Appendix for UniMiss

Yutong Xie[1][0000−0002−6644−1250], Jianpeng Zhang[2], Yong Xia[2], and Qi Wu[1⋆]

[1] The University of Adelaide, Australia
[2] School of Computer Science and Engineering, Northwestern Polytechnical University, China
yutong.xie678@gmail.com;qi.wu01@adelaide.edu.au

1 Overview

In this document, we provide more discussions and experimental details to supplement the main submission. We first continue to discuss the necessity of switchable patch embedding (SPE) module (Section 2). We then give more details for the downstream tasks, including the implementation details and architectures (Section 3). Finally, we provide an intuitive explanation of the proposed volume-slice consistency mechanism (Section 4).

2 Necessity of SPE (Cont.)

To further explain the necessity of the SPE module, we compared the pyramid U-like medical Transformer (MiT) to two variants without a SPE module. For the variant 1, we directly flatten the 2D/3D images to a sequence based on the pixels/voxels level and then use a linear layer for the embedding. Such a crude flattening operation suffers the very high computation complexity and memory requirements, especially for 3D images. Thus, it is hard to perform the variant 1 for quantitative comparisons. For the variant 2, we perform a naive embedding strategy to reduce the complexity. We first down-sample (for encoder)/up-sample (for decoder) the 2D/3D images by using a parameter free interpolation, then flatten them into a sequence based on the pixels/voxels level, and finally use a linear layer for both 2D and 3D embedding. The results in Table 1 show that MiT with the SPE module is significantly superior to the naive embedding strategy (*i.e.* variant 2) whenever with or without using the pre-training. It suggests that our SPE is better than the parameter-free interpolation and linear layer. The reason may be that the strided convolution with a large kernel is able to model the local continuity of 2D/3D images, which cannot be implemented by the linear layer.

3 Downstream Tasks

3.1 Implementation Details

In Table 2, we provide the implementation details of six downstream datasets, including the task type, modality, number of training and test cases, loss func-

⋆ Corresponding author.

Table 1. Segmentation performance of MiT and its two variants without a SPE module on BCV offline test set (3D CT).

Methods		SPE	Dice
Random initialization	Variant 1	No	unaffordable
	Variant 2	No	73.31
	Ours	Yes	79.93
UniMiSS pre-training	Variant 1	No	unaffordable
	Variant 2	No	76.65
	Ours	Yes	84.99

Table 2. Implementation details of downstream tasks. Seg: Segmentation; Cls: Classification; CE: Cross-entropy loss; off: offline test set; on: online test set.

Dataset	BCV	RICORD	JSRT	ChestXR	CHAOS	ISIC
Task	Seg	Cls	Seg	Cls	Seg	Seg
Modality	3D CT	3D CT	2D X-ray	2D X-ray	3D MRI	2D Dermoscopic
Training data	24	182	124	17,955	16	2000
Test data	6 (off)+20 (on)	45	123	3,430	4	600
Loss	Dice+CE [4]	CE	Dice+CE	CE	Dice+CE [4]	Hybrid loss [6]
Patch size	48×192^2	64×128^2	224^2	224^2	$48 \times 192 \times 256$	224^2
Augmentation	✓	✓	✓	✓	✓	✓
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
Learning rate	0.0001	0.00001	0.0001	0.0001	0.0001	0.0001
Batch size	2	8	32	32	2	16
Iterations	25,000	14,000	10,000	17,000	50,000	37,500

tion, patch size, batch size, optimizer, learning rate, and maximum iterations. Note that we randomly split 25% training scans as a validation set to select the hyper-parameters of UniMiSS in the ablation study. We use the online data augmentation to alleviate the over-fitting of UniMiSS on training data. We augment 2D images via random cropping and zooming, random rotation, shear, shift, and horizontal/vertical flip. As for 3D images, we perform random rotation, scaling, flipping, adding white Gaussian noise, Gaussian blurring, adjusting rightness and contrast, simulation of low resolution, and Gamma transformation [4]. All the downstream experiments were performed on a NVIDIA GTX 2080Ti GPU.

3.2 Architectures of MiT and ResUnet

Figure 1 shows the detailed settings of the MiT network. The MiT encoder follows a progressive shrinking pyramid Transformer, as done in [5]. It consists explicitly of four stages, each of which is composed of a SPE module and several stacked Transformers. In each stage, the SPE module down-samples the input features and generates the dimension-specific embedded sequence. Notably, we append an extra learnable SSL token [1,2] to the patch embedded sequence. The SSL token is similar to the [CLS] token in ViT, which is able to aggregate information from the whole patch embedding tokens via the self-attention. The resultant sequences, combined with the learnable positional embedding, are inputted into the following Transformers for the long-term dependency modeling. Each Transformer layer includes a self-attention module and a feed-forward network (FFN) with two hidden layers. To reduce the computational cost and enable MiT to process high-resolution images, we follow the spatial-reduction attention (SRA) layer to reduce the spatial complexity [5]. MiT has a symmetric decoder structure that consists of three stages. In each stage, the input feature map is first up-sampled by the SPE module, and then refined by the stacked Transformer layers. Besides, we also add skip connections between the encoder and decoder to keep more low-level but high-resolution information. We devise two MiT by changing the number of Transformer layers, namely MiT-7 and MiT-22. Noticed that default MiT-22 is used in the main submission unless otherwise specified.

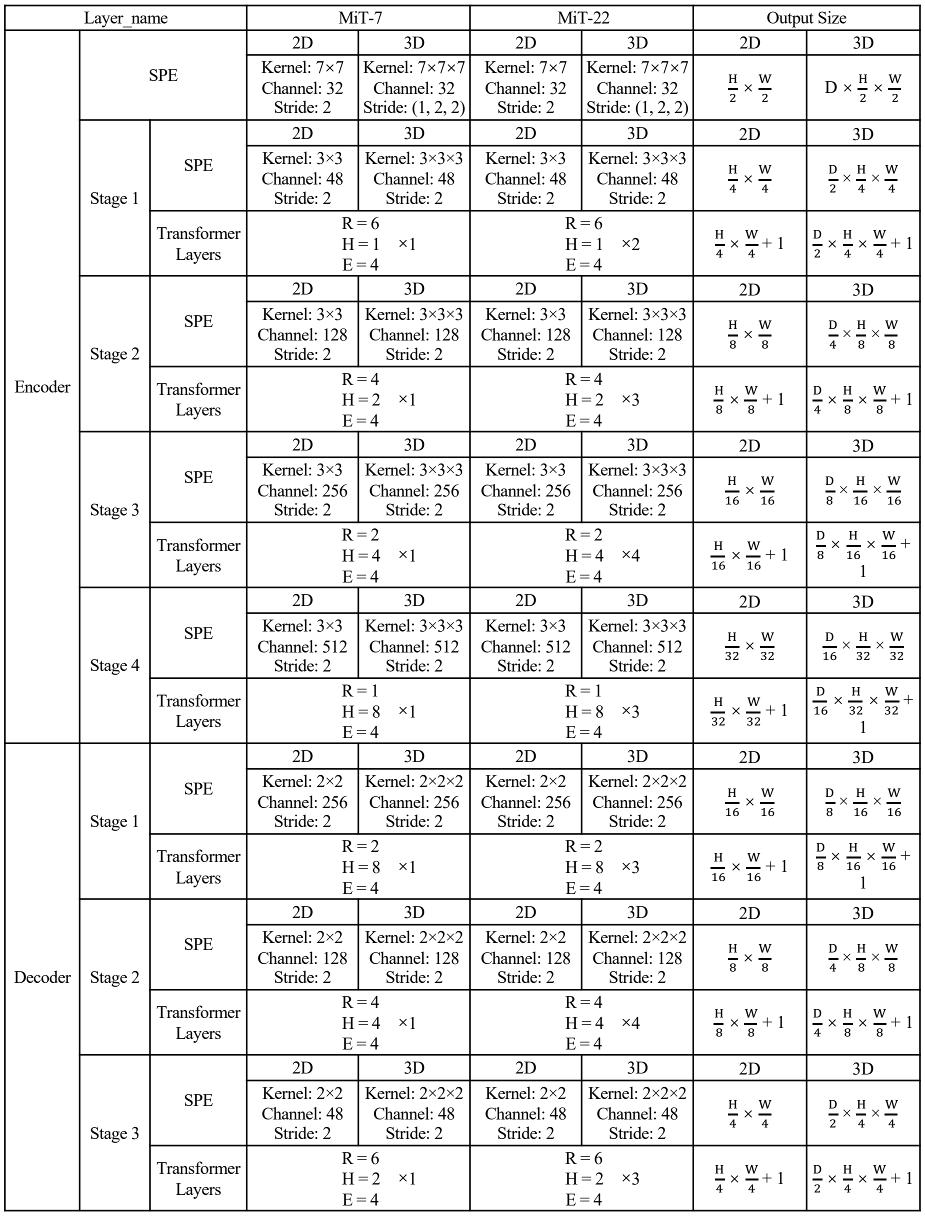
Figure 2 shows the architecture of CNN-based ResUnet, used by the compared PCRL [7]. It consists of a 2D/3D ResNet-50 [3] encoder, a decoder, and four skip connections between encoder and decoder. The decoder contains five up-sampling modules. Each of the first four modules has a transposed convolutional (TransConv) layer followed by a convolution block (ConvBlock) and a pixel-wise summation with the corresponding feature maps from the encoder and the TransConv layer. The last module comprises an Up-sampling layer followed by a 1×1 Conv layer that maps each 32-channel feature map to the desired number of classes.

4 Volume-slice consistency mechanism

Figure 3 gives an intuitive explanation of the proposed volume-slice consistency mechanism. Given a 3D volumetric image, we first create two augmented views via data augmentation, each of which has m 2D slices. We then compute the volumetric or slice representations of dual paths, *i.e.* $\boldsymbol{f}_1^{\text{Volume}}$, $\boldsymbol{f}_2^{\text{Volume}}$, $\boldsymbol{f}_1^{\text{Slices}}$, and $\boldsymbol{f}_2^{\text{Slices}}$. Here the slice representations $\boldsymbol{f}_1^{\text{Slices}}$ and $\boldsymbol{f}_2^{\text{Slices}}$ are generated by averaging the outputs of m slices. The loss function is composed of four items, including $\mathcal{L}^{\text{Volume}}$, $\mathcal{L}^{\text{Slices}}$, $\mathcal{L}^{\text{Volume}\rightarrow\text{Slices}}$, and $\mathcal{L}^{\text{Slices}\rightarrow\text{Volume}}$. The first two items aim to achieve the consistency at the level of global volume and local slices, respectively. Besides, the consistency across both levels should also be satisfied, which is achieved by the latter two items. By jointly using these four loss items, our model is able to capture richer representations from 3D medical images.

References

1. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: ICCV (2021) 2
2. Chen*, X., Xie*, S., He, K.: An empirical study of training self-supervised vision transformers. In: ICCV (2021) 2
3. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016) 3, 5



Layer_name			MiT-7		MiT-22		Output Size	
Encoder	SPE		2D	3D	2D	3D	2D	3D
			Kernel: 7×7 Channel: 32 Stride: 2	Kernel: 7×7×7 Channel: 32 Stride: (1, 2, 2)	Kernel: 7×7 Channel: 32 Stride: 2	Kernel: 7×7×7 Channel: 32 Stride: (1, 2, 2)	$\frac{H}{2} \times \frac{W}{2}$	$D \times \frac{H}{2} \times \frac{W}{2}$
	Stage 1	SPE	2D	3D	2D	3D	2D	3D
			Kernel: 3×3 Channel: 48 Stride: 2	Kernel: 3×3×3 Channel: 48 Stride: 2	Kernel: 3×3 Channel: 48 Stride: 2	Kernel: 3×3×3 Channel: 48 Stride: 2	$\frac{H}{4} \times \frac{W}{4}$	$\frac{D}{2} \times \frac{H}{4} \times \frac{W}{4}$
		Transformer Layers	R = 6 H = 1 ×1 E = 4		R = 6 H = 1 ×2 E = 4		$\frac{H}{4} \times \frac{W}{4} + 1$	$\frac{D}{2} \times \frac{H}{4} \times \frac{W}{4} + 1$
	Stage 2	SPE	2D	3D	2D	3D	2D	3D
			Kernel: 3×3 Channel: 128 Stride: 2	Kernel: 3×3×3 Channel: 128 Stride: 2	Kernel: 3×3 Channel: 128 Stride: 2	Kernel: 3×3×3 Channel: 128 Stride: 2	$\frac{H}{8} \times \frac{W}{8}$	$\frac{D}{4} \times \frac{H}{8} \times \frac{W}{8}$
		Transformer Layers	R = 4 H = 2 ×1 E = 4		R = 4 H = 2 ×3 E = 4		$\frac{H}{8} \times \frac{W}{8} + 1$	$\frac{D}{4} \times \frac{H}{8} \times \frac{W}{8} + 1$
	Stage 3	SPE	2D	3D	2D	3D	2D	3D
			Kernel: 3×3 Channel: 256 Stride: 2	Kernel: 3×3×3 Channel: 256 Stride: 2	Kernel: 3×3 Channel: 256 Stride: 2	Kernel: 3×3×3 Channel: 256 Stride: 2	$\frac{H}{16} \times \frac{W}{16}$	$\frac{D}{8} \times \frac{H}{16} \times \frac{W}{16}$
		Transformer Layers	R = 2 H = 4 ×1 E = 4		R = 2 H = 4 ×4 E = 4		$\frac{H}{16} \times \frac{W}{16} + 1$	$\frac{D}{8} \times \frac{H}{16} \times \frac{W}{16} + 1$
	Stage 4	SPE	2D	3D	2D	3D	2D	3D
			Kernel: 3×3 Channel: 512 Stride: 2	Kernel: 3×3×3 Channel: 512 Stride: 2	Kernel: 3×3 Channel: 512 Stride: 2	Kernel: 3×3×3 Channel: 512 Stride: 2	$\frac{H}{32} \times \frac{W}{32}$	$\frac{D}{16} \times \frac{H}{32} \times \frac{W}{32}$
		Transformer Layers	R = 1 H = 8 ×1 E = 4		R = 1 H = 8 ×3 E = 4		$\frac{H}{32} \times \frac{W}{32} + 1$	$\frac{D}{16} \times \frac{H}{32} \times \frac{W}{32} + 1$
Decoder	Stage 1	SPE	2D	3D	2D	3D	2D	3D
			Kernel: 2×2 Channel: 256 Stride: 2	Kernel: 2×2×2 Channel: 256 Stride: 2	Kernel: 2×2 Channel: 256 Stride: 2	Kernel: 2×2×2 Channel: 256 Stride: 2	$\frac{H}{16} \times \frac{W}{16}$	$\frac{D}{8} \times \frac{H}{16} \times \frac{W}{16}$
		Transformer Layers	R = 2 H = 8 ×1 E = 4		R = 2 H = 8 ×3 E = 4		$\frac{H}{16} \times \frac{W}{16} + 1$	$\frac{D}{8} \times \frac{H}{16} \times \frac{W}{16} + 1$
	Stage 2	SPE	2D	3D	2D	3D	2D	3D
			Kernel: 2×2 Channel: 128 Stride: 2	Kernel: 2×2×2 Channel: 128 Stride: 2	Kernel: 2×2 Channel: 128 Stride: 2	Kernel: 2×2×2 Channel: 128 Stride: 2	$\frac{H}{8} \times \frac{W}{8}$	$\frac{D}{4} \times \frac{H}{8} \times \frac{W}{8}$
		Transformer Layers	R = 4 H = 4 ×1 E = 4		R = 4 H = 4 ×4 E = 4		$\frac{H}{8} \times \frac{W}{8} + 1$	$\frac{D}{4} \times \frac{H}{8} \times \frac{W}{8} + 1$
	Stage 3	SPE	2D	3D	2D	3D	2D	3D
			Kernel: 2×2 Channel: 48 Stride: 2	Kernel: 2×2×2 Channel: 48 Stride: 2	Kernel: 2×2 Channel: 48 Stride: 2	Kernel: 2×2×2 Channel: 48 Stride: 2	$\frac{H}{4} \times \frac{W}{4}$	$\frac{D}{2} \times \frac{H}{4} \times \frac{W}{4}$
		Transformer Layers	R = 6 H = 2 ×1 E = 4		R = 6 H = 2 ×3 E = 4		$\frac{H}{4} \times \frac{W}{4} + 1$	$\frac{D}{2} \times \frac{H}{4} \times \frac{W}{4} + 1$

Fig. 1. Detailed settings of MiT network. Here, 'R': reduction ratio of SRA; 'H': head number of SRA; and 'E': expansion ratio of FFN

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

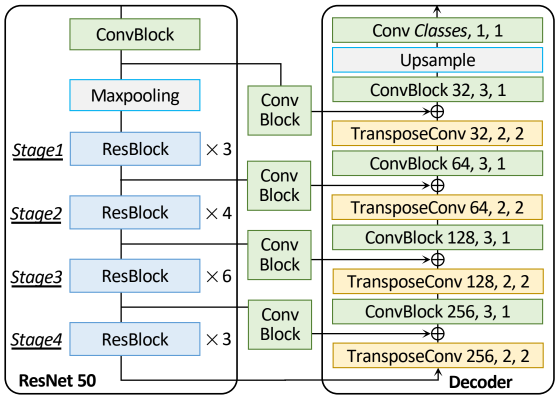


Fig. 2. Detailed architecture of ResUnet: A 2D/3D ResNet-50 [3] encoder, a decoder, and four skip connections between encoder and decoder. Green 'ConvBlock': 2D Conv-Batch Normalization(BN)-ReLU or 3D Conv-IN-LeakyReLU; Yellow 'TransConv': 2D/3D transposed convolutional layer. Note that the numbers in each block / layer indicate the number of filters, kernel size, and stride, respectively.

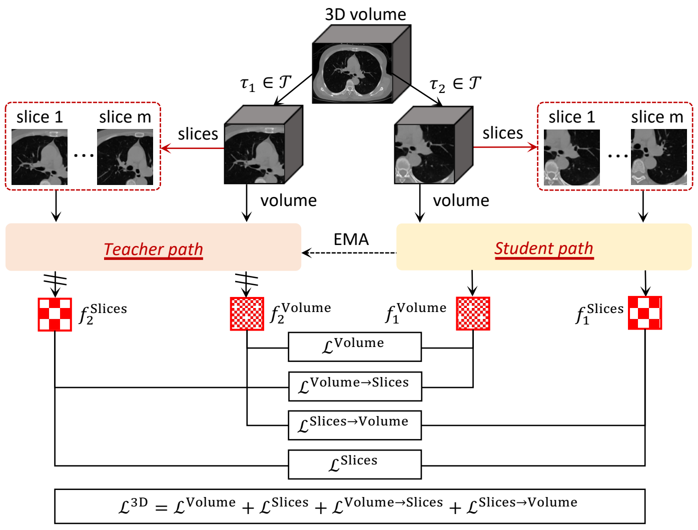

3D volume
$\tau_1 \in \mathcal{T}$
$\tau_2 \in \mathcal{T}$
slice 1
slice m
slices
volume
slices
slice 1
slice m
volume
Teacher path
EMA
Student path
f_2^{Slices}
f_2^{Volume}
f_1^{Volume}
f_1^{Slices}
$\mathcal{L}^{\text{Volume}}$
$\mathcal{L}^{\text{Volume}\to\text{Slices}}$
$\mathcal{L}^{\text{Slices}\to\text{Volume}}$
$\mathcal{L}^{\text{Slices}}$
$\mathcal{L}^{3D} = \mathcal{L}^{\text{Volume}} + \mathcal{L}^{\text{Slices}} + \mathcal{L}^{\text{Volume}\to\text{Slices}} + \mathcal{L}^{\text{Slices}\to\text{Volume}}$

Fig. 3. Intuitive explanation of volume-slice consistency mechanism.

5. Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV (2021) 2, 3
6. Xie, Y., Zhang, J., Xia, Y., Shen, C.: A mutual bootstrapping model for automated skin lesion segmentation and classification. IEEE Transactions on Medical Imaging **39**(7), 2482–2493 (2020) 2
7. Zhou, H.Y., Lu, C., Yang, S., Han, X., Yu, Y.: Preservational learning improves self-supervised medical image models by reconstructing diverse contexts. In: ICCV.

pp. 3499–3509 (2021) 3