Adaptive Patching for High-resolution Image Segmentation with Transformers

Enzhi Zhang[1], Isaac Lyngaas[2], Peng Chen[3,5,*], Xiao Wang[2], Jun Igarashi[5], Yuankai Huo[4], Mohamed Wahib[5,*], Masaharu Munetomo[1,*]

[1] *Hokkaido University, Sapporo, Japan*

[2] *Oak Ridge National Laboratory, USA*

[3] *National Institute of Advanced Industrial Science and Technology, Japan*

[4] *Vanderbilt University, Tennessee, USA*

[5] *RIKEN Center for Computational Science, Hyogo, Japan*

enzhi.zhang.n6@elms.hokudai.ac.jp, lyngaasir@ornl.gov, chin.hou@aist.go.jp, wangx2@ornl.gov, jigarashi@riken.jp, yuankai.huo@vanderbilt.edu, mohamed.attia@riken.jp, munetomo@iic.hokudai.ac.jp

***Abstract*—Attention-based models are proliferating in the space of image analytics, including segmentation. The standard method of feeding images to transformer encoders is to divide the images into patches and then feed the patches to the model as a linear sequence of tokens. For high-resolution images, e.g. microscopic pathology images, the quadratic compute and memory cost prohibits the use of an attention-based model, if we are to use smaller patch sizes that are favorable in segmentation. The solution is to either use custom complex multi-resolution models or approximate attention schemes. We take inspiration from Adapative Mesh Refinement (AMR) methods in HPC by adaptively patching the images, as a pre-processing step, based on the image details to reduce the number of patches being fed to the model, by orders of magnitude. This method has a negligible overhead, and works seamlessly with any attention-based model, i.e. it is a pre-processing step that can be adopted by any attention-based model without friction. We demonstrate superior segmentation quality over SoTA segmentation models for real-world pathology datasets while gaining a geomean speedup of $6.9\times$ for resolutions up to $64K^2$, on up to $2,048$ GPUs.**

I. Introduction

Recently, Vision Transformers (ViTs) have emerged as a transformative paradigm in computer vision, demonstrating remarkable success in image classification tasks [1], [2], [3], [4], [5]. To effectively tackle dense prediction tasks like segmentation, numerous efforts have introduced variations on ViTs [6], [7], [8], [9]. Others have explored combinations of transformers with U-Net like architectures [7], [10], [11], [12]. However, employing ViTs with high-resolution images presents distinct scalability challenges, especially when processing small-sized image patches arranged into long sequences of complex visual data in medical imaging [10], [13].

The challenge of handling long sequences in ViTs arises from the quadratic computational complexity associated with self-attention mechanism, leading to significant computational demands [14]. Consequently, traditional ViTs encounter limitations when applied to high-resolution images, where detailed information spans extensive spatial contexts.

There are other two approaches that address the long sequence scaling problem, yet they do not reduce the total amount of work. The Sequence parallelism approach distributes long sequences into sequence segments among workers (GPUs): Deep-Speed Ulysses [15], LightSeq [16], RingAttention [17], and LLS [18]. The blocking/titling approach aims to tile the attention matrix into sub-matrices that fit into the user-managed cache memory: FlashAttention 1 [19] and 2 [20], and the Swin transformer [7] that adopts a shifted windowing technique, breaking down the image into smaller overlapping windows for processing within the transformer. The blocking/tiling approach allows scaling for longer sequences to the available memory, however, the total amount of compute is not reduced.

In contrast, there are two other approaches that address the long sequence scaling problem by reducing the amount of work (not necessarily specific to vision transformers): attention approximation and hierarchical training.

The approximation attention approaches approximate the self-attention mechanism through spectral attention [21], [22], [23], low-rank approximation [24], [25], sparse attention matrix sampling [26], [27], [28], [29], [30], infrequent self-attention updates [31], [32], or their combinations [33]. Approximation methods greatly reduce the memory and computation cost, with some reducing the quadratic complexity of self-attention to be linear. Yet, loss of information due to approximating the self-attention could have negative impact on accuracy, especially for long-range sequences. Experiments show a notable drop in accuracy when the compression ratio surpasses 70% [34]. Finally, implementing approximation approaches is complex and often requires custom operators and sparse formats.

Hierarchical training of ViTs comprises multiple transformers being trained at different levels of resolution [35], [36], [9], [37]. Training begins with the lowest-level transformer processing short sequence segments. Higher-level transformers iterate on using outputs from lower levels to process longer segments. However, employing multiple transformers increases the training time and memory usage. Moreover, managing multiple interacting transformers is complex, demanding hyperparameter tuning for the model at each resolution level.

*Corresponding authors

TABLE I: A summary of relevant long sequence training methods that reduce the amount of work. N = sequence length.

Approach	Method	Merits & Demerits	Complexity (Best)	Model	Implementation
Attention Approximation	Longformer [29] ETC [38]	(+) Better time complexity vs Transformer. (-) Sparsity levels insufficient for gains to materialize.	$O(N)$ $O(N\sqrt{N})$	Some Models w/ Forked PyTorch	Custom Self-attention Implementation
	BigBird [39] Reformer [40]	(+) Theoretically proven time complexity. (-) High-order derivatives	$O(N \log N)$		
	Sparse Attention [41]	(+) Introduced sparse factorizations of the attention. (-) Higher time complexity.	$O(N\sqrt{N})$		
	Linformer [42] Performer [43]	(+) Fast adaptation (-) Assumption that self-attention is low rank.	$O(N)$		
Hierarchical	Hier. Transformer [35] (Text Classification)	(+) Independent hyperpara. tuning of hierarc. models. (-) No support for ViT.	$O(N \log N)$	Custom Model w/ Plain PyTorch	Custom Model Implementation
	CrossViT [9] (Classification)	(+) Better time complexity vs standard ViT. (-) Complex token fusion scheme in dual-branch ViTs.	$O(N)$		
	HIPT [36] (Classification)	(+) Model inductive biases of features in the hierarchy. (-) High cost for training multiple models.	$O(N \log N)$		
	MEGABYTE [37] (Prediction)	(+) Support of multi-modality. (-) High cost for training multiple models.	$O(N^{\frac{4}{3}})$		
Ours	**Adaptive Patching** **(Segmentation & Class.)**	(+) Attention mechanism intact. (+) Negligible overhead. (+) Largely reduces computation cost; maintains quality. (-) Efficiency depends on level of details in an image.	$O(log^2 N)$	Any Model w/ Plain PyTorch	Image Pre-processing

In summary, to scale long sequences for high-resolution image segmentation trained on ViT models or U-Net models that use transformers to ingest the images, we need the following: a) be able to use smaller patch sizes that are favorable in segmentation [13], b) avoid the potential loss in performance that comes with self-attention altering mechanisms, c) avoid the high aggregate compute cost of sequence parallelism and tiling/blocking methods, and d) have a general solution that can work with transformer model, and not custom built models.

To that end, we take inspiration from the tree-based Adaptive Mesh Refinement (AMR) methods pioneered [44] and used [45], [46] for decades in HPC to dramatically reduce the computational cost of solvers applied on structured discretized meshes. We propose an Adaptive Patch Framework (APF) that is compatible with any vision transformer. APF is a preprocessing solution that uses a quadtree to partition each image in the dataset into mixed-scale patches, based on the level of detail in different regions in the image. Larger patches, that carry fewer image details are then downscaled such that all patches become the same size when being fed to the model, while keeping the core attention mechanisms and ViT model architecture intact. To demonstrate APF's scalability, we conducted extensive training of transformer-based vision models with small patch sizes for long sequences of high-resolution images. The primary contributions outlined in this paper are as follows:

- **Adaptive Patch Framework** A solution to reduce the total number of patches extracted from an image, thereby reducing the overall training cost. This not only reduces the cost of computing and memory, it also allows for using small sizes for patches, e.g., 4x4 or 2x2, which is favorable for high segmentation quality [13]. Our quantitative results demonstrate that at the same resolution levels $[512, 1024, 4096, 8192, 16384]$, a model using AFP can employ nearly $8\times$ smaller patch sizes or $64\times$ longer sequence lengths, while maintaining the same cost of traditional patching.
- **High-quality segmentation on real-world datasets** We conducted experiments on Frontier supercomputer, with up to $2,048$ MI250X using real-world high-resolution pathology datasets. At a fixed compute budget, and up to the depth of 13 multi-resolutions, we can scale to image resolutions up to $16K^2$, and lower the patch size from 16×16 to the minimum 2×2 on a vision transformer. Meanwhile, due to the smaller patch size at the same computational cost, we improve the segmentation quality by 5.5% over widely used models. Alternatively, we can reach the same segmentation quality with speedups ranging between $12.7\times$ to $3.9\times$. We also demonstrate the versatility of AFP by achieving more than 7% classification accuracy over the most sophisticated model for classifying of high-resolution microscopic pathology images.
- **Simplicity and low-overhead** Unlike existing methods that modify attention mechanisms, our solution preserves the original attention mechanism. This ensures seamless integration into any vision transformer. APF is a very low-overhead pre-processing solution, that is further amortized over epochs: the overhead is effectively negligible.

In summary, AFP offers a novel and general solution to the long-sequence challenge in ViTs, it preserves the dense self-attention merits, and reduces sequence length dramatically to boost the segmentation efficiency. This paves the way for enhanced applications of ViTs in high-resolution scientific imaging domains.

II. Background and Motivation

A. Adaptive Mesh Refinement and Quadtrees in Imaging

Structured AMR [44] uses a hierarchical spatial representation of mesh spacing. In the 2D tree-based scheme, the

mesh is organized into a hierarchy of refinement levels in a tree that represents the hierarchy of the mesh. The mesh is usually decomposed into relatively small fixed-sized quadrants of mesh cells. Each quadrant can be recursively refined into a set of quadrants of fine cells. A quadtree manages the mesh by maintaining explicit child-parent relationships between coarse and fine quadrants. At most one level of refinement difference is typically allowed between neighboring quadrants to maintain size relations. Traversing the quadrants across the three leaves corresponds to a Morton z-shaped space filling curve in the geometric domain [47]. Accordingly, sorting the tree leaf blocks by their Morton ID would give a series of blocks that are affine in the geometric space of the mesh.

A similar concept appears in computer graphics, under the name of *quadtrees*, where a mesh is replaced by an image, and mesh cells are replaced by image pixels. The history of quadtree structures dates back to early advancements in computer graphics and image processing [48], [49], [48], [49], [50], [51]. In the work of [48], [49], quadtrees (octrees) were used in 2D (3D) computer games to detect the collision of two objects efficiently in $O(n \log n)$ time complexity, where n is the number of particles. Quadtrees are also used as an image representation at different resolution levels and have been efficiently applied in image [48] and video compression [49]. Recently, quadtrees have been used in image segmentation to improve attention efficiency, e.g., quadtree attention [52], and octree transformer [53]. Both of those approaches employ quadtrees, like the work in this paper. However, we introduce a quadtree-based pre-processing patching strategy without changing the model or attention scheme. In other words, our proposal doesn't involve additional complexity and custom model design; our solution can be integrated seamlessly into the current and future transformer-based encoders.

B. Vision Transformers and Attention

Our proposed methods act as a pre-processing step to feed patches to vision transformers, or U-Net [54] like models employing transformer encoders. ViTs [2] comprise an embedding layer, transformer encoder layers, and a classification head. The embedding layer linearly projects the image patches sequence input into a sequence of flattened embeddings. Transformer encoder layers process these embeddings, capturing local and global context through self-attention mechanisms.

The 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 $T : R^{N \times F} \rightarrow R^{N \times F}$ defined by the composition of L transformer layers $T_1(\cdot), ..., T_L(\cdot)$ as follows,:

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

$A_l(\cdot)$ 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:

$$Q = xW_Q \tag{2}$$

$$K = xW_K \tag{3}$$

$$V = xW_V \tag{4}$$

$$A_{ij} = \text{Softmax}\left(\frac{(Q_i K_j)^T}{\sqrt{d_k}}\right) \tag{5}$$

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. The same is true for the memory requirements because the full attention matrix must be stored to compute the gradients for the weights of the queries, keys, and values.

We further assume that the input is the content of a square image x with a resolution of Z, that is, let $x \in R^{Z \times Z}$, and by assuming that patches arise from the uniform grid patch method of patch size p. Thus the sequence $N = (\frac{Z}{P})^2$. Therefore, the total computation and memory cost of attention scores according to resolution and patch size is $O([\frac{Z}{P}]^4)$. This complexity demonstrates the difficulties of increasing the resolution while decreasing patch size P with the uniform grid patch strategy.

C. Long Sequence Problem

Due to the quadratic cost of transformers w.r.t. the sequence length, numerous efforts have been dedicated to overcoming the long-sequence problem by reducing the amount of work. The first approach questions the necessity of full attention between all input embedding pairs. Longformer [29] introduced a localized sliding window-based mask with few global masks to reduce computation scales linearly with the input sequence. Child et al. [41] proposed a set of sparse attention kernels that reduces the complexity to $O(n\sqrt{n})$ and saves memory usage of the backward pass. Reformer [40] further reduces the complexity to $O(n \log n)$ based on locality-sensitive hashing. ETC [38] uses local and global attention instead of full self-attention to scale transformers to long documents. BigBird [39] is closely related to and built on the work of ETC. Linformer[56] assumes the self-attention is low rank, and also proposes a linear complexity transformer. Later, Performers [43] also achieved linear space and time complexity and did not rely on any priors such as sparsity or low-rankness.

The second approach reduces the attention computation by training a hierarchy of models at different resolutions. Hierarchical transformers for text classification [35] use three models to capture the structure in long sequences in documents. CrossViT [9] classifies images by running a dual-attention model, in which each branch creates a non-patch token to exchange information with the other branch by attention. HIPT [36] is a classifier for high-resolution images that trains multiple models at different resolutions to leverage the hierarchical geometric structure of visual tokens. The highest resolution model is trained with large patch sizes to reduce the

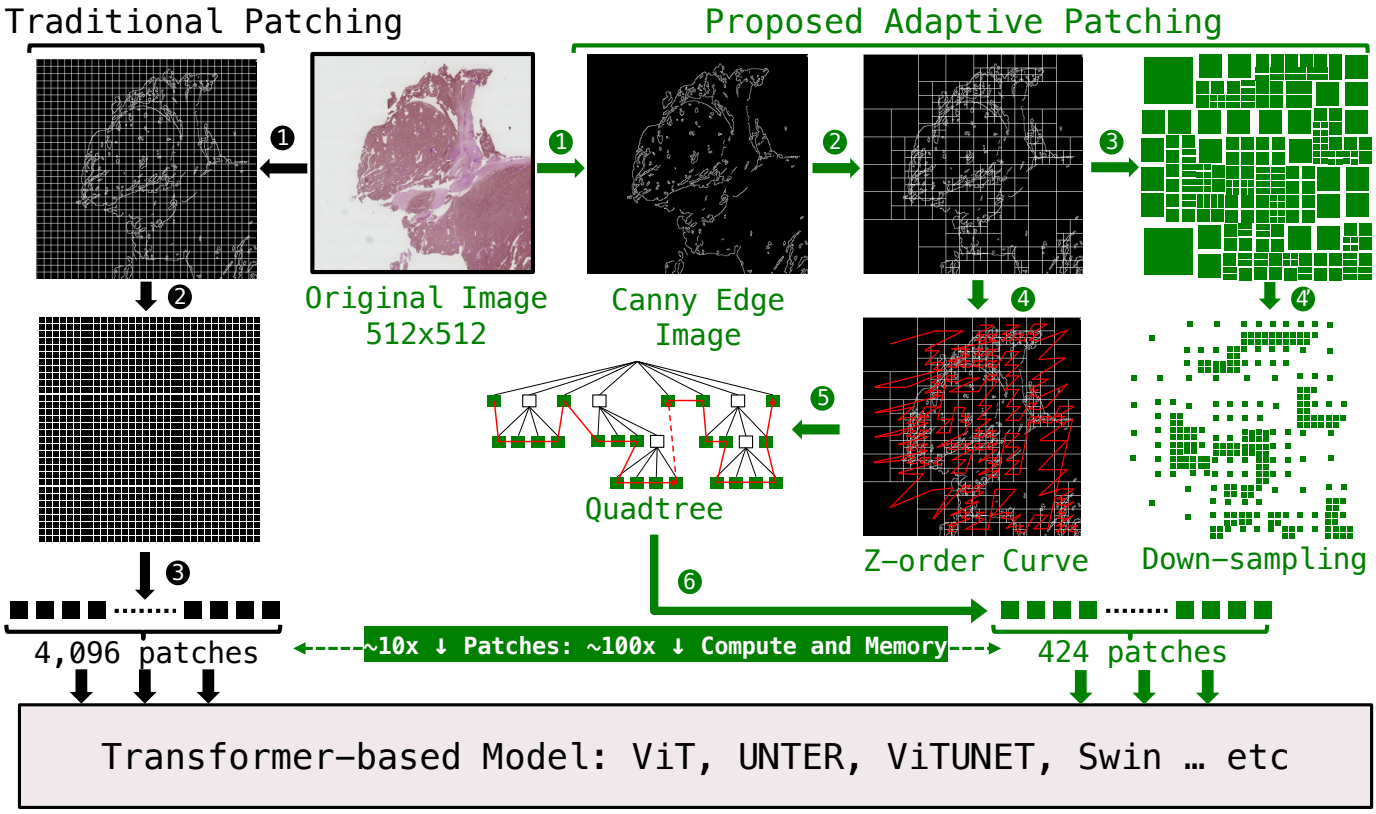


Fig. 1: Overview of AFP. The right-side flow (green) shows all the steps, starting from the original image, and ending up with feeding the patches (tokens) to an intact transformer-based model. The reduction from 4,096 to 424 patches (of size 4×4) while achieving the same dice score is from a real example of training 512×512 images from the PAIP [55] liver cancer dataset on the UNTER [10] model: $\sim 9.6\times$ reduction in sequence length, and $\sim 12.7\times$ speedup in end-to-end training.

sequence length. MEGABYTE [37] predicts patches of bytes by running local and global models at different patch sizes.

We summarize the core idea of each approach and their limitations in Table I. Hierarchical and attention approximation methods exploit the hierarchy and sparsity of the features inside the model. On the other hand, our solution is a lightweight mechanism that exploits the hierarchy and sparsity of features at different resolutions directly on the images in a pre-processing step, which leaves the attention mechanism and the model architecture intact.

D. High-Resolution Segmentation

High Resolution (HR) aggravates the long-sequence problem. Initially, the common way in literature to handle this problem was to rely on a convolutional input encoder, which first down-samples the image to learn low-resolution features [57], [58] and then up-sample to complete the prediction [59]. To benefit from the effective entire-image receptive field of transformers, many efforts turned to transformer encoders (as pure ViT or CNN+ViT), and resorted to the techniques mentioned in the previous section for handling the long sequence problem. HRViT[60], HRFormer[61], and HRNet[62] learn the HR representations by cross-resolution stream. Vision-LongFormer [63] uses a pyramid-like hierarchical structure of models at different scales to combine local attention and global memory. HIPT [36] also applied a hierarchical pyramid transformer to a pathology dataset with the utmost $4K^2$ resolution. However, in comparison to these models, our method is a pre-processing strategy, which doesn't require additional revision to of the model or attention design.

III. Adaptive Patching for High-Resolution Segmentation

Figure 1 gives an overview of the flow of AFP, in comparison to the traditional method of dividing images uniformly into equal-sized patches. AFP divides the image into patches of different sizes based on the level of details, and then downsamples the large patches so that all patches have the same size. In the next section, we follow the flow of AFP starting from the original image up until the patches are fed to the model.

A. Quadtree-based Adaptive Patches

Image and Patches We use the following notation to distinguish the size of an "image" and the "patch" corresponding to that image. Consider an image dataset D consisting of input images $x \in R^{Z \times Z}$ where Z is the resolution of image x. Then, the sequence of non-overlapping patches can be noted as $\{x_i\}_{i=1}^{N} \in R^{N \times P}$ where N is the sequence length and P is the patch size. For the traditional uniform grid patching in

ViT [2], the sequence length is $N = (\frac{Z}{P})^2$. For an image x with resolution $Z = 512$ (i.e. the image is 512×512) and patch size $P = 8$ (i.e. the patch is 8×8), the sequence length N is 4096 patches (tokens).

Edge Extraction To ignore the irrelevant details in the images x in APF, as shown in step 1 of Figure 1, we apply Gaussian Blur with kernel k and Canny [64] edge detection with lower t_l and higher threshold t_h to the original input images x. The Gaussian blur smooths the irrelevant details, and the Canny edge detection extracts the grayscale edges x_e of the image. The kernel k and threshold t can also be used as hyper-parameters for controlling the smoothing effect. During our experiments, we kept the threshold as $[100, 200]$; the kernel size is set to be $[3, 3, 5, 7, 9, 11, 13]$ for resolutions $[512, 1024, 4096, 8192, 16384, 32768, 65536]$, respectively.

Quadtree Patches The input edge x_e undergoes a recursive quadtree partitioning shown by step 2 of Figure 1, creating nodes Q_h that represent specific regions where h is the depth of the quadtree. The quadtree node Q_{h+1} is defined recursively as follows:

$$Q_{h+1} = \begin{cases} Q_h & \text{if } \sum_i D_i \leq v \text{ or } h = H \\ \{Q^h_{\text{NW}}, Q^h_{\text{NE}}, Q^h_{\text{SW}}, Q^h_{\text{SE}}\} & \text{if } \sum_i D_i > v \text{ and } h < H \end{cases} \tag{6}$$

where H is the maximum quadtree depth, v is the subdivision criterion, $Q^h_{\text{NW}}, Q^h_{\text{NE}}, Q^h_{\text{SW}}, Q^h_{\text{SE}}$ are the h-th depth child nodes representing the northwest, northeast, southwest, and southeast quadrants, respectively [65], [66]. In our implementation, the subdivision criterion constraints the total number of pixels $\sum_i D_i$ confined in the data area by the split value v. The depth limitation H is set to $[9, 10, 12, 13, 14, 15, 16]$ w.r.t. resolutions, which practically allows the input x_e to be subdivided all the way down to the 2×2 patch size level.

For uniform grid patches, we concatenate horizontal lines of patches into a 1D sequence of patches. On the other hand, for adaptive patching, after the quadtree is constructed, the patches, that is, the leaf nodes, need to be arranged. Here we show by steps 4 and 5 of Figure 1, we use a Morton Z-order curve [47] to arrange the nodes starting from the left end of the tree and going to the right. Z-order curves have the desirable property of keeping geometrically affine patches closer in the constructed sequence. After arranging the patches, since different images have different quadtree sequence lengths, firstly, we project all the different patches into the same minimized size P_m (step 4' of Figure 1). Next, we randomly drop or pad them to the same length L. Finally, the sequence of patches $x_p \in R^{L \times P_m}$ are fed to the model (step 6 of Figure 1) to train any underlying segmentation model using a transformer encoder $f(x_p; \theta)$. We summarize the above steps in Algorithm 1.

It is worth mentioning that for quadtree in the worst case, where all objects and details are in the same quadrant at the deepest level of the tree, the time complexity becomes $O(N^2)$. In the best cases, the quadtree patching strategy leads to $O(\log^2 N)$, where N is the total number of patches. However, from empirical observations in pathology datasets, instead of the best or worst cases, we observed sub-linear growth in sequence length as the average patch size decreased. This linear complexity in sequence length suggests the empirical time complexity is approximately $O(n)$.

Algorithm 1 Adaptive Patch Framework

Require: $v, H, k, t_l, t_h, f(x;\theta), N, T, D, D_p$
Initialize segment model $f(x;\theta)$.
for $n \leftarrow 1$ **to** N **do**
  $x_g = GaussianBlur(x_n; k)$
  $x_e = CannyEdge(x_g; (t_l, t_h))$
  $x_p = QuadTreePatch(x_g; v, H)$
  Add to $D_p = D_p \cup (x_p, x_n)$
end for
for $t \leftarrow 1$ **to** T **do**
  for $n \leftarrow 1$ **to** N **do**
    $x_p = D_p.pop()$
    Train $f(x;\theta)$ on the x_p.
  end for
  Evaluate $f(x;\theta)$ on validation set.
end for
Evaluate $f(x;\theta)$ on Test set.

IV. Evaluation

A. Experimental Setup

All the experiments were performed using the Frontier Supercomputer [67] at ORNL. Each Frontier node has a single 64-core AMD EPYC CPU and four AMD Instinct MI250X GPUs (128GB per GPU). The four MI250X GPUs are connected with Infinity Fabric GPU-GPU of 50GB/s. The nodes are connected via a Slingshot-11 interconnect with 100GB/s, to a total of 9,408 nodes. For the software stack, we used Pytorch 2.4 nightly build 03/16/2024. ROCm v5.7.0, MIOpen v2.19.0, RCCL v2.13.4 with libfabric v1.15.2 plugin.

B. Datasets

PAIP [55] is a high-resolution liver cancer pathology (real-world) dataset. The sample resolution size is close to $64K$, far higher than the resolution of conventional image datasets. PAIP includes $2,457$ Whole-Slide Images (WSIs). When needed to use smaller resolutions, we down-scale the images into uniform $[512, 1,024, 4,096, 8,192, 16,384, 32,768]$ square images. Before applying our quadtree patching method, we first apply Gaussian smoothing with kernel size 3×3 and $\sigma = 0$. Then, we used Canny edge detection with a lower/higher threshold of $[100, 200]$ to extract the edges from the smoothed input. During the training process, we randomly select 0.7 samples for training, 0.1 samples for validation, and 0.2 samples for testing. All data sets are shuffled and normalized to $[0.0, 1.0]$ when used as model input.

BTCV challenge [68] for 3D multi-organ segmentation contains 30 subjects with abdominal CT scans where 13 organs are annotated by experts. Each CT scan consists of 80 to 225 slices with 512^2 pixels. The multi-organ segmentation problem

TABLE II: Speedup of AFP end-to-end training for PAIP dataset at the same segmentation quality of the baseline. We use the highest dice score of the baseline model (in Table III), and report the APF configuration with similar dice scores.

Resolution	Model-Patch	Sec/Image	Sequence Length	Quadtree Depth	Dice Score (%)	Speedup (Sec/Image)	Speedup (Time to Convergence)
512×512	**APF-4**	0.06495	1,024	7	77.88	7.48×	12.71×
1 GPU	UNETR-4	0.4863	16,384	-	77.31		
$1,024 \times 1,024$	**APF-8**	0.14284	1,024	7	75.63	7.6×	12.92×
8 GPUs	UNETR-8	1.0863	16,384	-	75.72		
$4,096 \times 4,096$	**APF-16**	0.32231	2,116	8	75.74	5.77×	9.8×
128 GPUs	UNETR-32	1.8613	16,384	-	75.77		
$8,192 \times 8,192$	**APF-16**	1.1613	2,116	9	76.13	2.29×	3.89×
256 GPUs	UNETR-64	2.6618	16,384	-	75.27		
$16,384 \times 16,384$	**APF-32**	1.7613	1,024	9	75.92	2.9×	4.93×
512 GPUs	UNETR-128	5.1179	16,384	-	75.89		
$32,768 \times 32,768$	**APF-32**	2.1567	2,116	10	75.32	3.79×	6.44×
1024 GPUs	UNETR-256	8.1896	16,384	-	74.96		
$65,536 \times 65,536$	**APF-32**	5.733	4,096	11	75.82	2.3×	3.91×
2048 GPUs	UNETR-512	13.218	16,384	-	75.31		

is formulated as a 13 classes segmentation task where the dice score typically reported is the average of the 13 classes. BTCV is relatively low in resolution in comparison to the PAIP dataset (512^2 vs. $64K^2$), yet is widely used as a benchmark by the high-resolution medical segmentation community.

C. Models

Because our method is a patching strategy, it can easily replace the uniform grid patching method typically used in transformers. In our experiments, we use one of the widely-used models, UNETR [10], as the baseline model we use for AFP to conduct experiments on the high-resolution medical image segmentation task. It is worth nothing that in all our results we train the model from scratch for the target dataset: we do not do any pre-training on other datasets or fine-tune. We also report results for various other highly performing models as baselines, TransUnet [69], HIPT [36], Swin UNETR [70], ViT [2], and U-Net [54], to demonstrate different aspect about the performance and efficiency of AFP.

UNETR uses a contraction-expansion pattern consisting of several transformers as an encoder. It is connected to the decoder via a skip connection. UNETR's initial target application was 3D medical imaging for human organs. The original work [10] also discussed the impact of patch size on the model: the smaller the patch size, the better the model performance will be. However, due to the memory capacity and compute power limitation associated with quadratic attention, the authors reported that conducting experiments with a small patch size is unfeasible. Since our target experimental data is 2D medical images, we only swap the 3D convolution and deconvolution blocks in UNETR with the 2D version without additional changes to the model structure. Other than that, we make no changes nor do we tune the original UNETR model.

D. Training Setup

The loss function we applied is a combination of dice loss and binary cross-entropy loss:

$$L(\hat{y}, y) = w \cdot L_{bce}(\hat{y}, y) + (1 - w) \cdot L_{dice}(\hat{y}, y) \tag{7}$$

$$= -w \cdot \frac{1}{N} \sum_{i=1}^{N} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)] \tag{8}$$

$$+ (1 - w) \cdot (1 - \frac{2 \sum_{i=1}^{N} (\hat{y}_i \cdot y_i) + \epsilon}{\sum_{i=1}^{N} \hat{y}_i + \sum_{i=1}^{N} y_i + \epsilon}) \tag{9}$$

where $L(\hat{y}, y)$ represents the combined loss function, composed of a weighted sum of Binary Cross-Entropy (BCE) loss and dice loss. w is the weight parameter controlling the contribution of BCE loss versus the dice loss:, we set it to 0.5 during the experiments. ϵ is a smoothing term, and we keep it to 1.0 during the experiments. For the resolutions $[512, 1024, 4, 096]$, all models were trained with a batch size of 16, using the AdamW optimizer [71] with an initial learning rate of 0.0001 for 300 epochs and decay by a factor of 0.1 at epoch step $[500, 750, 875]$. For the resolutions $[8, 192, 16, 384, 32, 768, 65, 536]$, we countered the problem of fitting a single sample in memory by tuning the sequence length and training for 200 epochs.

E. Evaluation Metrics

For computational performance, we report the seconds/image of end-to-end training. For the quantitative evaluation of the segmentation result, we use the dice score, which measures the similarity between a predicted segmentation mask and the ground truth segmentation mask. The dice score (also known as the dice similarity coefficient) is defined as:

$$\text{Dice(X,Y)} = \frac{2 \times |X \cap Y|}{|X| + |Y|}$$

where X and Y are the two sets being compared. $|X \cap Y|$ represents the cardinality of the intersection of sets X and Y. $|X|$ and $|Y|$ represent the cardinality of sets X and Y

TABLE III: Improvement in quality of segmentation for the PAIP dataset against different baselines.

Resolution	Model	Patch	GPUs	Sec/Image/GPU	Depth	Sequence Length	Dice Score	Dice Improvement
512×512	**APF (+UNTER)**	2	1	0.06112	8	729	**78.32**	4.11%
		4	1	0.05975	7	676	77.88	
		8	1	0.05812	6	576	75.17	
	UNETR	4	1	0.4863	-	16,384	77.31	
		8	1	0.3746	-	4,096	75.23	
		16	1	0.1477	-	1,024	74.88	
	TransUNet	-	1	0.1783	-	1,024	73.32	
	U-Net	-	1	0.0438	-	-	70.32	
$1,024 \times 1,024$	**APF (+UNTER)**	2	8	0.2314	9	1,024	**78.42**	7.10%
		4	8	0.1786	8	900	77.64	
		8	8	0.1428	7	784	75.63	
		16	8	0.1313	6	576	74.88	
	UNETR	8	32	1.0863	-	16,384	75.72	
		16	16	0.9731	-	4,096	75.12	
		32	8	0.8874	-	1,024	73.22	
	TransUNet	-	8	1.3247	-	4,096	72.38	
	U-Net	-	1	0.0981	-	-	68.92	
$4,096 \times 4,096$	**APF (+UNTER)**	2	128	0.6938	11	4,096	**79.63**	5.09%
		4	128	0.4695	10	2,116	78.17	
		8	64	0.3824	9	1,521	75.74	
		16	32	0.3223	8	1,024	74.96	
	UNETR	32	128	1.8613	-	16,384	75.77	
	TransUNet	-	128	2.1637	-	-	71.32	
	U-Net	-	16	0.3712	-	-	64.11	
$8,192 \times 8,192$	**APF (+UNTER)**	2	256	2.3314	12	10,609	**79.56**	5.70%
		4	256	2.1314	11	8,464	78.31	
		8	128	1.7867	10	4,096	77.61	
		16	64	1.1613	9	2,116	76.13	
	UNETR	64	256	2.6618	-	16,384	75.27	
	TransUNet	-	256	2.3678	-	-	70.89	
	U-Net	-	32	1.2858	-	-	63.21	
$16,384 \times 16,384$	**APF (+UNTER)**	2	512	4.8792	13	16,384	**80.62**	6.23%
		4	256	3.1231	12	8,464	79.31	
		8	256	1.8574	11	4,096	78.84	
		16	128	1.6421	10	2,116	77.43	
	UNETR	128	512	5.1179	-	16,384	75.89	
	TransUNet	-	512	6.1296	-	-	70.46	
	U-Net	-	256	2.7825	-	-	62.97	
$32,768 \times 32,768$	**APF (+UNTER)**	4	1024	7.8916	13	16,384	**78.98**	5.36%
		8	512	6.1792	12	8,464	78.31	
		16	512	4.1685	11	4,096	77.61	
		32	256	2.1567	10	2,116	76.13	
	UNETR	256	1024	8.1896	-	16,384	74.96	
	TransUNet	-	1024	10.001	-	-	69.88	
	U-Net	-	512	4.2714	-	-	61.38	
$65,536 \times 65,536$	**APF (+UNTER)**	8	2048	12.697	13	16,384	**77.77**	3.27%
		16	1024	8.793	12	8,464	76.11	
		32	512	5.733	11	4,096	75.41	
		64	256	3.961	10	2,116	75.13	
	UNETR	512	2048	13.218	-	16,384	75.31	
	TransUNet	-	2048	14.3516	-	-	67.67	
	U-Net	-	1024	5.961	-	-	59.69	

respectively. A dice score of 100% means identical similarity between the prediction and the ground truth.

F. Results

*1) **Speedup of End-to-end Training at the Same Segmentation Quality**:* In Table II we show that under the same dice score, AFP is just a pre-processing step (on top of UNTER as baseline) that achieves a geomean speedup of $4.1\times$, if we compare on the basis that both AFP and the baseline run to the same number of epochs. Since we further observe the convergence speed in AFP to be $1.7\times$ faster, the speedup to get to the same dice score goes up to the geomean speedup of $6.9\times$. At the highest resolution of 64^2 training on 2,048 GPUs, AFP achieves $\sim 4\times$ speedup. It is worth mentioning that AFP also brings significant savings in memory and not just speedup.

*2) **Gain in Segmentation Quality**:* Table III shows segmentation improvement over different models, at different PAIP resolutions. At similar resolution, with adaptive patches we can use nearly $8\times$ smaller patch sizes at the same, computational complexity, and improve upon the original model dice score with an average of 5.5%. It is worth noting that on top of improving the dice score, we achieve those improvements with additional speedups to the training time up to $4.6\times$.

TABLE IV: Segmentation of BTCV [68] for multi-organ segmentation on one GPU. *Time* reported is the end-to-end time to reach the reported dice score.

Model	Patch Size	Time	Speedup	Dice (%)
U-Net [54]	N/A	843.90 Sec	0.79×	80.2
TransUNet [69]	N/A	3115.25 Sec	2.91×	83.8
UNETR [10]	4	8386.56 Sec	7.85×	89.1
Swin UNETR* [70]	4	6609.45 Sec	6.19×	91.8
APF-UNETR	2	1067.88 Sec	1×	89.7

*Unlike APF-UNTER, Swin UNTER is pre-trained on five datasets.

Table IV shows segmentation results for BTCV (512^2 resolution). Following [72], [11], we applied APF to each 2D slice of each CT sample and inferred all the slices to reconstruct the final 3D predictions. As shown in the table, AFP-UNTER gives higher quality than other models, with the exception of Swin UNTER (which has the advantage of being pre-trained on five other datasets before fine-tuning on BTCV). On top of getting the highest dice score, this is achieved at $>8\times$ faster training time over models with similar dice score.

*3) **Segmentation Qualitative Results**:* We demonstrate the quality of segmentation at different resolutions using different models: TransUNet, U-Net, UNETR, and our proposed APF-UNETR. We summarize the real results of the mask and display them in Figure 2. The first column shows the original input, where the label is the resolution and the scaling percentage we use to show a portion of the image.

The second column shows the ground truth, followed by the prediction results of different models. It can be seen that at 512 resolution, small patches cannot fully express the subtle differences. However, for high-resolution images, the deviations in subtle details will become larger and larger. At higher resolutions, uniform grid patching can only allow for very large patch sizes, such as $16K^2$ patch size with UNTER at input image of 64^2 resolution. However, at the same input image resolution of $64K^2$, APF-UNETR, can still use patch sizes as small as 8^2 in areas of detail by having more depth in the tree. This is the core benefit of adaptive patching.

*4) **Classification: APF vs. HIPT [36]**:* To demonstrate the versatility of APF, we compare classification for the PAIP dataset with the top performing and most sophisticated hierarchical multi-resolution model designed specially for microscopic pathology classification: HIPT [36]. In this experiment, we divided the PAIP dataset, designed originally for segmentation, into six categories according to organs. Each category contains 40 samples, 28 of which are used for model training, 8 for testing, and 4 for validation. For HIPT, we resize all samples to three resolution scales $[256, 1024, 16384]$ and set the patch size for each scale to $[16, 256, 4096]$ according to the original settings. For the APF method, we only applied a level $16,384$ image for the classification; instead of using a decoder for segmentation work, we added an additional output channel for the class prediction. As seen in Table V, with the same compute budget, using AFP with a vanilla ViT gains a huge improvement in accuracy ($>7\%$) over the very well-tuned and highly customized HIPT. At high-resolution ($16K^2$), the smallest patch size HIPT can handle, before going OOM, is $4,096^2$. AFP on the other hand can go down to patches of size 2^2 at the regions of highest resolution in the images. This big gain in accuracy, despite using a vanilla ViT with AFP, indicates: a) the effectiveness of AFP, and b) that smaller patch sizes matter more than the sophistication of the model.

TABLE V: Classification (Top-1 accuracy) of vanilla ViT, HIPT [36], and APF-ViT on PAIP dataset ($16,384^2$ res.)

Model	Num. GPUs	Patch Size	Accuracy
ViT [2]	128	4,096	68.97
HIPT [36]	128	[16,256,4096]	72.69
APF-ViT-4096	8	4096	67.73
APF-ViT-2	128	2	**79.73**

G. Discussion

*1) **Adaptive Patches Empirical Growth Complexity**:* The core reason why APF can handle small patch sizes at high resolutions is that the sequence size can be reduced with adaptive meshing. In Figure 3, we show the extent to which the sequence length can be reduced by adjusting the split value of the quadtree, without significantly losing prediction performance. The split value v controls the total length and distribution of patch sizes. The first row in Figure 3 shows that when the split value is halved $[100, 50, 20]$, the patch size distribution or the average patch size $[9.37, 20.21, 30.73]$ is also close to being halved. This means the average patch size grows linearly with the split value. For the uniform grid patching strategy, the sequence length grows by $O(\frac{Z}{P})^2$. However, we observed an approximately linear increase in the average sequence length as the average patch size decreased. Note that APF sequence length depends on the complexity of the image itself, while the best case attention complexity is $O(log^2 N)$, the worst case would be $O(N^2)$ (it becomes like uniform grid patching).

*2) **Training Stability and Patch Size**:* Figure 4 shows the training and validation curves of the models: U-Net, UNETR-32 (patch size = 32), and APF-UNETR-2 (min. patch size = 2) at $4K^2$ resolution. We can see that at the same resolution and model complexity, the UNETR model using APF can converge to a better solution that is more stable than U-Net and UNETR. We hypothesize this is because APF allows the same model to use a smaller patch size under the same model complexity. To test this hypothesis, we further tried the performance of the UNETR model with different patch sizes $[4 \times 4, 16 \times 16, 64 \times 64]$ at $1K^2$ resolution. The results further confirmed our thoughts. In Figure 4 (d,e,f), the UNETR model using a smaller patch size 4×4 tends to converge more stably than the bigger patch size 64×64.

3) Overhead of APF: Negligable: In our experiments, the time is taken for the PAIP dataset with resolutions $[512, 1024, 4096, 32768, 65536]$ is $[4.232, 7.561, 37.160, 127.374, 286.568]$ in seconds. This is negligible when compared with training time (Hours).

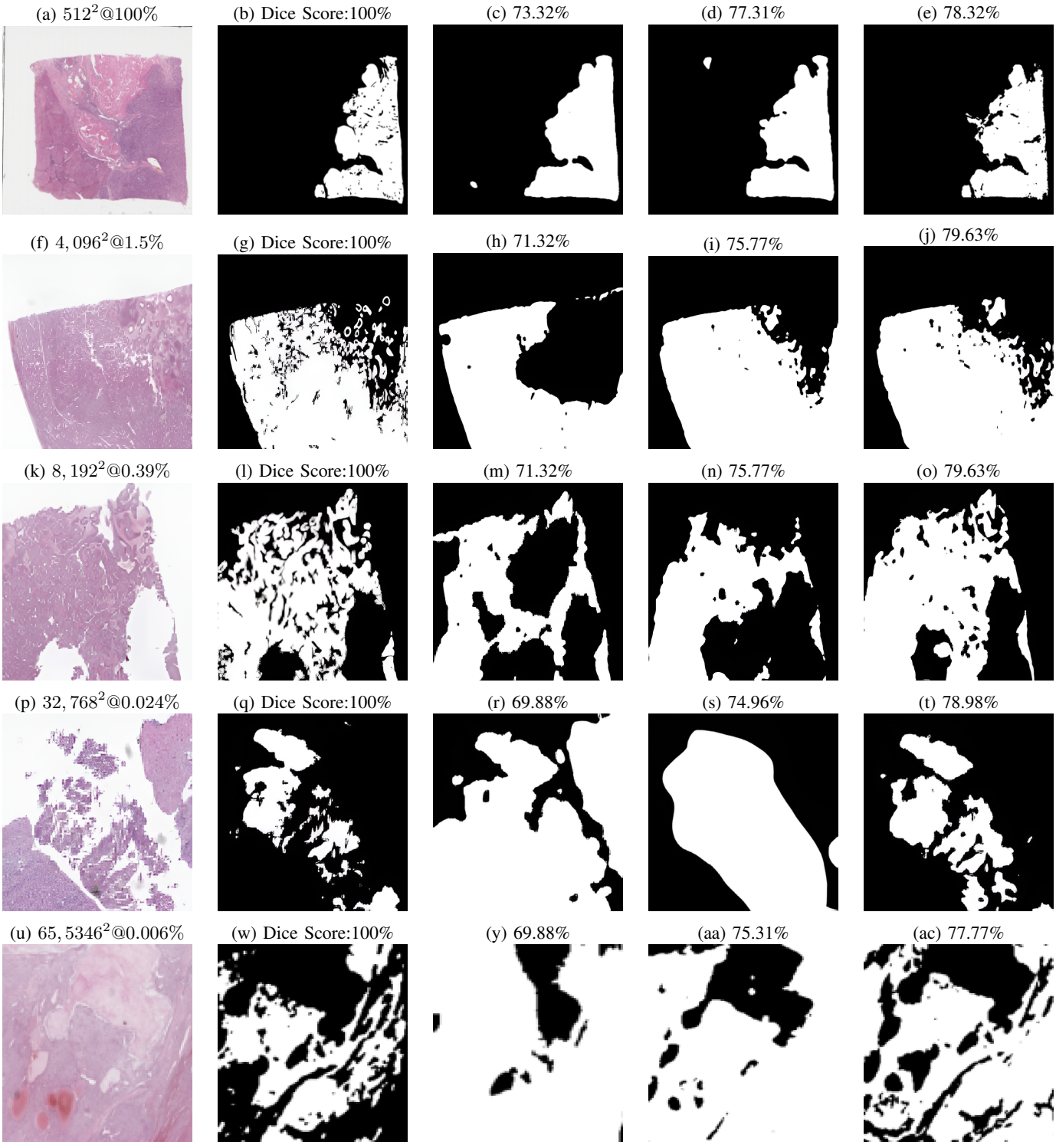
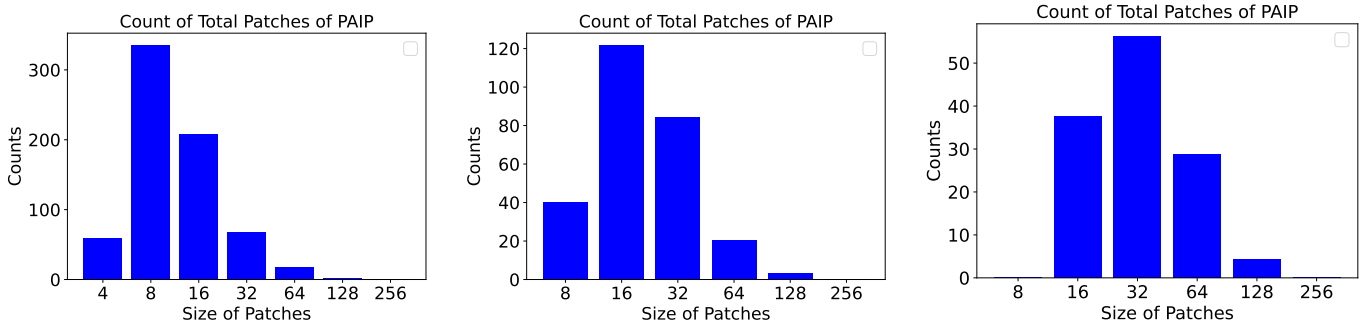


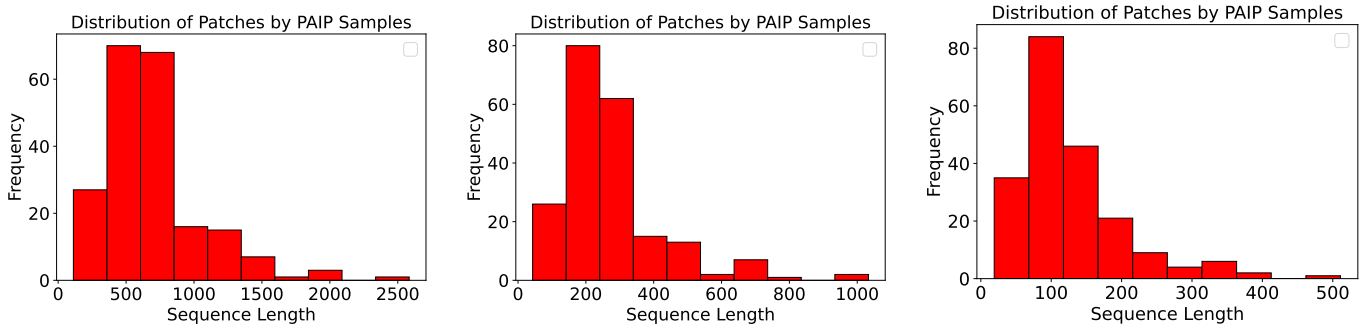
Fig. 2: Example of segmentation quality for PAIP dataset. From $4K^2$ to $64K^2$ we zoom-in to show a portion of the image.

V. Conclusion

We propose a solution that adaptively patches high-resolution images based on image details, drastically reducing the number of patches fed to vision transformer models. This pre-processing approach incurs minimal overhead. We achieve segmentation quality for $64K^2$ images comparable to SoTA models operating on no more than $4K^2$, at much higher efficiency (geomean speedup of $6.9\times$).



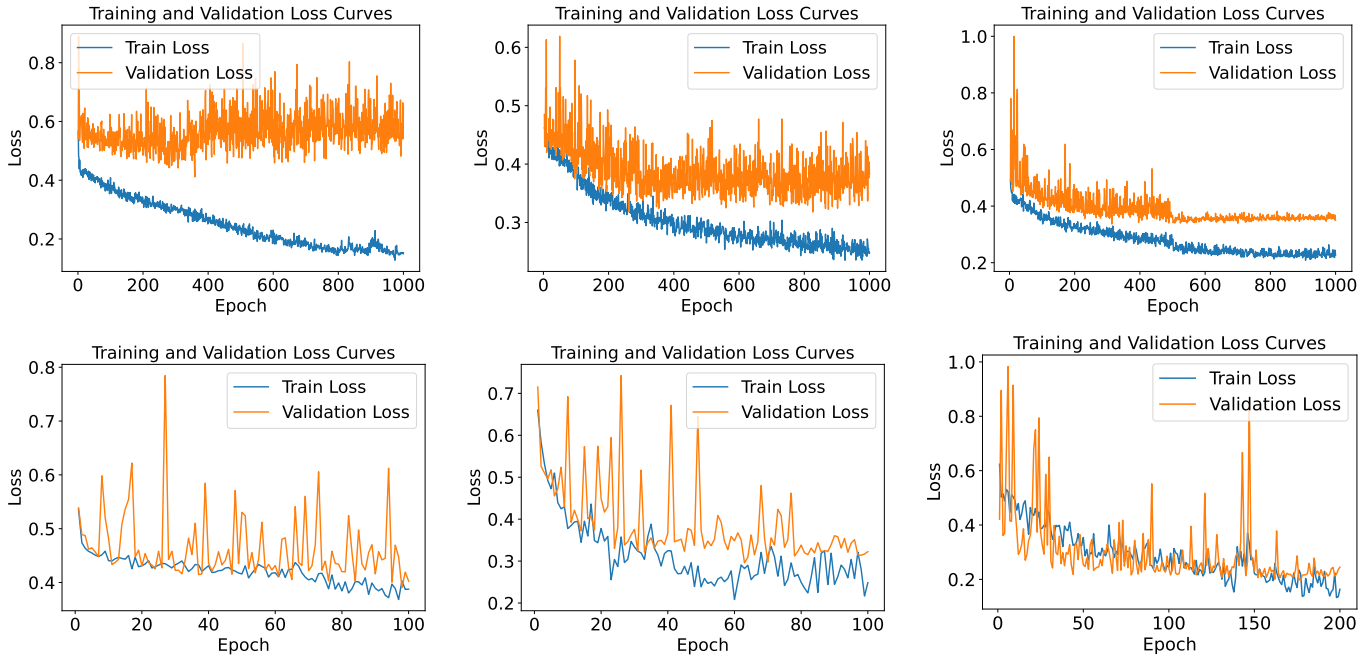
(a) $v = 20$, Avg patch size=9.37 (b) $v = 50$, Avg patch size=20.21 (c) $v = 100$, Avg patch size=30.73



(d) $v = 20$, Avg length=677.7 (e) $v = 50$, Avg length=286.9 (f) $v = 100$, Avg length=127.5

Fig. 3: Average quadtree patch size $[9.37, 20.21, 30.73]$ of training images in PAIP lead to empirical linear scaling of the corresponding average sequence length $[677.7, 286.9, 127.5]$, for different split values $[20, 50, 100]$.

(a) U-Net (b) UNETR (c) APF-UNETR



(d) UNETR: 64×64 patches (e) UNETR: 16×16 patches (f) UNETR: 4×4 patches

Fig. 4: Training and validation loss. (Top) different models. (Bottom) UNTER with different patch sizes.

ACKNOWLEDGMENT

The AI-Compliant Advanced Computer System Joint Research Project 2022 Information Initiative Center, Hokkaido University, Sapporo, Japan, partly supported the work. JST SPRING Grant Number JPMJSP2119 also supported this work.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory (ORNL), which is supported by the Office of Science of the U.S. Department of Energy (DOE) under Contract No. DE-AC05-00OR22725. This manuscript has been coauthored by ORNL, operated by UT-Battelle, LLC with the U.S.Department of Energy. This paper describes objective technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government.

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[2] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2103.17239*, 2021.

[3] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[4] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018.

[5] N. Carion, F. Massa, G. Synnaeve, N. Usunier, R. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *European Conference on Computer Vision*, 2020.

[6] H. Thisanke, C. Deshan, K. Chamith, S. Seneviratne, R. Vidanaarachchi, and D. Herath, "Semantic segmentation using vision transformers: A survey," 2023.

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

[8] Y. Li, Z. Wang, B. Li, L. Zhang, Y. Fu, Y. He, G. Xie, Z. Zeng, H. Yu, D. Chen *et al.*, "Vitgan: Training generative adversarial networks with vision transformers," *arXiv preprint arXiv:2108.05620*, 2021.

[9] C.-F. R. Chen, Q. Fan, and R. Panda, "Crossvit: Cross-attention multi-scale vision transformer for image classification," in *Proceedings of the IEEE/CVF international conference on computer vision*. New York, NY, USA: IEEE, 2021, pp. 357–366.

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

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

[12] F. Shamshad, S. Khan, S. W. Zamir, M. H. Khan, M. Hayat, F. S. Khan, and H. Fu, "Transformers in medical imaging: A survey," *Medical Image Analysis*, p. 102802, 2023.

[13] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 7262–7272.

[14] N. Parmar, A. Vaswani, J. Uszkoreit, L. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," in *International Conference on Learning Representations*, 2018.

[15] S. A. Jacobs, M. Tanaka, C. Zhang, M. Zhang, S. L. Song, S. Rajbhandari, and Y. He, "Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models," 2023.

[16] D. Li, R. Shao, A. Xie, E. P. Xing, J. E. Gonzalez, I. Stoica, X. Ma, and H. Zhang, "Lightseq: Sequence level parallelism for distributed training of long context transformers," 2023.

[17] H. Liu, M. Zaharia, and P. Abbeel, "Ring attention with blockwise transformers for near-infinite context," 2023.

[18] X. Wang, I. Lyngaas, A. Tsaris, P. Chen, S. Dash, M. C. Shekar, T. Luo, H.-J. Yoon, M. Wahib, and J. Gouley, "Ultra-long sequence distributed transformer," 2023.

[19] T. Dao, D. Y. Fu, S. Ermon, A. Rudra, and C. Ré, "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: Association for Computing Machinery, 2022. [Online]. Available: https://arxiv.org/abs/2205.14135

[20] T. Dao, "Flashattention-2: Faster attention with better parallelism and work partitioning," 2023.

[21] T. Dao, B. Chen, N. S. Sohoni, A. Desai, M. Poli, J. Grogan, A. Liu, A. Rao, A. Rudra, and C. Re, "Monarch: Expressive structured matrices for efficient and accurate training," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 4690–4721. [Online]. Available: https://proceedings.mlr.press/v162/dao22a.html

[22] D. Bo, C. Shi, L. Wang, and R. Liao, "Specformer: Spectral graph neural networks meet transformers," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=0pdSt3oyJa1

[23] D. Kreuzer, D. Beaini, W. Hamilton, V. Létourneau, and P. Tossou, "Rethinking graph transformers with spectral attention," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34. Curran Associates, Inc., 2021, pp. 21 618–21 629. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2021/file/b4fd1d2cb085390fbbadae65e07876a7-Paper.pdf

[24] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller, "Rethinking attention with performers," in *The International Conference on Learning Representations (ICLR)*. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: https://arxiv.org/abs/2009.14794

[25] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *Proceedings of the 37th International Conference on Machine Learning*, ser. ICML'20. New York, NY, USA: Association for Computing Machinery, 2020.

[26] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," 2019. [Online]. Available: https://arxiv.org/abs/1904.10509

[27] N. Kitaev, L. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," in *The International Conference on Learning Representations (ICLR)*. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://arxiv.org/abs/2001.04451

[28] A. Roy, M. Saffar, A. Vaswani, and D. Grangier, "Efficient Content-Based Sparse Attention with Routing Transformers," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 53–68, 02 2021. [Online]. Available: https://doi.org/10.1162/tacl_a_00353

[29] I. Beltagy, M. E. Peters, and A. Cohan, "Longformer: The long-document transformer," *arXiv preprint arXiv:2004.05150*, 2020.

[30] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in Neural Information Processing Systems*, vol. 33, pp. 17 283–17 297, 2020.

[31] C. Ying, G. Ke, D. He, and T.-Y. Liu, "Lazyformer: Self attention with lazy update," 2021.

[32] M. N. Rabe and C. Staats, "Self-attention does not need $o(n^2)$ memory," 2022.

[33] B. Chen, T. Dao, E. Winsor, Z. Song, A. Rudra, and C. Ré, "Scatterbrain: Unifying sparse and low-rank attention," *Advances in Neural Information Processing Systems*, vol. 34, pp. 17 413–17 426, 2021.

[34] H. Shi, J. Gao, X. Ren, H. Xu, X. Liang, Z. Li, and J. T. Kwok, "Sparsebert: Rethinking the importance analysis in self-attention," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139.

New York, NY, USA: PMLR, 2021, pp. 9547–9557. [Online]. Available: http://proceedings.mlr.press/v139/shi21a.html

[35] Y. Si and K. Roberts, "Three-level hierarchical transformer networks for long-sequence and multiple clinical documents classification," 2021. [Online]. Available: https://arxiv.org/abs/2104.08444

[36] R. J. Chen, C. Chen, Y. Li, T. Y. Chen, A. D. Trister, R. G. Krishnan, and F. Mahmood, "Scaling vision transformers to gigapixel images via hierarchical self-supervised learning," in *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. New York, NY, USA: IEEE, 2022, pp. 16 123–16 134.

[37] L. Yu, D. Simig, C. Flaherty, A. Aghajanyan, L. Zettlemoyer, and M. Lewis, "Megabyte: Predicting million-byte sequences with multiscale transformers," 2023.

[38] J. Ainslie, S. Ontanon, C. Alberti, V. Cvicek, Z. Fisher, P. Pham, A. Ravula, S. Sanghai, Q. Wang, and L. Yang, "Etc: Encoding long and structured inputs in transformers," *arXiv preprint arXiv:2004.08483*, 2020.

[39] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang *et al.*, "Big bird: Transformers for longer sequences," *Advances in neural information processing systems*, vol. 33, pp. 17 283–17 297, 2020.

[40] N. Kitaev, Ł. Kaiser, and A. Levskaya, "Reformer: The efficient transformer," *arXiv preprint arXiv:2001.04451*, 2020.

[41] R. Child, S. Gray, A. Radford, and I. Sutskever, "Generating long sequences with sparse transformers," *arXiv preprint arXiv:1904.10509*, 2019.

[42] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret, "Transformers are rnns: Fast autoregressive transformers with linear attention," in *International conference on machine learning*. PMLR, 2020, pp. 5156–5165.

[43] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser *et al.*, "Rethinking attention with performers," *arXiv preprint arXiv:2009.14794*, 2020.

[44] M. J. Berger and J. E. Oliger, "Adaptive mesh refinement for hyperbolic partial differential equations," Stanford, CA, USA, Tech. Rep., 1983.

[45] T. TU, D. R. O'HALLARON, and O. GHATTAS, "Scalable parallel octree meshing for terascale applications," in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, ser. SC '05. USA: IEEE Computer Society, 2005, p. 4. [Online]. Available: https://doi.org/10.1109/SC.2005.61

[46] M. Wahib, N. Maruyama, and T. Aoki, "Daino: a high-level framework for parallel and efficient AMR on gpus," in *SC*. IEEE Computer Society, 2016, pp. 621–632.

[47] H. Tropf and H. Herzog, "Multimensional range search in dynamically balanced trees," *Angew. Inform.*, vol. 23, pp. 71–77, 1981. [Online]. Available: https://api.semanticscholar.org/CorpusID:26857103

[48] G. v. d. Bergen, "Efficient collision detection of complex deformable models using aabb trees," *Journal of graphics tools*, vol. 2, no. 4, pp. 1–13, 1997.

[49] J. T. Klosowski, M. Held, J. S. Mitchell, H. Sowizral, and K. Zikan, "Efficient collision detection using bounding volume hierarchies of k-dops," *IEEE transactions on Visualization and Computer Graphics*, vol. 4, no. 1, pp. 21–36, 1998.

[50] J. Redding, J. Amin, J. Boskovic, Y. Kang, K. Hedrick, J. Howlett, and S. Poll, "A real-time obstacle detection and reactive path planning system for autonomous small-scale helicopters," in *AIAA Guidance, Navigation and Control Conference and Exhibit*, 2007, p. 6413.

[51] C. Ericson, *Real-time collision detection*. Crc Press, 2004.

[52] S. Tang, J. Zhang, S. Zhu, and P. Tan, "Quadtree attention for vision transformers," *arXiv preprint arXiv:2201.02767*, 2022.

[53] M. Ibing, G. Kobsik, and L. Kobbelt, "Octree transformer: Autoregressive 3d shape generation on hierarchically structured sequences," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2697–2706.

[54] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *International Conference on Medical Image Computing and Computer-Assisted Intervention*, 2015.

[55] Y. J. Kim, H. Jang, K. Lee, S. Park, S.-G. Min, C. Hong, J. H. Park, K. Lee, J. Kim, W. Hong, H. Jung, Y. Liu, H. Rajkumar, M. Khened, G. Krishnamurthi, S. Yang, X. Wang, C. H. Han, J. T. Kwak, J. Ma, Z. Tang, B. Marami, J. Zeineh, Z. Zhao, P.-A. Heng, R. Schmitz, F. Madesta, T. Rösch, R. Werner, J. Tian, E. Puybareau, M. Bovio, X. Zhang, Y. Zhu, S. Y. Chun, W.-K. Jeong, P. Park, and J. Choi, "Paip 2019: Liver cancer segmentation challenge," *Medical Image Analysis*, vol. 67, p. 101854, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1361841520302188

[56] S. Wang, B. Z. Li, M. Khabsa, H. Fang, and H. Ma, "Linformer: Self-attention with linear complexity," *arXiv preprint arXiv:2006.04768*, 2020.

[57] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 801–818.

[58] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[59] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[60] J. Gu, H. Kwon, D. Wang, W. Ye, M. Li, Y.-H. Chen, L. Lai, V. Chandra, and D. Z. Pan, "Multi-scale high-resolution vision transformer for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 12 094–12 103.

[61] Y. Yuan, R. Fu, L. Huang, W. Lin, C. Zhang, X. Chen, and J. Wang, "Hrformer: High-resolution transformer for dense prediction," *arXiv preprint arXiv:2110.09408*, 2021.

[62] J. Wang, K. Sun, T. Cheng, B. Jiang, C. Deng, Y. Zhao, D. Liu, Y. Mu, M. Tan, X. Wang *et al.*, "Deep high-resolution representation learning for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 10, pp. 3349–3364, 2020.

[63] P. Zhang, X. Dai, J. Yang, B. Xiao, L. Yuan, L. Zhang, and J. Gao, "Multi-scale vision longformer: A new vision transformer for high-resolution image encoding," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 2998–3008.

[64] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[65] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys (CSUR)*, vol. 16, no. 2, pp. 187–260, 1984.

[66] H. Finkel and J. Bentley, "Quad trees: a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[67] "The Frontier supercomputer," https://www.olcf.ornl.gov/frontier/.

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

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

[70] Y. Tang, D. Yang, W. Li, H. R. Roth, B. A. Landman, D. Xu, V. Nath, and A. Hatamizadeh, "Self-supervised pre-training of swin transformers for 3d medical image analysis," in *CVPR*. IEEE, 2022, pp. 20 698–20 708.

[71] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[72] Y. Zhou, L. Xie, W. Shen, Y. Wang, E. K. Fishman, and A. L. Yuille, "A fixed-point model for pancreas segmentation in abdominal ct scans," in *International conference on medical image computing and computer-assisted intervention*. Springer, 2017, pp. 693–701.