# SHF: Symmetrical Hierarchical Forest with Pretrained Vision Transformer Encoder for High-Resolution Medical Segmentation

**Enzhi Zhang**[1,2] **Peng Chen**[2] * **Rui Zhong**[1] **Du Wu**[2] **Isaac Lyngaas**[3]
**Jun Igarashi**[2] **Xiao Wang**[3] **Masaharu Munetomo**[1] **Mohamed Wahib**[2]

[1]Hokkaido University, Sapporo, Japan
[2]RIKEN Center for Computational Science, Hyogo, Japan
[3]Oak Ridge National Laboratory, Tennessee, USA
{zhangenzhi, zhongrui, munetomo}@iic.hokudai.ac.jp
{peng.chen, du.wu, jigarashi, mohamed.attia}@riken.jp
{lyngaasir, wangx2}@ornl.gov

## Abstract

This paper presents a novel approach to addressing the long-sequence problem in high-resolution medical images for Vision Transformers (ViTs). Using smaller patches as tokens can enhance ViT performance, but quadratically increases computation and memory requirements. Therefore, the common practice for applying ViTs to high-resolution images is either to: (a) employ complex sub-quadratic attention schemes or (b) use large to medium-sized patches and rely on additional mechanisms within the model to capture the spatial hierarchy of details. We propose Symmetrical Hierarchical Forest (SHF), a lightweight approach that adaptively patches the input image to increase token information density and encode hierarchical spatial structures into the input embedding. We then apply a reverse depatching scheme to the output embeddings of the transformer encoder, eliminating the need for convolution-based decoders. Unlike previous methods that modify attention mechanisms or use a complex hierarchy of interacting models, SHF can be retrofitted to any ViT model to allow it to learn the hierarchical structure of details in high-resolution images without requiring architectural changes. Experimental results demonstrate significant gains in computational efficiency and performance: on the PAIP WSI dataset, we achieved a $3\sim32\times$ speedup or a $2.95\%\sim7.03\%$ increase in accuracy (measured by Dice score) at a $64K^2$ resolution with the same computational budget, compared to state-of-the-art production models. On the 3D medical datasets BTCV and KiTS, training was $6\times$ faster, with accuracy gains of $6.93\%$ and $5.9\%$, respectively, compared to models without SHF.

## 1 Introduction

Recently, Transformers have been rapidly adopted in the field of computer vision [1, 2]. Building on the self-attention mechanism, Vision Transformers (ViTs) and their variants have achieved significant advancements in various image classification and downstream visual tasks [3–7]. Text tokens are atomic, semantically distinct, and rich in information, whereas visual tokens are geometrically related and sparse in semantics. In other words, feeding a sequence of image patches to a transformer encoder deprives the self-attention mechanism of direct information about spatial hierarchy.

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

The loss of spatial hierarchy information becomes more pronounced when working with high-resolution or multi-dimensional medical images, as the spatial hierarchy becomes more detailed and intricate [8]. This requires the use of very small patches so that the self-attention mechanism can capture local features [9]. However, using smaller patches quadratically increases the computational and memory costs of self-attention, prompting several approaches to address this issue by modifying the model architecture to help it learn the hierarchical structure in images. Most of these approaches fall into two broad categories: *model-hierarchical* and *attention-hierarchical*. Model-hierarchical solutions involve training ViTs hierarchically, with multiple transformers operating at different resolution levels [10, 8, 11, 12]. While this approach can improve model performance, it also increases training time and memory usage. Additionally, managing multiple interacting transformers adds complexity, requiring extensive hyperparameter tuning at each resolution level. Attention-hierarchical solutions alter the patching scheme at the self-attention stage to represent hierarchical features, as seen in Swin [13, 14], MViT [15], and MViTv2 [16]. Although these methods are more parameter-efficient than standard ViTs [4], they introduce additional spatial operations, increasing model complexity and reducing multi-modal capabilities.

From the above summary, two questions arise: **(1)** How can we design a patching strategy that uses the fewest possible patches to represent the original image, thus increasing the information density of each patch to maintain model performance, without altering the structure of the ViT model? **(2)** If an effective patching strategy could be developed to relay spatial hierarchical information to the model, could we leverage a post-encoder adaptive dispatching strategy during inference to eliminate the need for post-encoder convolution decoding?

To address the two questions above, this paper proposes Symmetrical Hierarchical Forest (`SHF`), which adapts the patching strategy to the hierarchical details of each training example. This approach enables the ViT to capture hierarchical local spatial information typically derived either from post-encoder convolutions (e.g. the convolution decoder used in the SAM 2 model [17]) or hierarchical architecture features (i.e. Swin [14] and HIPT [8]). We downscale patches in regions with fewer details, aligning them to the patch size used for regions with more details. However, without expert knowledge of hyperparameters, we rely solely on hierarchical forests to extract/represent the spatial hierarchy information. Using the `SHF` scheme, we demonstrate that the model receives sufficient information about the spatial hierarchy, allowing us to eliminate additional model components (such as U-Net [18] or convolution decoding blocks) that would otherwise be required. Notably, the use of additional components (e.g. the convolution decoder in SAM 2 [17]) can lead to high memory demands to store activations for high-resolution masks (e.g. over 20GB of memory for storing mask activations when using SAM 2 on $64K^2$ images). By employing a transformer encoder-only design with hierarchical forest and post-depatching, we can use smaller patch sizes, enabling self-attention to capture the spatial hierarchy more effectively than larger patches and additional mechanisms (such as U-Net or convolution decoding blocks). As an added benefit, our method simplifies model design and allows for swapping in different encoders, as it is a data-based approach that operates on the input and output of the transformer encoder without modifying the encoder itself.

The contributions of this work are as follows:

- **Symmetrical Hierarchical Forest**. By applying the Symmetrical Hierarchical Forest (`SHF`), we can extract the hierarchical information directly and eliminate the expert knowledge required for tuning hyperparameters. Additionally, we completely discard the convolution decoder, significantly reducing the computational and memory overhead ($\sim$75% GPUs) of mask processing in high-resolution segmentation tasks. Finally, by downscaling redundant regions in the input image space, we achieve a quadratic reduction in the computational cost of the ViT encoder.
- **Long Context Segmentation**. To demonstrate the efficiency of `SHF`, we conducted experiments on high-resolution medical imaging datasets, retrofitting state-of-the-art (SoTA) models such as SAM 1 [19] and SAM 2 [17] with our `SHF` scheme in place of their convolution decoders. When comparing models retrofitted with `SHF` to SoTA models without `SHF`, high-resolution pathology datasets (e.g. the PAIP dataset, ranging from $512^2$ to $64K^2$ pixels) and 3D MRI datasets (BTCV, KiTS) benefit from the efficiency of `SHF`, allowing patch sizes as small as $2\times2$ pixels and $2\times2\times2$ voxels. At the same performance level, we achieve a $3\times$ to $32\times$ speedup on the PAIP dataset, or, with the same computational budget, a 7.03% increase in Dice score at $64K^2$ resolution. On 3D medical imaging datasets, such as BTCV and KiTS, we see a $\sim6\times$ training speed improvement along with performance increases of 6.93% and 5.9%, respectively, compared to SoTA models without `SHF`.

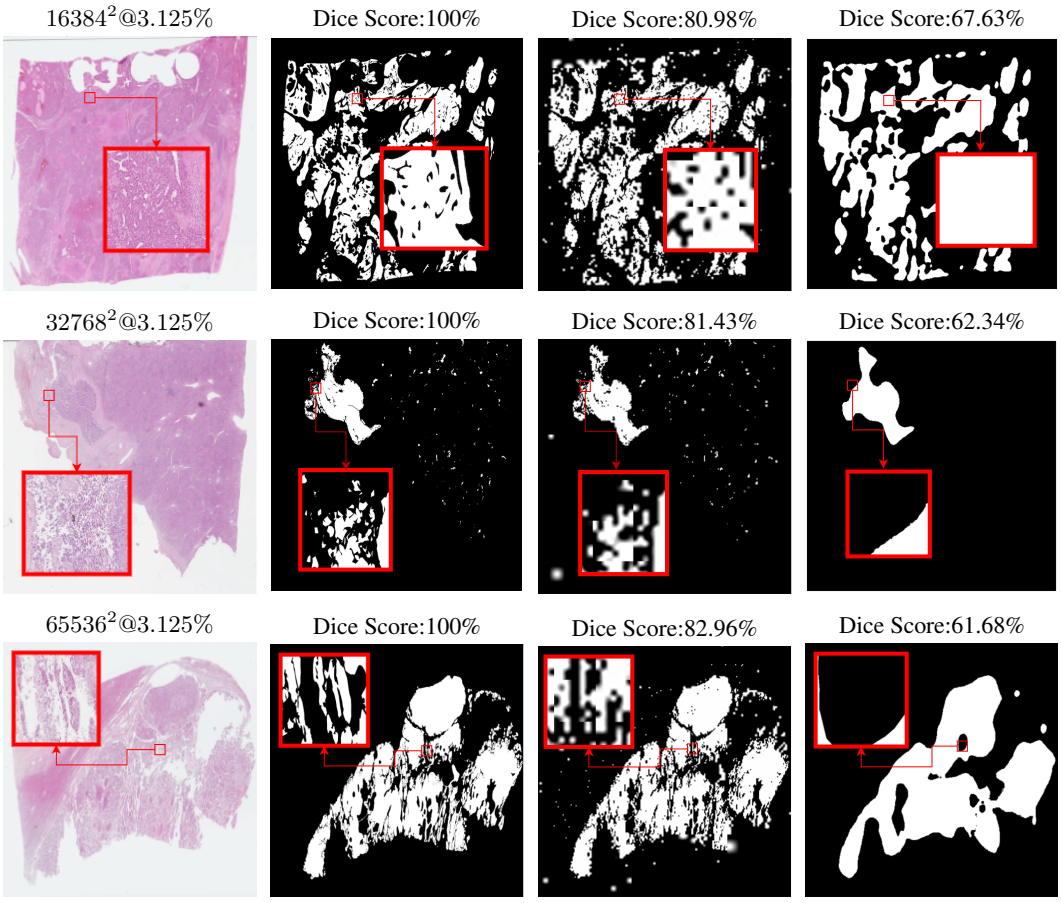



Figure 1: We compared the segmentation difference between SAM [19] and SAM retrofitted with our SHF scheme (SHF-SAM) instead of the original convolution decoder for PAIP [20] at $16,384^2$, $32,768^2$, and $65,536^2$ resolutions. At the same GPU budgets, SHF-SAM can go down to batch size of 8x8 (vs. $1,024$x$1,024$ at best for SAM before going OOM). As a result, SHF-SAM can extract and express mask details better than SAM, with the gap in accuracy favoring SHF-SAM as the resolution gets higher.

- **Simplicity and Low-Overhead**. Unlike existing methods that modify the self-attention or transformer encoder mechanisms, our solution preserves the original self-attention mechanism. This ensures seamless retrofitting of SHF into any vision transformer. SHF is a low-overhead pre-processing and post-processing solution that is further amortized over epochs, making the overhead effectively negligible.

The rest of this paper is organized as follows: Section 2 reviews related work. Section 3 presents the methodology. Section 4 describes the experimental setup. Section 5 reports the evaluation results. Finally, Section 6 concludes the paper and outlines future work.

## 2 Related Work

The computational cost of self-attention increases as the patch size decreases. To mitigate this, several strategies have been developed. Sequence parallel methods, including Deep-Speed Ulysses [21], LightSeq [22], RingAttention [23], LLS [24], FlashAttention [25], and [26]. Linear approximation methods, such as spectral attention[27–29], low-rank approximation [30, 31], sparse attention matrix sampling [32–36], infrequent self-attention updates [37, 38], or combinations of these [39]. These methods reduce the computational load of the attention mechanism; however, excessive reduction

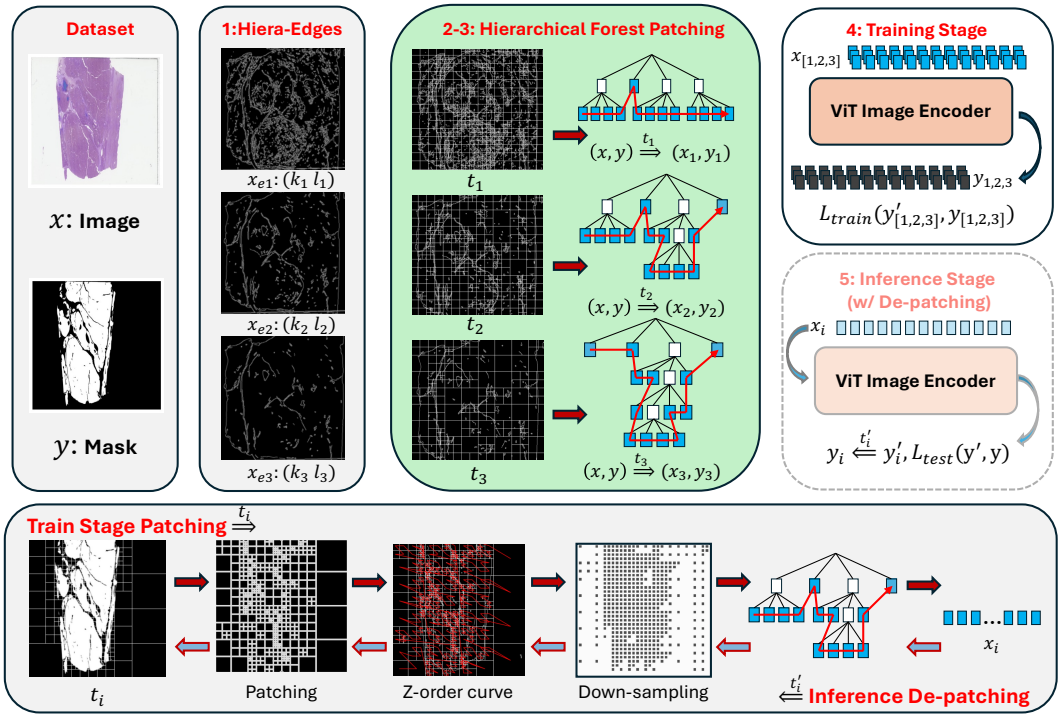



Figure 2: Overview of SHF. SHF begins with the original image and ends with feeding the extracted patches (tokens) into an intact transformer-based model. In a real training example on the SAM [19] model using 512×512 images from the PAIP [20] liver cancer dataset, SHF reduces the number of patches from 4,096 to 512 (each of size 4×4) while maintaining the same Dice score. This results in an ∼8× reduction in sequence length and a ∼7.53× speedup in end-to-end training.

can lead to performance loss, as reported in the literature [40]. Hierarchical training of ViTs, where multiple transformers are trained at different resolution levels [10, 8, 11, 12]. However, using multiple transformers increases training time and memory usage, and managing multiple interacting transformers is complex. Recently, quadtrees have been used in image segmentation to reduce attention cost, e.g. quadtree/octree attention or patch pre-processing [41–43]. Both of those approaches employ quadtrees, but involve additional model complexity or need expert knowledge in the processing stage for hyperparameters.

## 3 Methodology

### 3.1 Vision Transformers and Attention

The self-attention mechanism in transformers computes attention scores $A$ between input tokens, forming the attention matrix. Let $x \in R^{N\times F}$ denote a sequence of $N$ feature vectors of dimensions $F$. A transformer is a function $\mathcal{T} : R^{N\times F} \rightarrow R^{N\times F}$ defined by the composition of $L$ transformer layers $\mathcal{T}_1(\cdot), ..., \mathcal{T}_L(\cdot)$ as follows:

$$Tr_l(x) = f_l(A_l(x) + x) \tag{1}$$

$A_l(\cdot)$ is the self-attention function. The function $f_l(\cdot)$ transforms each feature independently of the others and is usually implemented with a small two-layer feedforward network. Formally, the input sequence $x$ is projected by three matrices $W_Q \in R^{F\times D}, W_K \in R^{F\times D}$, and $W_V \in R^{F\times D}$, to corresponding representations $Q$, $K$ and $V$. Thus, the attention scores are calculated as follows:

$$Q = xW_Q, K = xW_K, V = xW_V, A_{ij} = \text{Softmax}\left((Q_i K_j^T)/\sqrt{d_k}\right) \tag{2}$$

where $Q_i$ and $K_j$ are query and key vectors for tokens $i$ and $j$, and $d_k$ is the dimension of the key vectors. The complexity of the attention matrix is $O(N^2)$, where $N$ is the sequence length. We further assume that the input is the content of a square image $x$ with a resolution of $Z$, that is, let

Table 1: Speedup of SHF end-to-end training for PAIP dataset at the same segmentation quality as the baseline. We use the highest dice score of the baseline models, SAM and SHF-SAM.

| Resolution | Model-Patch | Seconds/Image | GPUs | Sequence Length | Dice Score (%) | Speedup (×) |
|---|---|---|---|---|---|---|
| $512 \times 512$ | **SHF-SAM-16** | 0.03616 | 1 | 256 | 71.03 | 7.75× |
| | SAM-4 | 0.28041 | 1 | 16,384 | 68.98 | |
| $1,024 \times 1,024$ | **SHF-SAM-16** | 0.09913 | 1 | 1,024 | 73.65 | 3.86× |
| | SAM-8 | 0.38261 | 8 | 16,384 | 66.56 | |
| $4,096 \times 4,096$ | **SHF-SAM-16** | 0.1031 | 8 | 1,024 | 74.17 | 15.7× |
| | SAM-32 | 1.6183 | 64 | 16,384 | 71.05 | |
| $8,192 \times 8,192$ | **SHF-SAM-16** | 0.3512 | 16 | 2,048 | 76.21 | 7.17× |
| | SAM-64 | 2.5168 | 128 | 16,384 | 67.31 | |
| $16,384 \times 16,384$ | **SHF-SAM-16** | 0.3672 | 32 | 2,048 | 76.89 | 15.4× |
| | SAM-128 | 5.6714 | 256 | 16,384 | 67.63 | |
| $32,768 \times 32,768$ | **SHF-SAM-32** | 0.3826 | 64 | 2,048 | 76.08 | 23.8× |
| | SAM-256 | 9.1213 | 512 | 16,384 | 62.34 | |
| $65,536 \times 65,536$ | **SHF-SAM-64** | 0.4013 | 256 | 2,048 | 75.33 | 32.3× |
| | SAM-1024 | 12.9833 | 1,024 | 16,384 | 61.68 | |

$x \in R^{Z \times Z}$, and by assuming that patches arise from the uniform grid patch method of patch size $p$. Thus the sequence $N = (Z/P)^2$. The total computation and memory cost of attention scores defined in Eqn 2 according to resolution and patch size is $O([Z/P]^4)$. This complexity shows the difficulties of increasing the resolution while decreasing the patch size $P$ with the uniform grid patch strategy.

### 3.2 Symmetrical Hierarchical Forest (SHF)

In the following paragraphs, we describe how SHF works, following the steps outlined in Fig. 2:

**Hiera-Edges Detection**. We aim to use different methods to extract hierarchical details, as this would allow us to augment the dataset. This, however, is different from traditional augmentation applied at the image level; we augment by providing the model with different views of the spatial hierarchy for each image, thereby giving the model a better opportunity to learn the hierarchy. To use different ways to extract the hierarchical details in the image $x \in X$, as Fig. 2-1, we use a Gaussian blur with kernel $k$ and Canny [44] edge detection with threshold $l$ to the original input images $x \in X$. The Gaussian blur with kernel $k$ smooths the irrelevant details, and the Canny edge detection with threshold $l$ extracts the grayscale edges $x_e$ of the image. To generate different spatial structures that can also recover the original inputs, during our experiments, we randomly choose the threshold to be in the range [100,200], and the kernel size is randomly set to one of [3,5,7,9,11,13].

**Hierarchical Forest**. In the Hierarchical Forest stage (Fig. 2-2), we build several quadtrees (octrees in 3D) from each $x_e$; $x_e$ undergoes a recursive tree partitioning. To construct the tree $T$, we create tree nodes $T_n$ representing specific regions where $n$ is the number of leaf nodes. The leaf nodes $T_{n+s-1}$ is defined recursively as follows:

$$T_{n+s-1} = \begin{cases} T_n, & \text{if } n \geq N \\ T_n[i] = \{T_n^1, T_n^2, ..., T_n^s\}, & \arg\max_i V(T_n[i]) \end{cases} \tag{3}$$

where $V$ is the criterion we use to differentiate the different levels of detail. We choose the maximum sum of pixel values among the tree nodes by $i = \arg\max_i V(T^n[i])$, $\{T_n^1, T_n^2, ..., T_n^s\}$ are the $s$ new child nodes after the subdivision of $T^n[i]$, $s = 2^d$ is the number of subdivisions and $d$ is the number of dimensions. $T$ can be used for both 2D and 3D tasks, for example $s = 4, 8$ which means quadtree and octree, respectively [45, 46]. In our implementation, to avoid unnecessary padding or dropping due to varying sample lengths, we control the number of splits at the leaf nodes to keep each sample at the same sequence length $N$. This approach fully utilizes the GPU resources without discarding sample information. The sequence length $N$ is set to $[1024, 4096, 8192, 16384, 16384, 16384]$ with respect to resolutions, which practically allows the input $x_e$ to be subdivided all the way down to the $2 \times 2$ patch size level.

After building the tree, as a property of quad/octrees, visiting all the tokens (appearing as leaves in the tree) from left to right gives the token sequence as a Z-order space-filling curve in the image space [47]. This operation($\overset{t_i}{\Longrightarrow}$) results in a sequence of image patches shown in step Fig. 2-3. We

not only use spatial trees to encode images $x$, but we also encode masks $y$. Using the same tree and z-order of the mask will also result in a sequence of mask patches $y_i$. Since the spatial trees are built from the image, the sequence length of the image and mask patches $(x_i, y_i)$ should be the same $N$.

**Image Encoder**. After completing hierarchical forest patching, we obtain an image patch sequence and a mask patch sequence of the same length, allowing us to (continuously) train the ViT model. We use the well-known pre-trained image encoder from SAM in steps Fig. 2-4 and then continue training the model on our dataset after retrofitting our `SHF` scheme into the SAM model. SAM uses an MAE [2] pre-trained ViT [4], minimally adapted to process high-resolution inputs. For 3D MRI data tasks, we use the SAM 2 [17] encoder and similarly retrofit `SHF` into SAM 2. When retrofitting `SHF` into both SAM and SAM 2, we remove the convolution decoder that is part of the original SAM and SAM 2 models. It should be noted that during the training phase of the model, we do not directly generate masks $y^{'}$ but instead generate encoded masks $y_i^{'}$. The advantage of this approach is that it saves memory, which would otherwise be needed for high-resolution masks, and reduces the backpropagation compute costs associated with the heavy decoders. We summarized the objective function as follows:

$$\min_{\theta} \mathbb{E}_{\substack{(x,y)\sim\mathcal{D} \\ k\sim\mathcal{K}, l\sim\mathcal{L}}} \left[ \sum_{i=1}^{M} \mathcal{L}_{\text{hiera}} \Big( f_\theta \big( T_{\text{tree}}(x,k,l)_i \big),\ T_{\text{sym}}(y,k,l)_i \Big) \right] \tag{4}$$

Where $M$ is total number of samples, $\theta$ represents the trainable weights, $\mathcal{L}_{\text{hiera}}$ denotes the Dice loss, $(k, l)$ specify the Gaussian filter size and Canny threshold, and $T_{tree} = T_{sym}$ are the recursive tree operators applied to both the input $x$ and the mask $y$.

**Symmetrical Depatching**. The function of depatching ($\overset{t_i^{'}}{\Longleftarrow}$) is to upscale the sequence obtained from the ViT into a mask during the inference stage. By depatching, we can eliminate the need for general U-Net or lightweight decoders and calculate the backpropagation gradient directly at the output of the encoder (in Fig. 2-5). This is because we have observed that the information in the mask itself is sparser compared to the input image (see the Fig. 4). Therefore, when reconstructing the mask at the evaluation stage, we simply use the same quad/octree structure derived from the image to linearly upscale the patches at the corresponding positions in the sequence.

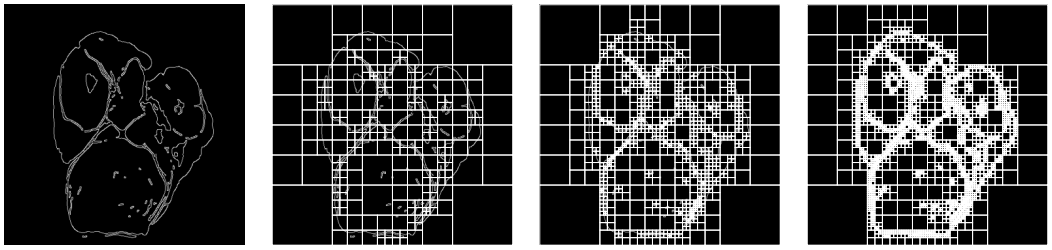
## 4 Experimental Setup

### 4.1 High-Resolution Medical Image Datasets: PAIP, BTCV, & KiTS

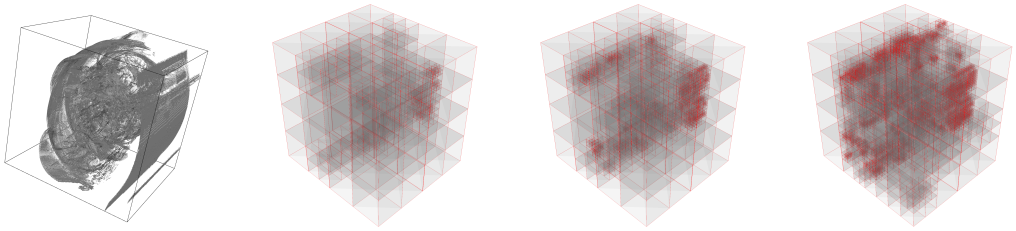
**PAIP:** [20] is a high-resolution, real-world liver cancer pathology dataset, with sample resolutions up to $64K^2$, significantly surpassing those of conventional image datasets. PAIP contains $2,457$ Whole-Slide Images (WSIs). When lower resolutions are needed, we downscale the images to uniform sizes of $[512, 1024, 4096, 8192, 16384, 32768, 65536]$ square pixels. For training, we randomly select $70\%$ of samples, $10\%$ for validation, and $20\%$ for testing. All datasets are shuffled and normalized to $[0.0, 1.0]$ as input for the model. The **BTCV:** challenge [48] for 3D multi-organ segmentation includes 30 subjects with abdominal Computed Tomography (CT) [49] scans, with 13 organs annotated by experts. Each CT scan consists of 80 to 225 slices, each with $512^2$ pixels. The multi-organ segmentation task is defined as a 13-class segmentation problem, where the average dice score across all classes is typically reported. Although BTCV has a lower resolution compared to the PAIP dataset ($512^2$ vs. $64K^2$), it remains a widely used benchmark in the high-resolution medical segmentation community. **KiTS:** the 2019 Kidney and Kidney Tumor Segmentation Challenge (KiTS19 [50]) aimed to develop algorithms for segmenting kidneys and kidney tumors in contrast-enhanced 3D CT scans. The dataset consists of 300 anonymized scans in NIfTI (.nii.gz) format, each manually annotated with kidney and tumor labels. Each scan has a typical resolution of 512×512 pixels per slice, with the number of slices varying based on patient anatomy. KiTS is a common benchmark for 3D MRI segmentation [51].

### 4.2 Evaluating Models: Baselines & Proposed

**Baseline Model:** We use the well-known segmentation model SAM [19], which employs a Vision Transformer (ViT) encoder as its backbone for segmentation tasks. SAM offers ViT variants (ViT-Base (b), ViT-Large (l), and ViT-Huge (h)), each with 12, 24, or 32 transformer layers, respectively.



(a) Edge Image (b) $L$=256, $\gamma$=4.07 (c) $L$=1024,$\gamma$=12.18 (d) $L$=2050,$\gamma$=27.66



(e) Edge MRI (f) $L$=260, $\gamma$=6.06 (g) $L$=1030, $\gamma$=22.47 (h) $L$=2052, $\gamma$=47.76

Figure 3: Spatial subdivision of an edge image with a resolution of 1024×1024. $L$ represents the sequence length, and $\gamma$ denotes the compression ratio relative to the grid patches. SHF significantly compresses the image, with higher dimensions resulting in a higher compression ratio.

These weight configurations -b, l, and h- are pre-trained using Masked Autoencoders (MAE) [2]. Unlike Unet-style models [52, 18, 53, 13], the SAM mask decoder is lightweight, containing only two convolutional layers. However, reconstructing high-resolution masks from the latent space requires additional upscaling layers, leading to increased memory usage.

**Proposed Model:** As defined in Eqn 4, our approach utilizes the tree structure from the patching stage in SHF, reducing the need for decoder training. We also experimented with the updated SAM2 [17] encoder ViT model, adapting it for 3D MRI data tasks, such as BTCV and KiTS19. Unlike SAM2 original video frame processing, we use 3D convolutional layers for voxel processing in the patch embedding stage. Unet-shaped models, such as UNETR [18], TransUnet [53], and SWIN Unet [13], follow a contraction-expansion pattern with transformer or convolutional layers as encoders to extract image details, connected to decoders via skip connections. These decoders upscale the representation vectors from the dense latent space to match the mask's size. For comparison with SHF, we test Unet [52] with pre-trained weights from the timm package, along with the SAM-pretrained UNETR [18] and TransUnet [53] models.

**Performance Metrics:** Computational performance is reported in seconds per image for end-to-end training. Segmentation accuracy is evaluated using the Dice score, which quantifies the overlap between predicted and ground truth segmentation masks. It is defined as:

$$\text{Dice(X,Y)} = 2 \cdot |X \cap Y| / (|X| + |Y|) \tag{5}$$

where $X$ and $Y$ are the sets being compared, and $|X \cap Y|$ is the size of their intersection.

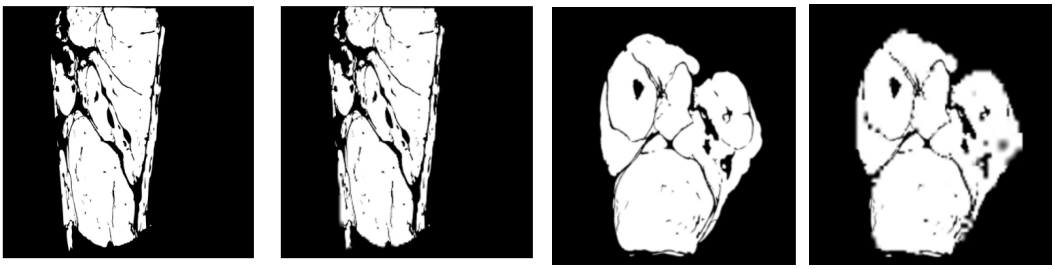
## 5 Evaluation

### 5.1 Speedup: SHF vs w/o SHF on the same models, datasets, and comparable performance

Training with SHF is faster due to its ability to significantly compress the sequence length , shown in Figure. 3, and eliminate the decoder component. As shown in Table 1, SHF, which combines pre-processing and post-processing steps on top of the pre-trained SAM baseline, achieves a geometric mean speedup ranging from 3.86× to 32.3× while maintaining comparable dice scores. This speedup is measured when both SHF and the baseline are trained for the same number of epochs. At the highest resolution of $64K^2$ with training on 256/1024 GPUs, SHF delivers approximately a 32× speedup compared to naive SAM encode and a reduction of 75% usage of GPUs. Such a speedup

Table 2: Improvement in quality of segmentation (IQS) for the PAIP dataset against different baselines.

| Resolution | Model | Patch | Seconds/Image/GPU | GPUs | Sequence Length | Dice Score (%) |
|---|---|---|---|---|---|---|
| $1,024 \times 1,024$ (IQS: **+2.95%**) | SAM | $8^2$ | 0.3826 | 8 | 16,384 | 66.56 |
| | UNETR | $8^2$ | 1.0863 | 32 | 16,384 | 75.72 |
| | TransUNet | - | 1.3247 | 8 | - | 72.38 |
| | UNet | - | 0.0981 | 1 | - | 68.92 |
| | **SHF+SAM** | $2^2$ | 0.0991 | 1 | 1,024 | **78.67** |
| $4,096 \times 4,096$ (IQS: **+3.22%**) | SAM | $32^2$ | 1.6183 | 64 | 16,384 | 71.05 |
| | UNETR | $32^2$ | 1.8613 | 128 | 16,384 | 75.77 |
| | TransUNet | - | 2.1637 | 128 | - | 71.32 |
| | UNet | - | 0.3712 | 16 | - | 64.11 |
| | **SHF+SAM** | $2^2$ | 0.3766 | 8 | 4,096 | **78.99** |
| $8,192 \times 8,192$ (IQS: **+4.41%**) | SAM | $64^2$ | 2.5168 | 128 | 16,384 | 67.31 |
| | UNETR | $64^2$ | 2.6618 | 256 | 16,384 | 75.27 |
| | TransUNet | - | 2.3678 | 256 | - | 70.89 |
| | UNet | - | 1.2858 | 32 | - | 63.21 |
| | **SHF+SAM** | $2^2$ | 1.5327 | 16 | 8,192 | **79.68** |
| $16,384 \times 16,384$ (IQS: **+5.09%**) | SAM | $128^2$ | 5.6714 | 256 | 16,384 | 67.63 |
| | UNETR | $128^2$ | 5.1179 | 512 | 16,384 | 75.89 |
| | TransUNet | - | 6.1296 | 512 | - | 70.46 |
| | UNet | - | 2.7825 | 256 | - | 62.97 |
| | **SHF+SAM** | $2^2$ | 3.2741 | 32 | 16,384 | **80.98** |
| $32,768 \times 32,768$ (IQS: **+6.47%**) | SAM | $256^2$ | 9.1213 | 512 | 16,384 | 62.34 |
| | UNETR | $256^2$ | 8.1896 | 1,024 | 16,384 | 74.96 |
| | TransUNet | - | 10.001 | 1,024 | - | 69.88 |
| | UNet | - | 4.2714 | 512 | - | 61.38 |
| | **SHF+SAM** | $4^2$ | 3.4631 | 64 | 16,384 | **81.43** |
| $65,536 \times 65,536$ (IQS: **+7.03%**) | SAM | $1,024^2$ | 12.983 | 1,024 | 16,384 | 61.68 |
| | UNETR | $1,024^2$ | 13.218 | 2,048 | 16,384 | 75.31 |
| | TransUNet | - | 14.352 | 2,048 | - | 67.67 |
| | UNet | - | 5.9611 | 1,024 | - | 59.69 |
| | **SHF+SAM** | $8^2$ | 3.6112 | 256 | 16,384 | **82.96** |



(a) Mask (b) Recon-16,384 (c) Mask (d) Recon-1,024

Figure 4: Reconstructed mask through the SHF build from the input $64K^2$ image with sequence length $16,384$ and $1,024$ with recovery accuracy $99.5\%$ and $93.7\%$. These high recovery accuracies suggest the information contained in the masks are sparse.

benefited from eliminating the heavy decoder, allowing more parallelization and reducing inference and back-propagation.

### 5.2 Segmentation Performance: SHF vs w/o SHF on the different models and datasets

**Qualitative Results:** Table 2 demonstrates the segmentation improvements across different models and PAIP resolutions. At comparable resolutions, SHF achieves a nearly $8\times$ reduction in patch size while maintaining the same computational complexity. This results in an average $5.5\%$ improvement in dice score over the original model. Additionally, these improvements come with training time speedups of up to $4.6\times$. At high resolution ($64K^2$), SHF lacks a decoder during training, offering substantial computational and GPU memory savings compared to SAM, which relies on an upsampling restoration mask. Consequently, SAM requires a shorter sequence length at high resolutions, as it performs more upsampling to generate masks. For instance, as shown in Table 1, moving from $16K^2$ to $64K^2$ resolution results in a slight 2% performance drop for SAM. Table 3 presents 3D MRI

Table 3: Segmentation of BTCV [48] for multi-organ segmentation and KiTS19 [50] for Kidney Tumor Segmentation on a single GPU. *Time* indicates the end-to-end runtime to achieve the corresponding Dice Score.

| Datset | Model | Patch Size | Time | Speedup ($\times$) | Dice Score (%) |
|---|---|---|---|---|---|
| BTCV [48] | U-Net [52] | N/A | 843.90 Seconds | 9.04$\times$ | 80.2 |
| | U-Mamba [52, 54] | N/A | 8,016.24 Seconds | 0.95$\times$ | 83.51 |
| | TransUNet [55] | N/A | 3115.25 Seconds | 2.45$\times$ | 83.8 |
| | UNETR [18] | $4^3$ | 8386.56 Seconds | 0.91$\times$ | 89.1 |
| | Swin UNETR [56] | $4^3$ | 5861.93 Seconds | 1.30$\times$ | 89.5 |
| | SAM2[17] | $4^3$ | 7637.28 Seconds | 1.0$\times$ | 82.77 |
| | **SHF-SAM2** | $2^3$ | **1067.88 Seconds** | **7.15**$\times$ | **89.71** |
| KiTS [50] | U-Net [52] | N/A | 243.7 Minutes | 1.99$\times$ | 83.23 |
| | U-Mamba [52, 54] | N/A | 969.0 Minutes | 0.5$\times$ | 86.22 |
| | CoTr [55] | N/A | 488.7 Minutes | 0.99$\times$ | 84.59 |
| | UNETR [18] | $8^3$ | 513.6 Minutes | 0.94$\times$ | 86.45 |
| | nnFormer [51] | $8^3$ | 876.5 Minutes | 0.55$\times$ | 75.85 |
| | Swin UNETR [51] | $8^3$ | 748.3 Minutes | 0.65$\times$ | 81.27 |
| | Swin UNETR-V2 [51] | $8^3$ | 766.4 Minutes | 0.63$\times$ | 84.14 |
| | SAM2[17] | $8^3$ | 483.1 Minutes | 1.0$\times$ | 81.35 |
| | **SHF-SAM2** | $2^3$ | **187.3 Minutes** | **2.58**$\times$ | **87.25** |

segmentation results for BTCV and KiTS datasets at a $512^3$ resolution. Instead of processing each 2D slice independently and reconstructing the final 3D prediction as in previous works [57, 53], we replace the 2D convolution patch embedding with 3D convolution for voxel processing. As shown in the tables, SHF outperforms the prestrained SAM2 Hiera image encoder, yielding a 5.9% higher dice score and requiring $4\times$ less computational resources.

**Visual Results:** We compare the segmentation quality across different high resolutions ($[16K^2, 32K^2, 64K^2]$) for the baseline models SAM, and our proposed SHF. The segmentation results are summarized in Fig. 1. The first column shows the original input, with the label indicating the resolution. The red square highlights a small portion of the image, approximately $3.125\%$ of the entire slide. The second column presents the ground truth, followed by the predictions from different models. At higher resolutions, all models can capture the general segmentation areas. However, due to the limitations of heavy convolution-based decoders and uniform grid patching, only large patch sizes are feasible, such as the $16K^2$ patch size used by SAM and UNETR at a $64K^2$ input resolution. In contrast, at the same $64K^2$ resolution, SHF can use smaller patch sizes, as small as $8^2$, significantly improving the quality of the detailed masks.

### 5.3 SHF vs HIPT hierarchical model: training from scratch on the same datasets

To demonstrate the versatility of SHF, we compare its classification performance on the PAIP dataset with HIPT [8], a SoTA highly advanced hierarchical multi-resolution model specifically designed for microscopic pathology classification. For this experiment, we restructured the PAIP dataset, originally intended for segmentation, into six organ-based categories. Each category consists of 40 samples, with 28 for training, 8 for testing, and 4 for validation. For HIPT, we resized all samples to three resolution scales ($[256, 1024, 16384]$) and set the patch sizes for each scale to $[16, 256, 4096]$, adhering to the original settings. For SHF, we used only the $16K^2$ resolution images for classification. Instead of relying on a decoder for segmentation, we added an output channel dedicated to class prediction. In Table 4, SHF achieves significant accuracy improvement (>8%) over HIPT, even when using a vanilla ViT model and the same computational budget. At high resolution ($16K^2$), HIPT is limited to a patch size of $4096^2$ before running OOM, whereas SHF can use patch sizes as small as $2^2$ in the highest-resolution regions. This substantial accuracy boost, despite SHF using a basic ViT, indicates

Table 4: Classification (Top-1 accuracy) of vanilla ViT, HIPT [8], and SHF-ViT on PAIP dataset ($16, 384^2$ res.)

| Model | GPUs | Patch Size | Accuracy |
|---|---|---|---|
| ViT [4] | 128 | $4,096^2$ | 68.97 |
| HIPT [8] | 128 | $[16, 256^2, 4,096^2]$ | 72.69 |
| SHF-ViT-4096 | 8 | $4,096^2$ | 69.11 |
| SHF-ViT-2 | 128 | $2^2$ | **80.14** |

that: a) SHF avoids random patch dropping or padding, and b) smaller patch sizes play a more crucial role in improving performance than model complexity.

## 6 Conclusion and Future Work

This paper presents SHF, a lightweight and efficient method for enhancing ViT performance on high-resolution images. By increasing token information density and encoding hierarchical spatial structures through a hierarchical patching strategy and reverse depatching, SHF enables standard ViTs to handle long sequences effectively without requiring architectural changes. Experimental results on both 2D and 3D medical imaging tasks demonstrate significant improvements in computational efficiency and segmentation accuracy, establishing SHF as a practical solution for high-resolution medical image analysis. In future work, we aim to extend SHF to broader scientific domains, including materials discovery [58, 59], brain neuron structure detection [60, 61], and non-medical CT applications [62–64]. Additionally, we plan to enhance SHF's performance on multimodal problems [65] to support more complex and diverse applications.

## 7 Acknowledgment

This manuscript has been co-authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The U.S. Government retains a nonexclusive, worldwide license to publish or reproduce the published form of this manuscript, or to authorize others to do so, for U.S. Government purposes, as acknowledged by the publisher. This research was partially supported by the ORNL AI Initiative sponsored by the ORNL Director's Research and Development Program. This work was supported by JSPS KAKENHI under Grant Number JP21K17750.

## References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[4] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2103.17239*, 2021.

[5] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[6] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Ross Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *European Conference on Computer Vision*, 2020.

[8] Richard J. Chen, Chengkuan Chen, Yicong Li, Tiffany Y. Chen, Andrew D. Trister, Rahul G. Krishnan, and Faisal Mahmood. Scaling vision transformers to gigapixel images via hierarchical self-supervised learning. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16123–16134, New York, NY, USA, 2022. IEEE. doi: 10.1109/CVPR52688.2022.01567.

[9] Robin Strudel, Ricardo Garcia, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 7262–7272, 2021.

[10] Yuqi Si and Kirk Roberts. Three-level hierarchical transformer networks for long-sequence and multiple clinical documents classification, 2021. URL https://arxiv.org/abs/2104.08444.

[11] Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366, New York, NY, USA, 2021. IEEE.

[12] Lili Yu, DÃąniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers, 2023.

[13] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation. In *European conference on computer vision*, pages 205–218. Springer, 2022.

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.

[15] Haoqi Fan, Bo Xiong, Karttikeya Mangalam, Yanghao Li, Zhicheng Yan, Jitendra Malik, and Christoph Feichtenhofer. Multiscale vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6824–6835, 2021.

[16] Yanghao Li, Chao-Yuan Wu, Haoqi Fan, Karttikeya Mangalam, Bo Xiong, Jitendra Malik, and Christoph Feichtenhofer. Mvitv2: Improved multiscale vision transformers for classification and detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4804–4814, 2022.

[17] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. Sam 2: Segment anything in images and videos. *arXiv preprint arXiv:2408.00714*, 2024. URL https://arxiv.org/abs/2408.00714.

[18] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger R Roth, and Daguang Xu. Unetr: Transformers for 3d medical image segmentation. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 574–584, 2022.

[19] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[20] Yoo Jung Kim, Hyungjoon Jang, Kyoungbun Lee, Seongkeun Park, Sung-Gyu Min, Choyeon Hong, Jeong Hwan Park, Kanggeun Lee, Jisoo Kim, Wonjae Hong, Hyun Jung, Yanling Liu, Haran Rajkumar, Mahendra Khened, Ganapathy Krishnamurthi, Sen Yang, Xiyue Wang, Chang Hee Han, Jin Tae Kwak, Jianqiang Ma, Zhe Tang, Bahram Marami, Jack Zeineh, Zixu Zhao, Pheng-Ann Heng, RÃijdiger Schmitz, Frederic Madesta, Thomas RÃűsch, Rene Werner, Jie Tian, Elodie Puybareau, Matteo Bovio, Xiufeng Zhang, Yifeng Zhu, Se Young Chun, Won-Ki Jeong, Peom Park, and Jinwook Choi. Paip 2019: Liver cancer segmentation challenge. *Medical Image Analysis*, 67:101854, 2021. ISSN 1361-8415. doi: https://doi.org/10.1016/j.media.2020.101854. URL https://www.sciencedirect.com/science/article/pii/S1361841520302188.

[21] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models, 2023.

[22] Dacheng Li, Rulin Shao, Anze Xie, Eric P. Xing, Joseph E. Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. Lightseq: Sequence level parallelism for distributed training of long context transformers, 2023.

[23] Hao Liu, Matei Zaharia, and Pieter Abbeel. Ring attention with blockwise transformers for near-infinite context, 2023.

[24] Xiao Wang, Isaac Lyngaas, Aristeidis Tsaris, Peng Chen, Sajal Dash, Mayanka Chandra Shekar, Tao Luo, Hong-Jun Yoon, Mohamed Wahib, and John Gouley. Ultra-long sequence distributed transformer, 2023. URL https://arxiv.org/abs/2311.02382.

[25] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher RÃl'. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *NeurIPS: Proceedings of the 35th Neural Information Processing Systems Conference*, New York, NY, USA, 2022. Association for Computing Machinery. doi: 10.48550/ARXIV.2205.14135. URL https://arxiv.org/abs/2205.14135.

[26] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning, 2023.

[27] Tri Dao, Beidi Chen, Nimit S Sohoni, Arjun Desai, Michael Poli, Jessica Grogan, Alexander Liu, Aniruddh Rao, Atri Rudra, and Christopher Re. Monarch: Expressive structured matrices for efficient and accurate training. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 4690–4721. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/dao22a.html.

[28] Deyu Bo, Chuan Shi, Lele Wang, and Renjie Liao. Specformer: Spectral graph neural networks meet transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=0pdSt3oyJa1.

[29] Devin Kreuzer, Dominique Beaini, Will Hamilton, Vincent Létourneau, and Prudencio Tossou. Rethinking graph transformers with spectral attention. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 21618–21629. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/b4fd1d2cb085390fbbadae65e07876a7-Paper.pdf.

[30] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

[31] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20, New York, NY, USA, 2020. Association for Computing Machinery.

[32] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers, 2019. URL https://arxiv.org/abs/1904.10509.

[33] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *The International Conference on Learning Representations (ICLR)*, New York, NY, USA, 2020. Association for Computing Machinery. doi: 10.48550/ARXIV.2001.04451. URL https://arxiv.org/abs/2001.04451.

[34] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient Content-Based Sparse Attention with Routing Transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 02 2021. ISSN 2307-387X. doi: 10.1162/tacl_a_00353. URL https://doi.org/10.1162/tacl_a_00353.

[35] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[36] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33: 17283–17297, 2020.

[37] Chengxuan Ying, Guolin Ke, Di He, and Tie-Yan Liu. Lazyformer: Self attention with lazy update, 2021.

[38] Markus N. Rabe and Charles Staats. Self-attention does not need $o(n^2)$ memory, 2022.

[39] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34:17413–17426, 2021.

[40] Han Shi, Jiahui Gao, Xiaozhe Ren, Hang Xu, Xiaodan Liang, Zhenguo Li, and James T. Kwok. Sparsebert: Rethinking the importance analysis in self-attention. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9547–9557, New York, NY, USA, 2021. PMLR. URL `http://proceedings.mlr.press/v139/shi21a.html`.

[41] Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers. *arXiv preprint arXiv:2201.02767*, 2022.

[42] Moritz Ibing, Gregor Kobsik, and Leif Kobbelt. Octree transformer: Autoregressive 3d shape generation on hierarchically structured sequences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2697–2706, 2023.

[43] Enzhi Zhang, Isaac Lyngaas, Peng Chen, Xiao Wang, Jun Igarashi, Yuankai Huo, Masaharu Munetomo, and Mohamed Wahib. Adaptive patching for high-resolution image segmentation with transformers. In *SC24: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2024.

[44] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986. doi: 10.1109/TPAMI.1986.4767851.

[45] Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2):187–260, 1984.

[46] Hal Finkel and Jon Bentley. Quad trees: a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

[47] Timothy M. Chan, Sariel Har-Peled, and Mitchell Jones. On locality-sensitive orderings and their applications. *SIAM Journal on Computing*, 49(3):583–600, 2020. doi: 10.1137/19M1246493. URL `https://doi.org/10.1137/19M1246493`.

[48] 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 VaultâĂŤWorkshop Challenge*, 2015.

[49] Du Wu, Peng Chen, Xiao Wang, Issac Lyngaas, Takaaki Miyajima, Toshio Endo, Satoshi Matsuoka, and Mohamed Wahib. Real-time high-resolution x-ray computed tomography. In *Proceedings of the 38th ACM International Conference on Supercomputing*, pages 110–123, 2024.

[50] Nicholas Heller, Fabian Isensee, Klaus H Maier-Hein, Xiaoshuai Hou, Yucheng Xie, Fengyi Li, Yang Nan, Guangrui Mu, Zhiyong Lin, Miofei Han, Zhenlin Peng, Chunmei Sun, Ziyu Wu, Xiaojie Qiu, Zaiyi Chen, Chaojie Liang, Erick M Remer, Jennifer C Teh, Shengkai Cui, Arveen Kalapara, Valentina Sandfort, Zhoubing Xu, Neal Schenkman, Alexander Kutikov, Joel Rosenberg, and Fakhran. The kits19 challenge data: 300 kidney tumor cases with clinical context, ct semantic segmentations, and surgical outcomes.

[51] Fabian Isensee, Tassilo Wald, Constantin Ulrich, Michael Baumgartner, Saikat Roy, Klaus Maier-Hein, and Paul F Jaeger. nnu-net revisited: A call for rigorous validation in 3d medical image segmentation. *arXiv preprint arXiv:2404.09556*, 2024.

[52] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18*, pages 234–241. Springer, 2015.

[53] Jieneng Chen, Yongyi Lu, Qing Yu, Xiaomeng Luo, Ehsan Adeli, Yan Wang, Le Wang, and Chenyang Lu. Transunet: Transformers make strong encoders for medical image segmentation. *arXiv preprint arXiv:2102.04306*, 2021.

[54] Jun Ma, Feifei Li, and Bo Wang. U-mamba: Enhancing long-range dependency for biomedical image segmentation. *arXiv preprint arXiv:2401.04722*, 2024.

[55] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L. Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *CoRR*, abs/2102.04306, 2021.

[56] Yucheng Tang, Dong Yang, Wenqi Li, Holger R. Roth, Bennett A. Landman, Daguang Xu, Vishwesh Nath, and Ali Hatamizadeh. Self-supervised pre-training of swin transformers for 3d medical image analysis. In *CVPR*, pages 20698–20708. IEEE, 2022.

[57] Yuyin Zhou, Lingxi Xie, Wei Shen, Yan Wang, Elliot K Fishman, and Alan L Yuille. A fixed-point model for pancreas segmentation in abdominal ct scans. In *International conference on medical image computing and computer-assisted intervention*, pages 693–701. Springer, 2017.

[58] Tian Xie and Jeffrey C Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120(14):145301, 2018.

[59] Yue Liu, Tianlu Zhao, Wangwei Ju, and Siqi Shi. Materials discovery and design using machine learning. *Journal of Materiomics*, 3(3):159–177, 2017.

[60] Alessandro Motta, Manuel Berning, Kevin M Boergens, Benedikt Staffler, Marcel Beining, Sahil Loomba, Philipp Hennig, Heiko Wissler, and Moritz Helmstaedter. Dense connectomic reconstruction in layer 4 of the somatosensory cortex. *Science*, 366(6469):eaay3134, 2019.

[61] Michał Januszewski, Jörgen Kornfeld, Peter H Li, Art Pope, Tim Blakely, Larry Lindsey, Jeremy Maitin-Shepard, Mike Tyka, Winfried Denk, and Viren Jain. High-precision automated reconstruction of neurons with flood-filling networks. *Nature methods*, 15(8):605–610, 2018.

[62] Du Wu, Enzhi Zhang, Isaac Lyngaas, Xiao Wang, Amir Ziabari, Tao Luo, Peng Chen, Kento Sato, Fumiyoshi Shoji, Takaki Hatsui, et al. Paradigm shift in infrastructure inspection technology: Leveraging high-performance imaging and advanced ai analytics to inspect road infrastructure. *arXiv preprint arXiv:2505.13955*, 2025.

[63] Peng Chen, Mohamed Wahib, Xiao Wang, Takahiro Hirofuchi, Hirotaka Ogawa, Ander Biguri, Richard Boardman, Thomas Blumensath, and Satoshi Matsuoka. Scalable fbp decomposition for cone-beam ct reconstruction. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16, 2021.

[64] Peng Chen, Mohamed Wahib, Shinichiro Takizawa, Ryousei Takano, and Satoshi Matsuoka. Ifdk: A scalable framework for instant high-resolution image reconstruction. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–24, 2019.

[65] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[66] Joshua Ainslie, Santiago Ontanon, Chris Alberti, Vaclav Cvicek, Zachary Fisher, Philip Pham, Anirudh Ravula, Sumit Sanghai, Qifan Wang, and Li Yang. Etc: Encoding long and structured inputs in transformers. *arXiv preprint arXiv:2004.08483*, 2020.

[67] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33: 17283–17297, 2020.

[68] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[69] Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[70] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[71] Jieru Mei, Liang-Chieh Chen, Alan Yuille, and Cihang Xie. Spformer: Enhancing vision transformer with superpixel representation. *arXiv preprint arXiv:2401.02931*, 2024.

[72] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[73] The Frontier supercomputer. `https://www.olcf.ornl.gov/frontier/`.

[74] Hugo Touvron, Matthieu Cord, Alaaeldin El-Nouby, Jakob Verbeek, and Hervé Jégou. Three things everyone should know about vision transformers. In *European Conference on Computer Vision*, pages 497–515. Springer, 2022.

## NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We explained the improvements in training speed and model performance in the introduction, which can be seen in the numerical values in Table 1, Table 2, Table 3 and Table 4, or the image quality in Figure 1.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations of this application in some scenarios, especially when subdivision is wrong, in the supplementary material.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: This paper does not contain theoretical proofs and results

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: See Sec 2, step-by-step code and pseudo code in the supplement material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We added a Python pseudocode and instructions in the supplement material and will make the code public once accepted.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so âĂIJNoâĂİ is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We described the framework in the Fig. 2, added the step-by-step algorithm and detailed settings in the supplement material.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We added statistical significance for the small scale in the experiments in the supplement material. We hope the reviewer understands that the computational resources for high-resolution experiments are huge.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We provide necessary hardware requirement in Tab. 1, Tab. 2. And in the supplement material, we provide detailed resource requirements and scripts for launching the jobs.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: This paper focuses on high-resolution image segmentation.

    Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All datasets are public.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: The core method development in this research does not involve LLMs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

# Appendix

## Appendix Contents

# A Summary of Methods for Training on Long Sequences

Table 5 provides a summary of recent approaches that address the long-sequence challenges encountered by ViT models when processing high-resolution images.

Table 5: A summary of relevant long sequence training methods for solving the quadratic attention through the reduce the amount of work. Here $N$ = sequence length.

| Approach | Method | Merits & Demerits | Complexity (Best) | Model | Implementation |
|---|---|---|---|---|---|
| Attention Approximation | Longformer [35] ETC [66] | (+) Better time complexity vs Transformer.<br>(-) Sparsity levels insufficient for gains to materialize. | $O(N)$<br>$O(N\sqrt{N})$ | Some Models w/ Forked PyTorch | Self-attention Implementation |
| | BigBird [67] Reformer [68] | (+) Theoretically proven time complexity.<br>(-) High-order derivatives | $O(N \log N)$ | Some Models w/ Forked PyTorch | Self-attention Implementation |
| | Sparse Attention [69] | (+) Introduced sparse factorizations of the attention.<br>(-) Higher time complexity. | $O(N\sqrt{N})$ | Some Models w/ Forked PyTorch | Self-attention Implementation |
| | Linformer [70] Performer [30] | (+) Fast adaptation<br>(-) Assumption that self-attention is low rank. | $O(N)$ | Some Models w/ Forked PyTorch | Self-attention Implementation |
| | SPFormer [71] (Prediction) | (+) Irregular tokens.<br>(-) No adaptation to high resolution. | $O(P^2)$<br>P:num of regions | Model w/ Plain PyTorch | Model Implementation |
| Hierarchical | Hier. Transformer [10] (Text Classification) | (+) Independent hyperpara. tuning of hierarc. models.<br>(-) No support for ViT. | $O(N \log N)$ | Model w/ Plain PyTorch | Model Implementation |
| | CrossViT [11] (Classification) | (+) Better time complexity vs standard ViT.<br>(-) Complex token fusion scheme in dual-branch ViTs. | $O(N)$ | Model w/ Plain PyTorch | Model Implementation |
| | HIPT [8] (Classification) | (+) Model inductive biases of features in the hierarchy.<br>(-) High cost for training multiple models. | $O(N \log N)$ | Model w/ Plain PyTorch | Model Implementation |
| | MEGABYTE [12] (Prediction) | (+) Support of multi-modality.<br>(-) High cost for training multiple models. | $O(N^{4/3})$ | Model w/ Plain PyTorch | Model Implementation |
| State Space Model | MEGABYTE [12] (Prediction) | (+) Support of multi-modality.<br>(-) High cost for training multiple models. | $O(N^{4/3})$ | Model w/ Plain PyTorch | Model Implementation |
| High-resolution | xT [12] (Prediction) | (+) Support of multi-modality.<br>(-) High cost for training multiple models. | $O(N^{4/3})$ | Model w/ Plain PyTorch | Model Implementation |
| | MEGABYTE [12] (Prediction) | (+) Support of multi-modality.<br>(-) High cost for training multiple models. | $O(N^{4/3})$ | Model w/ Plain PyTorch | Model Implementation |
| | MEGABYTE [12] (Prediction) | (+) Support of multi-modality.<br>(-) High cost for training multiple models. | $O(N^{4/3})$ | Model w/ Plain PyTorch | Model Implementation |
| **Ours** | **Symmetrical Hierarchical Forest (Segmentation & Class.)** | **(+) Attention mechanism intact.**<br>**(+) Largely reduces computation cost; maintains quality.**<br>**(+) Efficiency depends on level of details in an image.**<br>**(-) task semantics are independent of edge information.** | $O(\log^2 N)$ | Any Model w/ Plain PyTorch | Image Pre-processing |

# B Discussion and Ablation Studies

## B.1 Sequence length $L$ and compression ratio $\gamma$

Fig. 3 illustrates how different sequence lengths affect the compression ratio of the original image, using the same input image and edge extraction algorithm. The first column displays an edge image with a resolution of $1024 \times 1024$ pixels. We evaluated sequence lengths $L = [256, 1024, 2050]$, which correspond to average patch sizes of $[31.7, 4.3, 9.17]$ pixels under the grid patch configuration. The resulting compression ratios $\gamma$ are $[4.07, 12.18, 27.66]$. For image segmentation, a sequence length of 1024 appears to provide the best balance for images at 1K resolution. In 3D, the compression ratios increase to $[6.06, 22.47, 47.76]$, as higher dimensionality leads to sparser information within each patch, thereby making experiments in higher visual dimensions feasible.

The key reason `SHF` can handle small patch sizes at high resolutions is its ability to reduce sequence length through hierarchical forest patching. In Fig. 6, we illustrate how the sequence length can be adjusted by tuning the threshold of the split value without substantially affecting prediction performance. The split value $v$ governs both the total sequence length and the distribution of patch sizes. The first row of Fig. 6 shows that halving the split value $[100, 50, 20]$ leads to a roughly proportional change in patch size distribution, with average patch sizes $[12.77, 19.17, 29.88]$. This demonstrates a linear relationship between the split value and the average patch size. In contrast, for the uniform grid patching strategy, sequence length grows quadratically as $O\big((\frac{Z}{P})^2\big)$. Using

hierarchical forest patching, however, we observe an approximately linear increase in average sequence length as the average patch size decreases. In 3D, the average patch sizes $[6.27, 11.79, 17.86]$ are larger due to increased dimensionality, which results in sparser information per patch and makes experiments in higher visual dimensions feasible.

### B.2 Image Information Loss in Patching and Depatching

In Fig. 4, we present the tree structure generated from the input image using the patch partitioning strategy, showing the effects of compressing and reconstructing the original mask. When decoding the mask through spatial partitioning, the accuracy can theoretically reach an upper limit of 99.5% with perfect predictions. However, as our experiment's sequence mask achieves only 82.97%, this tree-structured reconstruction's impact on decoding performance is relatively minor.

Notably, SHF differs from a convolution-based decoder in terms of image quality degradation. We believe this is due to two types of losses: the first is texture loss, where incorrect model regression generates inaccurate textures, leading to noise artifacts in the SHF image. The second is structural loss, observed as an amplification of noise in patches with varying structures introduced during the Dispatching stage. We attribute both losses to the loss of geometric relationships between patches during compression. Fig. 5 illustrates the degradation of these geometric properties.

### B.3 Training the SHF Algorithm: Hyperparameters and Loss

Since depatching happens in the evaluation stage, our algorithm is also divided into the training stage and the evaluation stage. Let's first look at the training stage, where $L$ represents the length of the hierachical patching sequence, $K$ represents the optional kernel size, $t_l$ and $t_h$ represent the lower and higher threshold of canny edge detection, $f$ is the model, $x$ is the input, $\theta$ is the trainable parameter, $D$ is the dataset, $N$ is the number of batches, and $E$ is the total number of epochs required. Here, we provide a Python pseudocode in Code Listing 1 for reference.

```
def build_hierachical_forest(self):
    h,w = self.domain.shape
    root = Rect(0,w,0,h)
    self.nodes = [[root, root.contains(self.domain)]]
    while len(self.nodes) < self.fixed_length:
        bbox, value = max(self.nodes, key=lambda x:x[1])
        idx = self.nodes.index([bbox, value])
        if bbox.get_size()[0] == 2:
            break

        x1,x2,y1,y2 = bbox.get_coord()
        lt = Rect(x1, int((x1+x2)/2), int((y1+y2)/2), y2)
        v1 = lt.contains(self.domain)
        rt = Rect(int((x1+x2)/2), x2, int((y1+y2)/2), y2)
        v2 = rt.contains(self.domain)
        lb = Rect(x1, int((x1+x2)/2), y1, int((y1+y2)/2))
        v3 = lb.contains(self.domain)
        rb = Rect(int((x1+x2)/2), x2, y1, int((y1+y2)/2))
        v4 = rb.contains(self.domain)

        self.nodes = self.nodes[:idx] + [[lt,v1], [rt,v2], [lb,v3], [
    rb,v4]] + self.nodes[idx+1:]
```

Code Listing 1: The implementation of the proposed hierarchical forest-building algorithm in Python.

As shown in the training stage of overview in Section 2, the input image $x$ first goes through Gaussian smoothing and canny edge detection to get $x_e$, then the edge image $x_e$ goes through hierarchical patching to get the forest $T_x$ and an encoded sequence of length $L$, and we also use $T_x$ to encode the mask $y$ to get $y_p$. Then, we use the model to train on data with $x_p$ as input and $y_p$ as mask. After obtaining the predicted encoded sequence mask $\hat{y_p}$ for the evaluation stage, we send it to the hierarchical dispatching stage and reconstruct the prediction $\hat{y}$. Then, we can calculate the dice score between ground truth $y$ and $\hat{y}$.

We train the model using the AdamW optimizer [72] with an initial learning rate of 1e-4 over 800 epochs. The first 20 epochs are allocated for learning rate warm-up, followed by a decay by a factor of 0.1 at epochs 400 and 600. By epoch 800, the model converges. To maximize training speed, we select the largest possible batch size within the GPU memory limits. The loss function combines Dice loss and binary cross-entropy (BCE) loss:

$$\begin{aligned} L(\hat{y}, y) =& w \cdot L_{ce}(\hat{y}, y) + (1-w) \cdot L_{dice}(\hat{y}, y) \\ =& -w \cdot \frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1-y_i) \log(1-\hat{y}_i)] \\ &+ (1-w) \cdot (1 - \frac{2 \sum_{i=1}^{N} (\hat{y}_i \cdot y_i) + \epsilon}{\sum_{i=1}^{N} \hat{y}_i + \sum_{i=1}^{N} y_i + \epsilon}) \end{aligned} \tag{6}$$

where $L(\hat{y}, y)$ represents the combined loss function comprised of a weighted sum of cross-entropy (CE) loss and dice loss. The parameter $w$ controls the balance between CE loss and dice loss, set to $0.5$ in our experiments. To stabilize calculations, the smoothing term $\epsilon$ is set to $1.0$.

### B.4 Learning From the Encoded Image Space

A key concern is to ensure is that the spatial structure learned in the embeddings $\hat{y}$ corresponds to the spatial structure of the sequence $s$. We can assume that if the output prediction $\hat{y}$ can completely match the compressed mask $y_c$, then there should be a minimal loss in translation to the original mask through the spatial structure. To achieve this, we expect the regression ability of the transformer to learn the implicit spatial matching from the embedding space to the geometric pixel space, as empirically observed by a direct decrease in the loss. Since, for the positional encoding, we chose a trainable embedding vector with an initial value of zero, we speculate that if the order of patches, after hierarchical forest patching, affects the learning, then the trainable positional encoding may start to learn the positioning of tokens as manifested by the Z-order curve of the quad/octree.

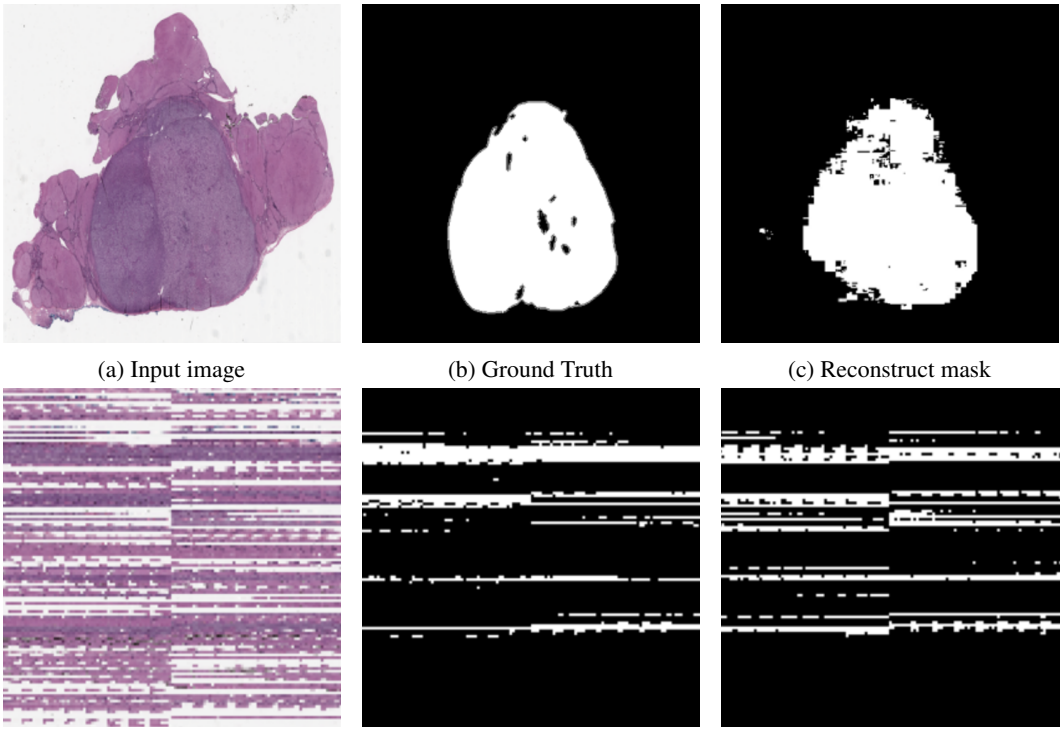
### B.5 Experimental Setup

**Hardware.** All experiments were conducted on the Frontier Supercomputer [73] at ORNL. Each Frontier node is equipped with a 64-core AMD EPYC CPU and four AMD Instinct MI250X GPUs, each with 128 GB of memory. The four MI250X GPUs on a node are interconnected via AMD Infinity Fabric at 50 GB/s. Nodes are connected through the Slingshot-11 interconnect with 100 GB/s bandwidth, across a total of 9,408 nodes.

**Software.** For the software stack, we used the PyTorch 2.4 nightly build (03/16/2024), ROCm v5.7.0, MIOpen v2.19.0, and RCCL v2.13.4 with the libfabric v1.15.2 plugin. This configuration enabled efficient multi-GPU training and high-performance communication across nodes for large-scale experiments.

### B.6 Compared to Adaptive Patching Methods

A key motivation for developing SHF was to overcome the core robustness issues and significant hyperparameter sensitivity inherent in the AP approach. Rather than merely fine-tuning hyperparameters, SHF introduces a more fundamental solution by reframing hyperparameter choice as a structured, hierarchical feature extraction problem. This design choice is the key to its superior performance. We will add a table to the main paper showing the performance gains of SHF over AP, particularly at high resolutions and on the large-scale ImageNet-1K dataset. Furthermore, we will provide visualizations in the appendix that clearly demonstrate how different parameters influence the feature extraction process across the hierarchy. To demonstrate the effectiveness of SHF over AP, in high-resolution images, the following table shows the Speedup of SHF end-to-end training for the PAIP dataset at the same segmentation quality as the baseline. We use the highest dice score of the baseline models SHF and AP.



(a) Input image (b) Ground Truth (c) Reconstruct mask

(d) Hierarchical Forest Patched Image (e) Hierarchical Forest Patched Mask (f) Hierarchical Forest Patched Prediction

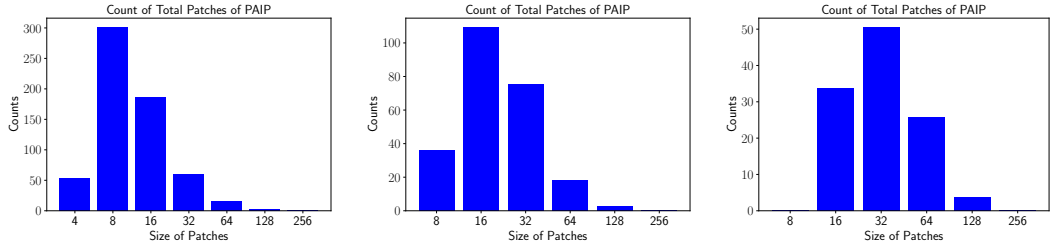
Figure 5: An illustrative example of an image processed using hierarchical forest patching, along with the corresponding reconstructed mask derived from the patching mask. During compression from the image domain to the sequence domain, the geometric relationships between patches are lost. Consequently, transformers must learn from sequences that do not preserve spatial structure and cannot rely on inherent geometric information.

### B.7 Generalization of SHF to Non-Medical Domains

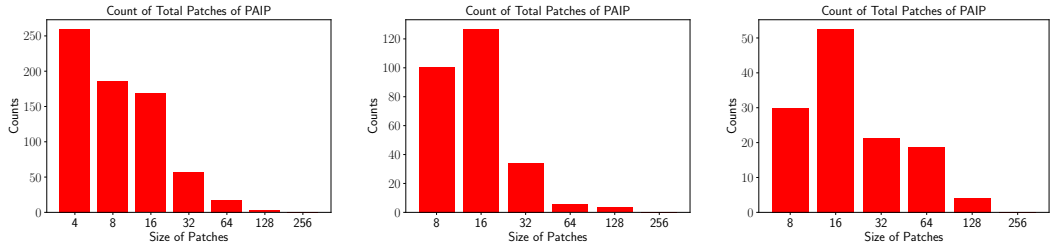
To evaluate the generalization of SHF beyond medical imaging, we conducted experiments on the widely used ImageNet dataset, as summarized in Table 6. It is important to note that SHF was originally designed for ultra-high-resolution medical images; therefore, its performance on ImageNet is not intended to replicate state-of-the-art results. Nevertheless, we include both our results and those reported in [74] for comparison. As shown in Table 6, SHF-ViT outperforms the standard ViT by leveraging hierarchical information. However, when only a single tree with fixed hyperparameters is used, performance falls below that of the basic ViT, indicating that fixed hyperparameters can limit generalization.

Table 6: Comparison of Vision Transformer (ViT) model accuracies on ImageNet-1K.

| Model | Sequence Length | Validation Acc. | V2 Acc. |
|---|---|---|---|
| ViT-B/16 [74] | 196 | 82.2% | 72.2% |
| Tree-ViT-B/16 | 256 | 77.3% | 65.1% |
| SHF-ViT-B/16 | 196 | 82.7% | 72.8% |
| ViT-L/16 [74] | 196 | 83.0% | 72.4% |
| Tree-ViT-L/16 | 256 | 77.8% | 65.7% |
| SHF-ViT-L/16 | 196 | 83.4% | 73.1% |




(a) $L = 512$, Avg patch size=12.77 (b) $L = 256$, Avg patch size=19.17 (c) $L = 128$, Avg voxel size= 29.88




(d) $L = 512$, Avg voxel size=6.27 (e) $L = 256$, Avg voxel size=11.79 (f) $L = 128$, Avg voxel size=17.86

Figure 6: Average patch size $[12.77, 19.17, 29.88]$ of training images with $1024$ resolution in PAIP and average octree voxel size $[6.27, 11.79, 17.86]$ of training images in KiTS lead to empirical linear scaling of the corresponding sequence length $[512, 256, 128]$. Compared to a 2D patch size in a normal distribution, a 3D voxel size distribution is preferred to small patches, which is considered a better compression ratio.