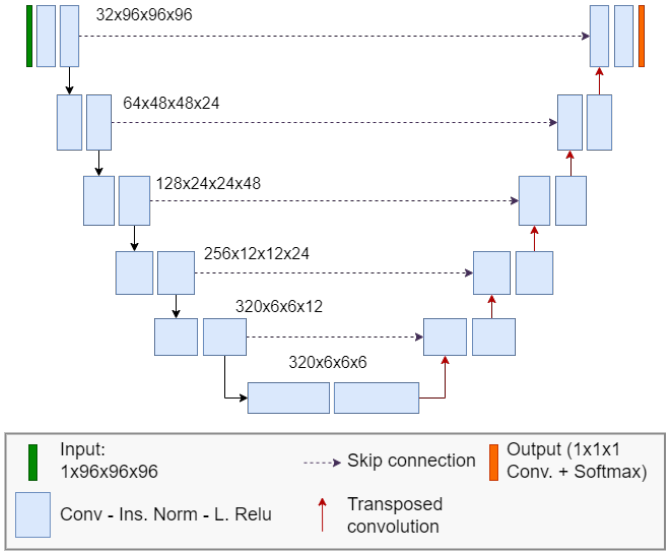


Fig. 1: Dynamic U-Net architecture. It is practically equal to a standard U-Net, except for the capacity to use anisotropic kernels and strides. This implementation also includes the possibility to substitute the standard convolutional blocks with residual convolutional blocks.

3.3.1. U-Net

The U-Net model developed in this study adopts a dynamic U-Net architecture. The dynamic U-Net was originally proposed by Isensee et al. in [2] and stands out from the original version due to its ability to handle kernels of anisotropic sizes and strides. In this work, we utilize a version of the dynamic U-Net from the open-source library MONAI [29], depicted in Figure 1. This re-implementation of the architecture extends its capabilities by enabling the use of residual connections in its convolutional blocks.

Apart from the aforementioned modifications, the dynamic U-Net can be considered a standard U-Net in practical terms. The encoder functions as a conventional CNN, extracting features and providing classification information [30]. Each block within the encoder consists of two convolutional layers, followed by instance normalization and an activation function. The number of feature channels progressively doubles with each depth level, reaching a maximum of 320. Unlike the original U-Net architecture, downsampling is accomplished by applying convolutions with a stride of 2, as opposed to utilizing a 2-strided max-pooling operation.

The decoder employs the features extracted from the encoder to generate a segmentation in the original input spacing, assigning each pixel to one of the classes. Each block in the decoder comprises a transposed convolutional layer, followed by concatenation with the corresponding feature map from the encoder (obtained through skip connections), and two convolutional layers, each followed by instance normalization and an activation function. In this case, the transposed convolutional layer reduces the number of feature channels while upsampling the feature map. After the final decoder

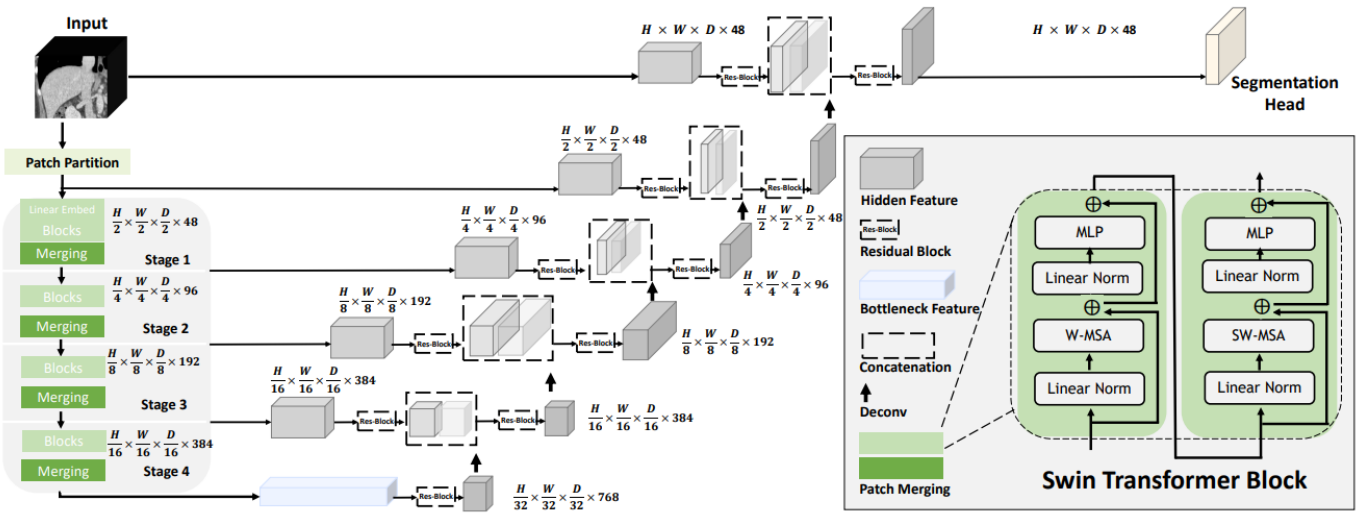


Fig. 2: Swin UNETR architecture, taken from [15]. It shares U-Net's structure, but the convolutional encoder is replaced by a Swin ViT encoder. This encoder contains the novel Swin transformer blocks. Within these blocks, W-MSA and SW-MSA refer to regular and shifted window multi-head self-attention respectively, and MLP means multi-layer perceptron. The decoder is similar to a U-Net's, except for the substitution of conventional convolutional blocks with residual blocks. The skip connections are also modified, by adding residual blocks before feature concatenation.

block, a 1x1x1 convolutional layer is applied to match the number of feature channels to the desired number of structures for segmentation. Finally, a softmax function is employed to derive the probability of each pixel belonging to one of the classes.

3.3.2. Swin UNETR

The Swin UNETR is a ViT-based architecture that was developed by Hatamizadeh et al. [15]. The implementation used in this work has been taken from MONAI [29]. As shown in Fig. 2, Swin UNETR has the characteristic U-shaped structure, replacing U-Net's fully convolutional encoder with its Swin ViT counterpart.

The encoder contains an initial patch partitioning layer that divides the input into 2x2x2 3D patches, which will represent the tokens (a term commonly used in natural language processing). This is followed by four stages. The first stage begins with a linear embedding layer that projects the patches into a C-dimensional space. In [17], C is set to 48, and the pre-trained weights shared by the authors are only compatible with this embedding space size. The linear embedding layer is followed by two Swin transformer blocks for feature transformation, which employ the highly efficient window-based multi-head self-attention mechanism proposed in [16]. After this, a patch-merging layer performs downsampling and doubles the feature channels. The subsequent three stages are identical to the first stage, with the exception of the initial linear embedding layer. To better understand the functionality of Swin ViTs, reading [3, 16] is encouraged.

Concerning the decoder, its structure is similar to the original U-Net architecture, although there are some modifications. On the one hand, the conventional up-convolutional blocks are replaced by residual up-convolutional blocks. On the other hand, skip-connections are modified by including a residual block before encoder-decoder feature concatenation, comprising two 3x3x3 convolutions followed by instance normalization. This way, the feature maps originated in the encoder go through a sequence of convolutional layers before being concatenated with the features in the decoder.

3.3.3. Downscaling Swin UNETR

Swin UNETR has approximately $62M$ parameters, which doubles the parameter count of the U-Net architecture used in this study. Moreover, while U-Net's parameters are evenly distributed between the encoder and decoder, Swin UNETR holds more than 85% of its total parameters (around $54M$) in its convolutional decoder. Taking these factors into account, in this study, we aim to reduce the parameter number of Swin UNETR, matching the U-Net in this aspect, with the additional benefit of creating a ViT model that is less resource-intensive. This was achieved in two different ways (one of which allows to reuse the pre-trained weights in the downscaled model), which are discussed in the following paragraphs.

Reducing the embedding space dimension

The most straightforward way of reducing Swin UNETR's parameters is to lower its embedding space dimension, a method already used by [15, 17] to reduce Swin UNETR's parameter number.

As mentioned before, the linear embedding layer projects the sequence of patches into a C-dimensional embedding space. At each stage in the encoder, this embedding space dimension is doubled, which causes an increase in the number of parameters. Therefore, by reducing the dimensionality of

the initial embedding space C, its dimensionality at each stage will also become smaller, and the number of parameters of both the encoder and decoder will be reduced. Originally, $C = 48$, which results in $62M$ parameters. By setting $C = 36$, the number of parameters is reduced to roughly $35M$, enabling a similar parameter number comparison between U-Net and Swin UNETR. We will refer to this "tiny" version of Swin UNETR as Swin UNETR$_T$.

However, reducing the network's parameters in this way has some disadvantages. Firstly, the number of parameters in the already diminutive encoder is reduced, which may affect its capabilities. Secondly, by using $C = 36$, the weights obtained in [17] from self-supervised pre-training can no longer be used, as they are only compatible with the encoder in the original embedding space size $C = 48$. Having a pre-trained downscaled model could be favorable, as it could benefit from the performance boost provided by the pre-learned feature embeddings while greatly reducing the computational burden with respect to the original-sized model. Nevertheless, repeating the pre-training process is time-consuming, which is why an additional downscaling method that exploits the possibility of re-using the available pre-trained weights was developed.

Reducing the bottleneck dimension

The majority of Swin UNETR's parameters are localized at the bottleneck, specifically, in the residual block before the up-convolution operation. This block consists of 2 convolutions, each of which has $N^3 \times C_{in} \times C_{out} = 3^3 \times 768 \times 768 \approx 16M$ parameters, where N is the kernel size and C_{in} and C_{out} are the number of input and output feature channels respectively. That is, in total, this block adds $32M$ parameters to the network. However, by just halving the size of the feature dimension at the bottleneck, the parameter number of this block would be reduced by 4, resulting in a network with a total of $38M$ parameters. Thus, the second method that we propose to reduce the parameter number consists in doing precisely this.

To achieve this without modifying the encoder, we added a 1x1x1 convolutional block that processes the feature map at the output of the encoder and halves the size of its feature dimension from 768 to 384. As the encoder is left unchanged, this downscaled version of Swin UNETR can benefit from the pre-trained weights, while employing fewer computational resources than the original size model. Besides, this approach enables to downscale Swin UNETR without affecting the capacity of its ViT encoder. We will designate this downscaled version of Swin UNETR as Swin UNETR$_B$.

3.4. Implementation details

3.4.1. Loss function

The loss function used for both parameter tuning and training was the equally weighted sum of Dice loss and cross-entropy loss (DiceCE). This loss function was developed to reduce class and output imbalance [31], and it is commonly used in medical image segmentation.

3.4.2. Parameters and configuration

Concerning the patch sampling strategy, the training patches were sampled using a 1 to 1 foreground-to-background-class ratio. Additionally, the number of patches sampled per image was calculated by multiplying the batch size (2) by the number of batches per epoch (fixed at 240) and dividing by the training set length (160 images) $240 \times 2/160 = 3$. In this way, all images in the training set are sampled at every epoch.

In addition, the tuning of the weight decay and learning rate was done by running grid searches, using RayTune [32]. Each parameter combination's performance was evaluated based on the DiceCE loss on the validation set. The weight decay did not show much effect on network performance and was kept to 10^{-5} for all models. For every model (see Appendix A), a learning rate of 10^{-4} proved to be the best choice.

The rest of the parameters shown in Table I were set based on the original configuration choices of nnU-Net [2] and Swin UNETR [17] for the pancreas dataset of the MSD challenge. In the case of Swin UNETR, although the original paper used a larger batch size (exact number undisclosed), a batch size of 2 was chosen due to hardware constraints. For the U-Net, instead of using the original patch size of (224, 224, 40) used for the pancreas dataset, a patch size of (96, 96, 96) was used. This was done to compare the efficiency of both models objectively, as input size greatly affects the computational burden of a model.

TABLE I: Model parameters and configuration.

Parameter	U-Net	Swin UNETR
Spacing (mm)	0.81, 0.81, 2.5	1, 1, 1
Batch size	2	
Patch size	96, 96, 96	
Optimizer	Adam W	
Activation	L. ReLU	GELU / L. ReLU
Normalization	Instance	Layer / Instance
Training batches	240	
Validation batches	60	
Data augmentation	[2]	[17]
Patch sampling	50% fg - 50% bg	
Num. patch samples	3	
Learning rate	10^{-4}	
Weight decay	10^{-5}	

3.4.3. Training and inference

All models were trained for 500 epochs, and no learning rate scheduler or early stopping was used during training. The reason for not employing a learning rate scheduler is that a simple configuration was enough for the purpose of this study, which is to perform a comparison between models. Next, inference was performed on the test set using the weights corresponding to the epoch with the highest validation Dice. A sliding-window approach was used, keeping the window size equal to the patch size used during training. An overlapping of 0.5 between windows (or patches) was set and the superimposed area was merged using Gaussian blending, to limit the segmentation errors in the edges of the patch [2, 33, 34]. During parameter optimization, training, and inference, an

NVIDIA GeForce RTX 2080 with 12 GB of VRAM is used, and mixed precision was employed [35].

3.4.4. Postprocessing

After performing the sliding window inference and stacking together the predicted patches, some postprocessing was applied to the entire predicted image. As commonly done in medical image segmentation, for each class, the largest connected component was kept, and the rest of the smaller objects were removed. Then, the image was converted to the original spacing, and the borders were padded to match the original image size.

3.5. Evaluation metrics

For the evaluation of model performance, the metrics used were:

- **Dice score (DSC)** It quantifies the overlap between two volumes, defined as twice the intersection of the volumes divided by their union.
- **Normalized surface distance (NSD)** It measures the overlap between two boundaries. Given the boundary of the reference S_A, the boundary of the prediction S_B, and a tolerance value τ, we define the border regions of the reference $\beta_A^{(\tau)}$ and prediction $\beta_B^{(\tau)}$ as the pixels within a distance τ from the respective boundaries. Then, according to [36], the NSD at a tolerance τ is defined as:

$$NSD(A,B)^{(\tau)} = \frac{|S_A \cap \beta_B^{(\tau)}| + |S_B \cap \beta_A^{(\tau)}|}{|S_A| + |S_B|}$$

  In this work, a tolerance of 5 mm is used, since it is the value used in [18] for the pancreas tumor dataset.
- **Hausdorff distance 95 percentile (HD95)** Given the 95% percentile of all shortest distances for all points from the reference boundary to the prediction boundary $d_{95}(A,B)$ and vice-versa $d_{95}(B,A)$, the HD95 is calculated as [36]:

$$HD95(A,B) = max\left\{d_{95}(A,B), d_{95}(B,A)\right\}$$

  This metric was calculated for completeness but reported only as part of the appendix.

3.6. Experiments

After inferring and evaluating the models on the test set, several experiments and model comparisons were conducted. To evaluate the significance of all the performed comparisons, the Wilcoxon signed-rank test [37] was used. This test was chosen for two reasons: it is suitable for paired samples (all models are inferred on the same test set), and it is non-parametric (it does not assume that results are normally distributed).

3.6.1. Performance of downscaled Swin UNETR

Firstly, Swin UNETR$_T$ and Swin UNETR$_B$ were compared, to see which of the downscaled networks performed better and validate the presented alternative method. Secondly, we compared both of them with the original Swin UNETR, in order to see the performance lost from downscaling the models.

3.6.2. Effect of pre-training on Swin UNETR

The Swin UNETR models with a non-modified encoder (Swin UNETR$_B$ and Swin UNETR) were run in two different configurations: first training from scratch and then initializing the encoder with the self-supervised pre-trained weights provided in [17]. To refer to the models run using pre-trained weights, we will use the upper score PT (Swin UNETR$_B^{PT}$ and Swin UNETR^{PT}). The performance gain provided by pre-training was determined by comparing Swin UNETR models with and without pre-training (for both the downscaled and the original size model).

3.6.3. U-Net vs. Swin UNETR

We compared the downscaled Swin UNETR models to the U-Net, in order to see how these two perform in a similar parameter-number scenario. Additionally, we compared the original Swin UNETR models to the U-Net (with and without pre-trained encoders), to see if, in a resource-constrained setting that is equal for both networks, the ViT is still superior to the CNN-based network, as reported in [15, 17].

3.6.4. Effect of Respaths and Resblocks

We tested the effect of removing Swin UNETR's *Respaths* and substituting the decoder *Resblocks* with conventional convolutional blocks, assessing the impact of such design choices on Swin UNETR's performance. Additionally, we evaluated whether the implementation of *Resblocks* (substituting all convolutional blocks in the encoder and decoder) and the addition of *Respaths* can boost the performance of the U-Net used in this study.

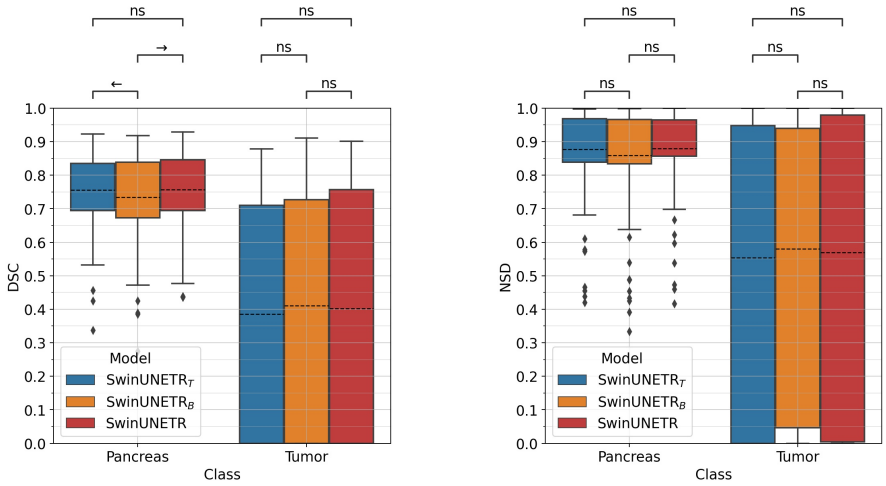
3.6.5. Model complexity, resource utilization, and efficiency

In addition to all the aforementioned quantitative comparisons of model performance, these were compared based on their complexity, resource utilization, and efficiency. For such a comparison, we measured their number of parameters, total size (GPU VRAM during training), FLOPs (number of floating-point multiplication-and-addition operations), and throughput (images processed by the network per second). These last two were measured during inference. All measurements were conducted using a batch size of 2, input image patches with shape (96, 96, 96), and mixed precision [35]. Additionally, for Swin UNETR models, gradient checkpointing was used, which affects the FLOPs and the model size.

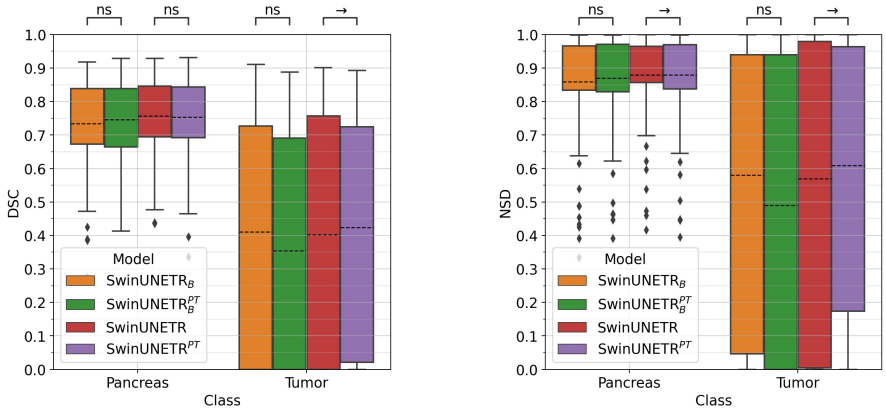
4. RESULTS

4.1. Performance of downscaled Swin UNETR

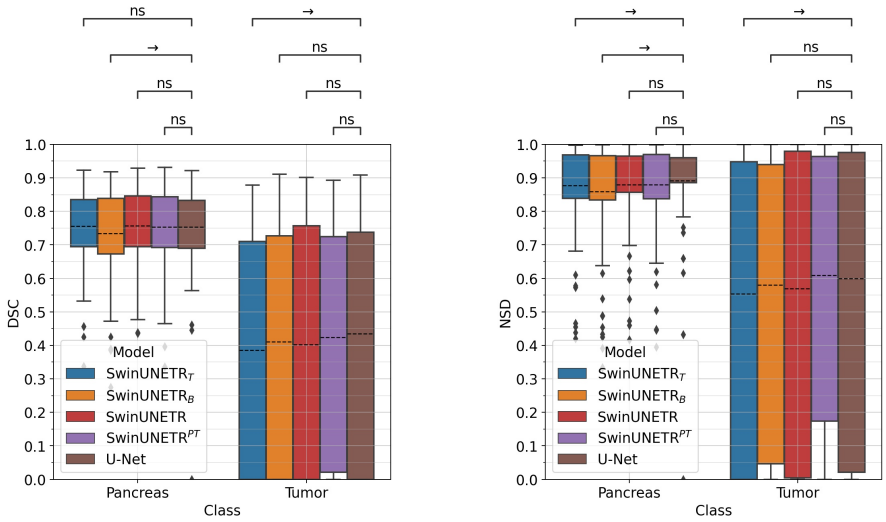
The results of this experiment are shown in Fig. 3a. First of all, when comparing both downscaled models, no big performance difference was observed. In terms of DSC, Swin UNETR$_T$ performed statistically better ($p < 0.05$) than Swin UNETR$_B$ for the pancreas (with a DSC of 0.755 vs. 0.735), and no statistical difference was observed for the tumor class. In the case of NSD, no statistically significant difference in model performance was observed. Despite the difference



(a) Performance of downscaled Swin UNETR.



(b) Effect of pre-training.



(c) U-Net vs. Swin UNETR.

Fig. 3: In each case, DSC (left) and NSD (right) are shown for both the pancreas and tumor. The black dashed line in each boxplot represents the average. On top of each figure, the results of the Wilcoxon signed rank test used for model comparison are shown. "ns": no statistically significant difference ($p > 0.05$) between compared models. "→": the model to the right is significantly better than the model to the left with $p < 0.05$. "←": the model to the left is significantly better than the model to the right with $p < 0.05$.

TABLE II: Effect of Respaths and Resblocks on model performance. The numbers represent the average score of models for a specific metric and class. Green represents an improvement of the model with residual functionality vs. the same model without it. Red represents the opposite, i.e., that the model with Resblock and Respaths performs worse than the same model without them. An asterisk (*) represents that the difference in performance is statistically significant ($p < 0.05$) according to the Wilcoxon signed-rank test.

Metric	**DSC**						**NSD**					
Class	**Pancreas**		**Tumor**		**Avg.**		**Pancreas**		**Tumor**		**Avg.**	
Residual	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**	**No**	**Yes**
Swin UNETR$_T$	0.734	**0.755***	0.421	**0.385***	0.577	0.570	0.855	0.877	0.572	0.554	0.714	0.715
Swin UNETR$_B$	0.748	**0.735***	0.414	0.410	0.581	0.572	0.879	0.859	0.572	0.580	0.726	0.720
Swin UNETR$_B^{PT}$	0.757	0.746	0.401	0.354	0.579	0.550	0.885	0.870	0.580	0.490	0.732	0.680
Swin UNETR	0.739	0.757	0.410	0.402	0.575	0.580	0.865	0.880	0.579	0.570	0.722	0.725
Swin UNETR^{PT}	0.745	**0.753***	0.362	**0.424***	0.553	**0.589***	0.875	**0.880***	0.523	**0.610***	0.699	**0.745***
U-Net	0.754	0.746	0.435	0.452	0.594	0.599	0.892	0.885	0.599	0.614	0.746	0.749

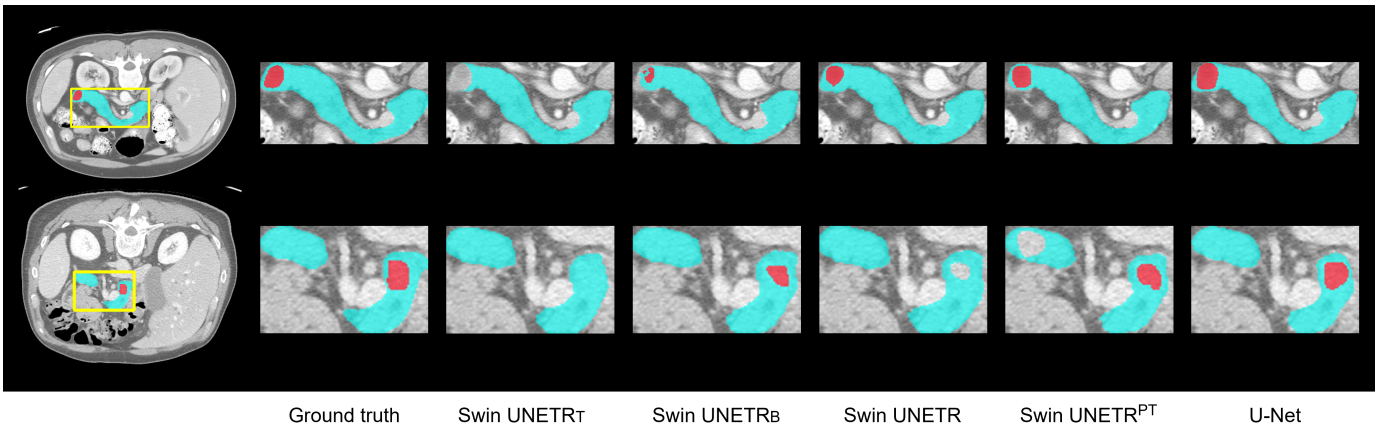


Fig. 4: Visual comparison of models for two representative subjects of the test set. The pancreas is shown in blue and the tumor is in red.

not being statistically significant, Swin UNETR$_B$ obtained a higher average DSC and NSD than Swin UNETR$_T$ for the tumor class. This was also visible in some representative cases (Fig. 4), where it was observed that Swin UNETR$_T$ failed to identify the tumor, as opposed to the other downscaled model. Averaging the scores over both classes, Swin UNETR$_B$ achieved a relative DSC 0.4% higher and an NSD 0.6% higher than Swin UNETR$_T$.

When comparing both downscaled models to the original size Swin UNETR, it is clear that downscaling had a weak impact on model performance. Compared to Swin UNETR$_B$, Swin UNETR showed a statistically significant improvement in terms of DSC for the pancreas (0.735 vs. 0.757 respectively). However, no significant difference was observed when comparing the original size model to Swin UNETR$_T$. Concerning the NSD, the performance comparison between the downscaled and the original-sized models yielded no significance. Overall, while downscaling almost halved the models' number of parameters, it only reduced the DSC and NSD by 1.4% and 1% respectively.

4.2. Effect of pre-training on Swin UNETR

When assessing the impact of pre-training (Fig. 3b), it is clear that its effect differs for the downscaled and original-sized models. Regarding Swin UNETR$_B$, pre-training did not show a statistically relevant difference for any class and metric. In fact, when averaging the scores over both classes and due to the low performance for the tumor class, the pre-trained model obtained a DSC 3.9% lower and an NSD 5.5% lower than the one trained from scratch.

However, in the case of the original-sized Swin UNETR, the picture is rather different. For this network, pre-training showed a significant improvement for the tumor class in terms of DSC and improved the NSD significantly for both classes. The positive effects of pre-training can also be observed when comparing both model predictions visually (Fig. 4), where it is seen that the pre-trained version segmented the tumor more accurately. In general terms, Swin UNETR^{PT} achieved a DSC and NSD 1.6% and 2.8% higher than its non-pre-trained counterpart.

4.3. U-Net vs. Swin UNETR

Comparing all Swin UNETR models to the U-Net (Fig. 3c), it is observed that both architectures showed similar performance.

Considering the downscaled models, their performance was not far from the CNN's, although the latter was superior. On one side, the NSD obtained by Swin UNETR$_T$ was significantly inferior to the U-Net for both classes. In terms

TABLE III: Model complexity, resource utilization, and efficiency. Models are divided into two groups: original (unmodified models) and modified (models where residual functionality has been pruned *NoRB* or added *RB*). The total size represents the GPU memory footprint of the models during training. FLOPs and throughput are measured at inference. The throughput is not based on the processing of full images but on the forward pass of image patches of the size (96, 96, 96). For Swin UNETR models, gradient checkpointing is used, which affects the FLOPs and the total size.

Group	Model	Params (M)	Total size (GB)	FLOPs (G)	Throughput (images/s)
Original	U-Net	30.7	3.9	624.5	37.3
	Swin UNETR$_T$	35.1	6.3	353.9	14.6
	Swin UNETR$_B$	37.4	8.5	626.5	12.6
	Swin UNETR	62.2	8.6	628.3	12.5
Modified	U-Net$_{RB}$	41.6	7.0	910.3	23.4
	Swin UNETR$_{T-NoRB}$	33.4	5.1	321.1	15.8
	Swin UNETR$_{B-NoRB}$	34.4	6.8	568.2	14.4
	Swin UNETR$_{NoRB}$	59.2	6.9	570.0	13.8

of DSC, the ViT was significantly inferior for the tumor class (0.385 vs. 0.435). On the other side, Swin UNETR$_B$ obtained a significantly lower DSC and NSD for the pancreas, but there was no significant difference for the tumor class. Swin UNETR$_B^{PT}$ was excluded from these comparisons, as it was clearly inferior to the non-pre-trained version (as seen in Section 4.2). Averaging the scores obtained for both classes and both downscaled Swin UNETR models, the U-Net obtained a DSC and NSD 4% and 3.8% higher than ViT-based architectures.

With respect to the original-sized Swin UNETR models (Swin UNETR and Swin UNETR^{PT}), their performance was comparable to the CNN's, with no statistically significant differences for any class and metric. Despite this, the pre-trained model obtained a higher average NSD (0.610) for the tumor class. The conservative superiority in this metric can also be observed in Fig. 4, where it is visible that the U-Net over-segmented the tumor, while Swin UNETR^{PT} tended to under-segment this class. Averaging the scores over both classes, U-Net was 2.6% superior to Swin UNETR in terms of DSC (0.594 vs. 0.580) and 2.9% better in terms of NSD (0.746 vs. 0.725). Compared to the pre-trained Swin UNETR, U-Net's average DSC was slightly higher (1%) and the average NSD was almost the same for both (only 0.1% superior for the CNN).

4.4. Effect of *Respaths* and *Resblocks*

Regarding the effect of *Respaths* and *Resblocks* on Swin UNETR's performance (Table II), there is a clear difference between the downscaled and original size models. Residual functionality negatively impacted the three downscaled models, reducing the average DSC for all of them. In terms of average NSD, only Swin UNETR$_T$ benefited from this design choice. Overall, the downscaled models with residual functionality suffered a decrease of 2.6% in DSC and NSD.

However, for the original-sized models (Swin UNETR and Swin UNETR^{PT}) *Resblocks* and *Respaths* increased model performance for both averaged metrics. While the performance gain was not statistically significant for the non-pre-trained model, Swin UNETR^{PT} was the ViT-based model that benefited most from the addition, obtaining a statistically significant improvement for all classes in both metrics. This improvement was especially relevant for the tumor class, with a relative increase in DSC and NSD of 17.1% and 16.6% respectively. In general, this feature resulted in an increase of 3.6% in DSC and 3.4% in NSD for original-sized Swin UNETR models.

With respect to the U-Net, substituting its convolutional blocks by *Resblocks* and adding residual blocks in the skip connections marginally increased the average DSC (0.599) and NSD (0.749) by 0.8% and 0.4%, although no statistically significant improvement in performance was observed.

4.5. Model complexity, resource utilization, and efficiency

The complexity, resource utilization, and efficiency of the different models are shown in Table III. Firstly, it is clear that the downscaling of Swin UNETR to obtain Swin UNETR$_T$ caused a considerable decrease in the model's requirements and efficiency, reducing the GPU memory footprint by 1.8 GB, approximately halving the FLOPs and improving the inference speed by 16.8%. However, the other downscaled model did not benefit from any of these improvements with respect to the original model, with the only significant difference being the number of parameters itself.

Secondly, when comparing U-Net and Swin UNETR models, it is evident that the first employed 50% fewer GPU memory, while being more than two times faster than Swin UNETR$_T$ and almost three times faster than Swin UNETR and Swin UNETR$_B$ at inference. However, the CNN also had a higher FLOP-to-parameter ratio with respect to the ViTs, roughly duplicating the latter in this aspect.

Finally, for Swin UNETR models, residual functionality increased the parameter number and the model size by 6.3% and 24.4% on average. Moreover, it caused an increase of 10% in the FLOP count, as well as an inverse effect in the throughput, with a 10% reduction. For the U-Net, the changes caused by *Resblocks* and *Respaths* were far more pronounced, with an increase of 35.5%, 79.5%, and 45.76% in parameters, model size, and FLOPs and a decrease of 37.3% in inference speed.

5. DISCUSSION

In the paper published by Tang et al. [17], Swin UNETR demonstrated state-of-the-art performance, surpassing the

competing architectures. Although its advantage over CNNs was attributed to the superiority of the Swin ViT encoder and the newly proposed pre-training scheme, the authors of the study did not consider differences in the respective resource settings, design choices, and network parameter counts. In this study, we aimed to compare Swin UNETR and U-Net in an objective manner, trying to investigate the effect of different design choices in the ViT. Two methods to balance the parameter count of Swin UNETR with respect to the U-Net were presented, one of which still allowed the re-use of pre-trained weights for the encoder, and we quantified the effect of pre-training in both original and downscaled Swin UNETR models. Furthermore, we removed the residual blocks in Swin UNETR's decoder and assessed whether they provided the network with an advantage. In addition, residual functionality was included in the U-Net, to see if it provided the network with some additional performance. Finally, we compared both architectures in terms of computational overhead and efficiency, providing some insight into which of both networks to choose for 3D medical image segmentation tasks in a constrained resource scenario.

Regarding the downscaling of ViTs, both models derived from the presented approaches yielded comparable results (although Swin UNETR$_T$ achieved a statistically significant improvement for the pancreas), thereby affirming the effectiveness of the proposed alternative downscaling method, which involved the inclusion of a 1x1x1 convolutional block in the bottleneck. Notably, this downscaling technique not only reduced the parameters but also facilitated the utilization of a pre-trained encoder, eliminating the need to repeat the time-consuming pre-training process. Remarkably, despite the significant reduction in parameter count, both downscaled ViTs demonstrated performance that was nearly comparable with the original-sized models. In addition, Swin UNETR$_T$ was more computationally efficient than the original model, employing fewer GPU resources and accelerating inference times. However, these benefits were not observed for Swin UNETR$_B$, which was as resource-intensive as its original size counterpart. Despite this, having such a downscaled model with a pre-trained encoder could be beneficial in some scenarios where limited data is available and an over-parameterized model would lead to overfitting.

In spite of the potential advantages of using a pre-trained encoder in a downscaled Swin UNETR model, the performance of Swin UNETR$_B^{PT}$ was inferior to its non-pre-trained equivalent's. This disparity could be attributed to the dimensionality reduction applied to the feature embeddings learned during pre-training (as these are projected from an original feature space with 768 dimensions to a feature space with 384 dimensions), potentially impacting the quality of these features. However, the results were different when considering the original-sized Swin UNETR model, as pre-training exhibited improvements in performance. This observed performance enhancement of approximately 1.6% in DSC aligned with the findings of [17] for the pancreas dataset, where the authors reported a 1% improvement in DSC associated with the utilization of a pre-trained encoder. It is worth mentioning that in the original paper, the pancreas dataset was among those that benefited the least from pre-training, which suggests that, had we performed our experiments on another dataset, we could have seen bigger effects from pre-training.

Concerning the comparison between U-Net and the original-sized Swin UNETR models, both exhibited similar performance, albeit with the CNN demonstrating a slight superiority in terms of average DSC and NSD. These results seemingly contradict the remarkable superiority shown by the pre-trained Swin UNETR in [17], where it achieved a 5% improvement in DSC over nnU-Net. In that study, the ViT-based network obtained a DSC of 0.707, which strongly contrasts with the DSC of 0.589 obtained in our work. However, it is worth noting that this performance disparity was anticipated, as the data splits used in our work are different from the original study (we only train with 60% of the data, and use part of the remaining data as the test set). Furthermore, the configuration employed for our Swin UNETR implementation differs from that described in the paper. To attain the state-of-the-art results reported in [17], Swin UNETR was trained for more epochs (exact number undisclosed), employing large batch sizes (specifics undisclosed), and incorporating techniques such as cross-validation with model ensembling, test time augmentation, and the inclusion of additional partially or fully annotated data not related to the challenge. However, our study's Swin UNETR configuration was based on a tutorial repository provided by the authors, which did not include all those techniques. The implementation of all those additional features would have required more hardware resources, while the scope of our work is comparing both networks (U-Net and Swin UNETR) in a constrained resource setting, as objectively as possible. In addition to the aforementioned, we showed that the U-Net employed 50% less GPU memory while being almost three times faster at inference, highlighting the resource efficiency of the CNN.

When comparing the U-Net with the downscaled ViTs, the CNN exhibited superior performance. This suggests that, when matching the parameter count of Swin UNETR with that of the U-Net, the latter holds an advantage. However, it is important to approach this conclusion with caution, as these results are specific to the configuration used in this study, which differs from the original paper's configuration, as mentioned earlier. Additionally, the CNN's memory footprint was almost 40% smaller than Swin UNETR$_T$'s, as well as being more than twice faster at inference, showing the high speed and low hardware requirements of the CNN even when balancing the parameter count of the ViT. However, compared to Swin UNETR$_T$, the FLOP count was almost two times larger for the U-Net, which seems to contradict the fact that the CNN is faster. In fact, we observed in our study that, while an increment in FLOPs caused a decrease in throughput within the same architecture, when comparing different architectures this tendency did not hold. A possible explanation for this discrepancy could be that the total number of FLOPs in a model is an inconsistent predictor of real execution time, due

to the highly parallel nature of tensor operations and hardware accelerators [38].

With respect to the addition of *Resblocks* and *Respaths*, these generally showed a negative effect on the downscaled models' performance. Although we are not sure of the reason behind this unexpected outcome, a possible explanation could be overfitting. We observed that residual blocks made Swin UNETR models converge faster (see Appendix C). As no early stopping is used, the training may continue after a plateau in the validation loss and the weights used for inference may be based on an overfitted model which does not perform that well on the test set. This is only a hypothesis and would require further research. For the original size ViTs, residual functionality showed a positive influence, which was especially significant for the pre-trained model, demonstrating that this design choice had some impact on Swin UNETR in [17]. In terms of computational overhead, residual Swin UNETR models increased GPU memory usage by 24.4%, as well as increased FLOPs and reduced inference speed by 10%. On the other side, despite the increase in performance shown by residual U-Nets in some studies [21, 39], our U-Net$_{RB}$ did not obtain a significant improvement over the standard model. While there are some examples of residual U-Nets used for natural images in the literature, we did not find any study reporting benefits of such networks for medical image analysis. Thus, the potential benefits of using a residual U-Net for 3D medical image segmentation are unclear. Furthermore, the residual U-Net increased the GPU memory requirements by 3.1 GB, while making the model almost 40% slower during inference. Therefore, we consider that the marginal benefits shown in performance (or any further potential benefits) do not justify the utilization of the residual U-Net presented in this work in favor of the standard U-Net.

Despite all the aforementioned, we acknowledge some limitations in our study. First of all, the study of the effect of matching U-Net's and Swin UNETR's parameters is only one-sided, and the effect of upscaling the U-Net to match the ViT is not investigated. Secondly, an alternative downscaling method, which has not been exploited, would have been to remove the bottleneck block in Swin UNETR. Although this might also have some negative effects on the network's performance, exploring this possibility might have been interesting, as it would have allowed us to use pre-training weights without affecting the dimensionality of the pre-learned feature embeddings. Thirdly, to study the effect of *Resblocks* and *Respaths*, we removed both of them at the same time. A possible alternative could be to remove them individually, which would give a better insight into their isolated contributions, and possibly explain the reason behind their combined poor performance on downscaled Swin UNETR models. All these three investigation lines could be the object of future research. Finally, the use of cross-validation would have made our models less subjective to random variations, contributing to the robustness of the presented results, while a more extensive parameter optimization or the use of a learning rate scheduler, could have improved the outcome of some models.

6. CONCLUSION

In this study, we performed a comparison between the CNN-based U-Net and the ViT-based Swin UNETR in the context of medical image segmentation. To the best of our knowledge, this was the first work in which these two networks were compared in a framework with equal hardware resources, and matching design and configuration choices. Furthermore, we presented two methods to approximate Swin UNETR's parameter count to the U-Net, one of which still enabled the use of the pre-trained weights employed in the larger model. In the used resource-constrained scenario, the CNN and the original size pre-trained Swin UNETR obtained comparable performance, with the former achieving a subtle improvement of 1% in DSC. Despite downscaling reducing the ViT's parameter count almost by half, these downsized models only suffered a loss of 1.4% in DSC. However, when compared to the U-Net, their performance was significantly lower. Concerning the effect of pre-training, although it showed no benefit on the downscaled model, it provided the original-sized network with an increase of 1.6% in DSC. Regarding *Respaths* and *Resblocks*, we demonstrated that these design choices play an important role in the original-sized Swin UNETR's performance, while they showed slight but not significant improvements for the U-Net. Finally, we proved that the U-Net is a less resource-intensive and more efficient model, requiring less GPU memory and being up to three times faster at inference. All in all, despite the fact that with more parameters and hardware resources, Swin UNETR may have an advantage over U-Net, the latter is nowadays more accessible, while still achieving excellent performance. Therefore, even if ViTs are rather promising, we think that it is premature to dismiss CNN-based architectures in the context of medical image segmentation.

REFERENCES

[1] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2015, pp. 234–241.

[2] F. Isensee, P. F. Jaeger, S. A. Kohl, J. Petersen, and K. H. Maier-Hein, "nnu-net: a self-configuring method for deep learning-based biomedical image segmentation," *Nature methods*, vol. 18, no. 2, pp. 203–211, 2021.

[3] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[6] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever *et al.*, "Improving language understanding by generative pre-training," 2018.

[7] J. Chen, Y. Lu, Q. Yu, X. Luo, E. Adeli, Y. Wang, L. Lu, A. L. Yuille, and Y. Zhou, "Transunet: Transformers make strong encoders for medical image segmentation," *arXiv preprint arXiv:2102.04306*, 2021.

[8] H.-Y. Zhou, J. Guo, Y. Zhang, L. Yu, L. Wang, and Y. Yu, "nnformer: Interleaved transformer for volumetric segmentation," *arXiv preprint arXiv:2109.03201*, 2021.

[9] H. Cao, Y. Wang, J. Chen, D. Jiang, X. Zhang, Q. Tian, and M. Wang, "Swin-unet: Unet-like pure transformer for medical image segmentation," *arXiv preprint arXiv:2105.05537*, 2021.

[10] A. Hatamizadeh, Y. Tang, V. Nath, D. Yang, A. Myronenko, B. Landman, H. R. Roth, and D. Xu, "Unetr: Transformers for 3d medical image segmentation," in *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 2022, pp. 574–584.

[11] G. Xu, X. Zhang, Y. Fang, X. Cao, W. Liao, X. He, and X. Wu, "Levit-unet: Make faster encoders with transformer for biomedical image segmentation."

[12] Y. Xie, J. Zhang, C. Shen, and Y. Xia, "Cotr: Efficiently bridging cnn and transformer for 3d medical image segmentation," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part III 24*. Springer, 2021, pp. 171–180.

[13] W. Wang, C. Chen, M. Ding, H. Yu, S. Zha, and J. Li, "Transbts: Multimodal brain tumor segmentation using transformer," in *Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part I 24*. Springer, 2021, pp. 109–119.

[14] B. Landman, Z. Xu, J. Igelsias, M. Styner, T. Langerak, and A. Klein, "Miccai multi-atlas labeling beyond the cranial vault–workshop and challenge," in *Proc. MICCAI Multi-Atlas Labeling Beyond Cranial Vault—Workshop Challenge*, vol. 5, 2015, p. 12.

[15] A. Hatamizadeh, V. Nath, Y. Tang, D. Yang, H. R. Roth, and D. Xu, "Swin unetr: Swin transformers for semantic segmentation of brain tumors in mri images," in *International MICCAI Brainlesion Workshop*. Springer, 2022, pp. 272–284.

[16] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 10 012–10 022.

[17] Y. Tang, D. Yang, W. Li, H. R. Roth, B. Landman, D. Xu, V. Nath, and A. Hatamizadeh, "Self-supervised pre-training of swin transformers for 3d medical image analysis," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 20 730–20 740.

[18] M. Antonelli, A. Reinke, S. Bakas, K. Farahani, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze, O. Ronneberger, R. M. Summers *et al.*, "The medical segmentation decathlon," *Nature communications*, vol. 13, no. 1, p. 4128, 2022.

[19] A. Taleb, W. Loetzsch, N. Danz, J. Severin, T. Gaertner, B. Bergner, and C. Lippert, "3d self-supervised methods for medical imaging," *Advances in neural information processing systems*, vol. 33, pp. 18 158–18 172, 2020.

[20] Z. Zhang, X. Zhang, C. Peng, X. Xue, and J. Sun, "Exfuse: Enhancing feature fusion for semantic segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 269–284.

[21] N. Ibtehaz and M. S. Rahman, "Multiresunet: Rethinking the u-net architecture for multimodal biomedical image segmentation," *Neural networks*, vol. 121, pp. 74–87, 2020.

[22] J. Zhang, Y. Zheng, D. Qi, R. Li, X. Yi, and T. Li, "Predicting citywide crowd flows using deep spatio-temporal residual networks," *Artificial Intelligence*, vol. 259, pp. 147–166, 2018.

[23] H. Chen, Q. Dou, L. Yu, J. Qin, and P.-A. Heng, "Voxresnet: Deep voxelwise residual networks for brain segmentation from 3d mr images," *NeuroImage*, vol. 170, pp. 446–455, 2018.

[24] L. H. Shehab, O. M. Fahmy, S. M. Gasser, and M. S. El-Mahallawy, "An efficient brain tumor image segmentation based on deep residual networks (resnets)," *Journal of King Saud University-Engineering Sciences*, vol. 33, no. 6, pp. 404–412, 2021.

[25] H. Li, Z. Xu, G. Taylor, C. Studer, and T. Goldstein, "Visualizing the loss landscape of neural nets," *Advances in neural information processing systems*, vol. 31, 2018.

[26] A. L. Simpson, M. Antonelli, S. Bakas, M. Bilello, K. Farahani, B. Van Ginneken, A. Kopp-Schneider, B. A. Landman, G. Litjens, B. Menze *et al.*, "A large annotated medical image dataset for the development and evaluation of segmentation algorithms," *arXiv preprint arXiv:1902.09063*, 2019.

[27] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE transactions on systems, man, and cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[28] F. Isensee, P. F. Jäger, S. A. Kohl, J. Petersen, and K. H. Maier-Hein, "Automated design of deep learning methods for biomedical image segmentation," *arXiv preprint arXiv:1904.08128*, 2019.

[29] M. J. Cardoso, W. Li, R. Brown, N. Ma, E. Kerfoot, Y. Wang, B. Murrey, A. Myronenko, C. Zhao, D. Yang *et al.*, "Monai: An open-source framework for deep learning in healthcare," *arXiv preprint arXiv:2211.02701*, 2022.

[30] N. Siddique, S. Paheding, C. P. Elkin, and V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *Ieee Access*, vol. 9, pp. 82 031–82 057, 2021.

[31] S. A. Taghanaki, Y. Zheng, S. K. Zhou, B. Georgescu, P. Sharma, D. Xu, D. Comaniciu, and G. Hamarneh, "Combo loss: Handling input and output imbalance in multi-organ segmentation," *Computerized Medical Imaging and Graphics*, vol. 75, pp. 24–33, 2019.

[32] R. Liaw, E. Liang, R. Nishihara, P. Moritz, J. E. Gonzalez, and I. Stoica, "Tune: A research platform for distributed model selection and training," *arXiv preprint arXiv:1807.05118*, 2018.

[33] C. Innamorati, T. Ritschel, T. Weyrich, and N. J. Mitra, "Learning on the edge: Explicit boundary handling in cnns," *arXiv preprint arXiv:1805.03106*, 2018.

[34] N. Pielawski and C. Wählby, "Introducing hann windows for reducing edge-effects in patch-based image segmentation," *PloS one*, vol. 15, no. 3, p. e0229839, 2020.

[35] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh *et al.*, "Mixed precision training," *arXiv preprint arXiv:1710.03740*, 2017.

[36] A. Reinke, M. D. Tizabi, C. H. Sudre, M. Eisenmann, T. Rädsch, M. Baumgartner, L. Acion, M. Antonelli, T. Arbel, S. Bakas *et al.*, "Common limitations of image processing metrics: A picture story," *arXiv preprint arXiv:2104.05642*, 2021.

[37] R. F. Woolson, "Wilcoxon signed-rank test," *Wiley encyclopedia of clinical trials*, pp. 1–3, 2007.

[38] D. Langerman, A. Johnson, K. Buettner, and A. D. George, "Beyond floating-point ops: Cnn performance prediction with critical datapath length," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2020, pp. 1–9.

[39] Z. Zhang, Q. Liu, and Y. Wang, "Road extraction by deep residual u-net," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 5, pp. 749–753, 2018.

APPENDIX

A. Model characteristics and parameter counts

TABLE IV: Models compared in this study and their parameter count. T = Tiny, DS = Downscaled, RB = Resblocks + Respaths, $NoRB$ = No Resblocks and Respaths, and PT = Pre-trained.

Model	Residual	Pretrained	Parameters
U-Net	No	No	30.7 M
U-Net$_{RB}$	Yes	No	41.6 M
Swin UNETR$_{T-NoRB}$	No	No	33.4 M
Swin UNETR$_{T}$	Yes	No	35.1 M
Swin UNETR$_{B-NoRB}$	No	No	34.4 M
Swin UNETR$^{PT}_{B-NoRB}$	No	Yes	34.4 M
Swin UNETR$_{B}$	Yes	No	37.4 M
Swin UNETR$^{PT}_{B}$	Yes	Yes	37.4 M
Swin UNETR$_{NoRB}$	No	No	59.2 M
Swin UNETR$^{PT}_{NoRB}$	No	Yes	59.2 M
Swin UNETR	Yes	No	62.2 M
Swin UNETR^{PT}	Yes	Yes	62.2 M

In this work, 8 different models were compared: U-Net, U-Net$_{RB}$, Swin UNETR$_{T}$, Swin UNETR$_{T-NoRB}$, Swin UNETR$_{B}$, Swin UNETR$_{B-NoRB}$, Swin UNETR, and Swin UNETR$_{NoRB}$.

The Swin UNETR models for which the encoder was not modified (Swin UNETR$_{B}$, Swin UNETR$_{B-NoRB}$, Swin UNETR, and Swin UNETR$_{NoRB}$) are run in two different configurations: first training from scratch and then initializing the encoder with the self-supervised pre-trained weights provided in [17]. To specify when those models have been run using pre-trained weights, we will use the upper score PT (Swin UNETR$^{PT}_{B}$, Swin UNETR$^{PT}_{B-NoRB}$, Swin UNETR^{PT}, and Swin UNETR$^{PT}_{NoRB}$). All models and their characteristics are shown in Table IV.

B. Performance metrics for all models

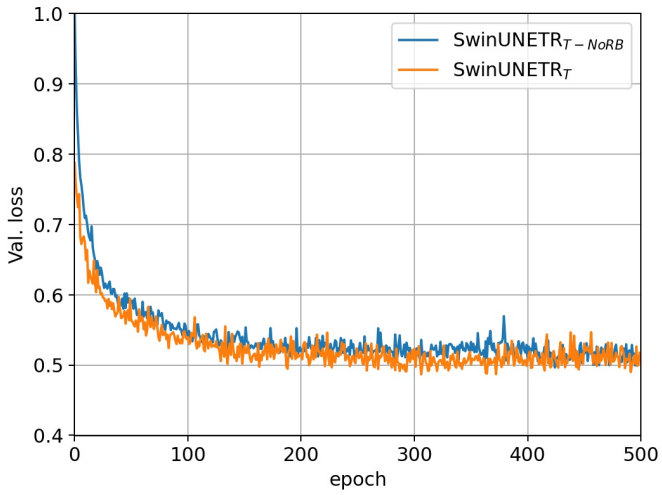
TABLE V: Average results obtained by every model for all metrics, including HD95. For each metric and class, the model with the highest average score is highlighted in bold.

Group	Model	Dice ↑			NSD ↑			HD95 (mm) ↓		
		Pancreas	Tumor	Avg.	Pancreas	Tumor	Avg.	Pancreas	Tumor	Avg.
Non residual	Swin UNETR$_{T-NoRB}$	0.734	0.421	0.577	0.855	0.572	0.714	17.7	28.9	23.3
	Swin UNETR$_{B-NoRB}$	0.748	0.414	0.581	0.879	0.572	0.726	15.6	30.2	22.9
	Swin UNETR$^{PT}_{B-NoRB}$	0.757	0.401	0.579	0.885	0.580	0.732	14.6	36.8	25.7
	Swin UNETR$_{NoRB}$	0.739	0.410	0.575	0.865	0.579	0.722	17.7	33.0	25.3
	Swin UNETR$^{PT}_{NoRB}$	0.745	0.362	0.553	0.875	0.523	0.699	15.1	34.2	24.7
	U-Net	0.754	0.435	0.594	**0.892**	0.599	0.746	**14.1**	33.3	23.7
Residual	Swin UNETR$_{T}$	0.755	0.385	0.570	0.877	0.554	0.715	15.9	33.3	24.6
	Swin UNETR$_{B}$	0.735	0.410	0.572	0.859	0.580	0.720	18.4	27.0	22.7
	Swin UNETR$^{PT}_{B}$	0.746	0.354	0.550	0.870	0.490	0.680	16.7	41.8	29.3
	SwinUNETR	**0.757**	0.402	0.580	0.880	0.570	0.725	15.2	36.2	25.7
	Swin UNETR^{PT}	0.753	0.424	0.589	0.880	0.610	0.745	16.0	**24.8**	**20.4**
	U-Net$_{RB}$	0.746	**0.452**	**0.599**	0.885	**0.614**	**0.749**	15.9	26.2	21.1

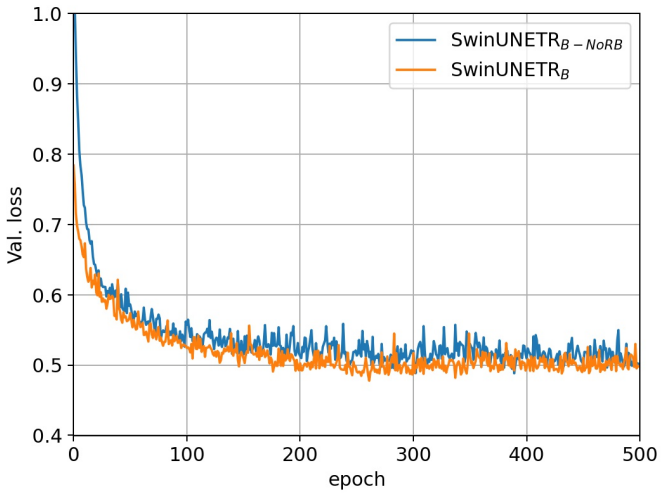
Table V shows the results obtained for all models inferred on the test set. In terms of average DSC and NSD over both classes, U-Net$_{RB}$ achieves the best scores, mainly due to its superior performance for the pancreas tumor class. However, when considering HD95, the pre-trained Swin UNETR is the best-performing model, improving by 5.3mm the HD95 obtained by its non-pre-trained equivalent. This shows again that in the original Swin UNETR model, pre-training has a notable impact.

C. Residual blocks and model convergence

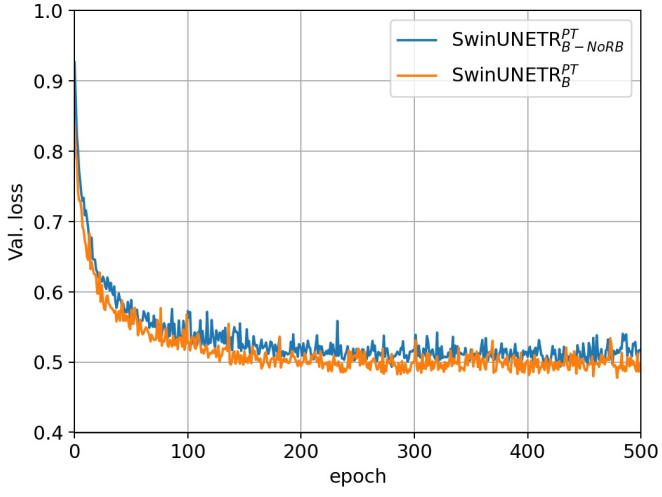
Even if the addition of *Resblocks* and *Respaths* showed different effects for different models on test set performance, it overall improved convergence for all models. As observed in Fig. 5, residual functionality made ViT-based models converge faster, reducing the starting point of the validation loss and making it reach a lower value during training. In light of this, a possible explanation for the low performance of the downscaled residual ViTs might be overfitting. That is, residual blocks



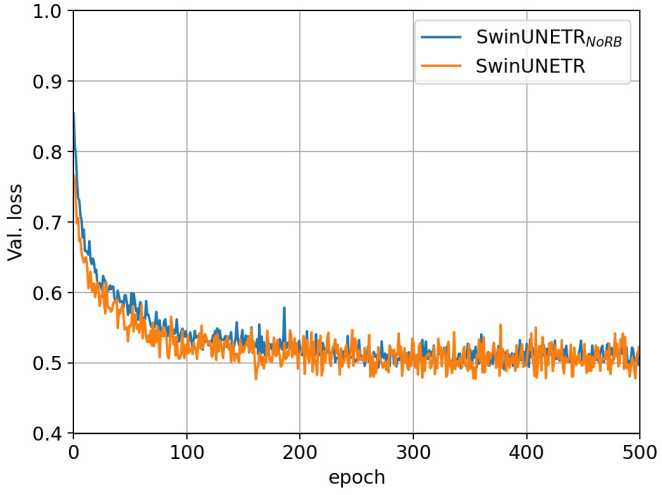
(a) Swin UNETR$_T$



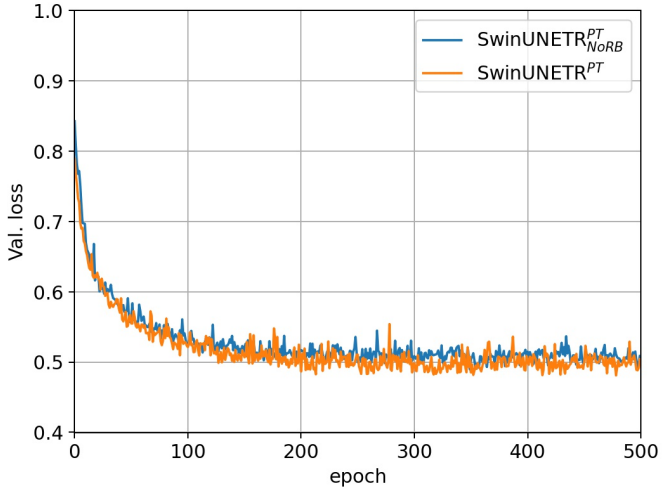
(b) Swin UNETR$_B$



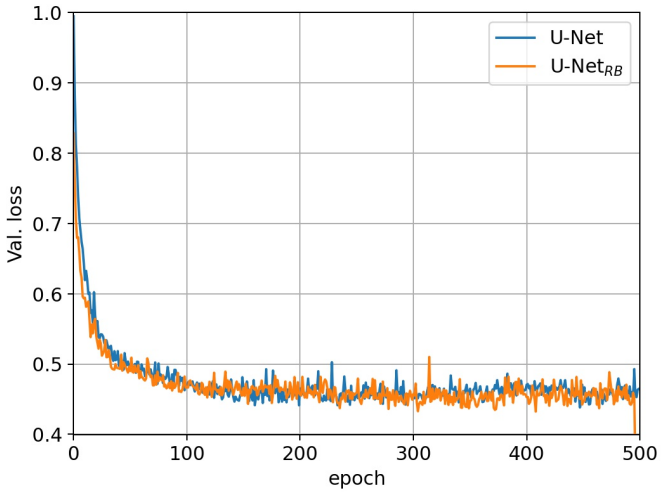
(c) Swin UNETR$_B^{PT}$



(d) Swin UNETR



(e) Swin UNETR^{PT}



(f) U-Net

Fig. 5: Validation loss against epoch for every model with (orange) and without (blue) residual functionality.

make the model converge faster, after which the training continues but there is no further improvement. Hence, the weights used for inference are based on an overfitted model, which does not perform that well on the test set. Some design choices like the use of a learning rate scheduler or early stopping may help to prevent this. In any case, to prove any of this, further research would be required.