FROM SEQUENTIAL REFINEMENT TO ONE-SHOT GLOBAL SELECTION: MONOTONE ADAPTIVE TOKENIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Adaptive tokenization reduces the cost of transformers on high-resolution and high-dimensional data by allocating fine-grained tokens to informative regions and coarse-grained tokens elsewhere, typically through sequential Top-1 refinement over a quadtree or octree. This refinement can itself become a bottleneck: when run on the GPU, it accounts for 53–80% of each training step on our 2D benchmarks, exceeding the runtime of the model itself, because its K dependent rounds must run sequentially, each dominated by kernel-launch overhead. We observe that when importance scores are monotone along the hierarchy, with every ancestor scoring no lower than its descendants, the hierarchy satisfies the heap order property; as in classical heap selection, K sequential Top-1 refinements then select exactly the K highest-scoring regions. Monotone Adaptive Tokenization (MAT) therefore replaces sequential refinement with one global Top-K selection while preserving the exact Top-1 allocation under monotone scores; non-monotone metrics are made monotone through bottom-up sum or max reductions. Across high-resolution 2D segmentation, large-scale 3D point clouds, and multichannel weather data, MAT achieves up to $3.73\times$ and $2.56\times$ higher training and inference throughput than sequential refinement on 2D data, $1.18\times$ and $2.06\times$ over a dense-voxel baseline in 3D with substantially lower memory usage, and $1.19\times$ and $1.66\times$ over sequential refinement on weather data, while preserving downstream quality. Across the tested token budgets and attention backends, the speedups reach up to $6.3\times$ in training and $19.2\times$ in inference.[1]

1 INTRODUCTION

High-resolution and high-dimensional data are increasingly common in computer vision and scientific computing, including gigapixel images, volumetric data, point clouds, and multichannel spatial fields (Chen et al., 2022; Hatamizadeh et al., 2022; Zhao et al., 2021; Bi et al., 2023). Uniformly representing such data with fine-grained tokens produces prohibitively long token sequences (Vaswani et al., 2017; Zhang et al., 2024; Choudhury et al., 2026), even though information is often distributed highly non-uniformly across space and scale (Ronen et al., 2023; Havtorn et al., 2023) (Figure 1a). Adaptive tokenization addresses this mismatch by allocating fine-grained tokens to informative regions while retaining coarse representations elsewhere (Ronen et al., 2023; Zhang et al., 2024; Havtorn et al., 2023; Choudhury et al., 2026). Hierarchical structures such as quadtrees and octrees provide a natural way to organize such coarse-to-fine representations and have long been used for adaptive spatial decomposition (Finkel & Bentley, 1974; Samet, 1984; Meagher, 1982; Berger & Oliger, 1984).

Adaptive tokenization based on hierarchical structures introduces ancestor–descendant precedence: a region can only be refined after its ancestors have been refined (Ronen et al., 2023; Zhang et al., 2024; Choudhury et al., 2026). Existing adaptive tokenization methods enforce this precedence explicitly in the sequential refinement process. Top-1 methods such as QuadFormer (Ronen et al., 2023) repeatedly select and split the highest-scoring region, updating the refinement frontier after

[1] Our code is publicly available at `https://anonymous.4open.science/r/Monotone_Adaptive_Tokenization-76FC`.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

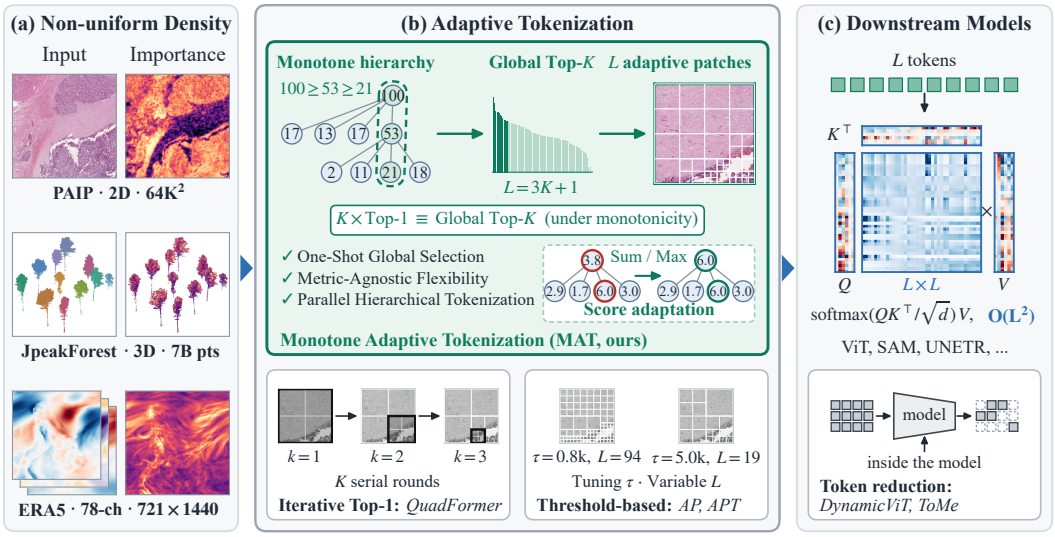

Figure 1: **Adaptive tokenization for non-uniform high-resolution and high-dimensional data.** **(a)** Different data modalities exhibit highly non-uniform information density. **(b)** MAT exploits monotone scores along the hierarchy to reformulate sequential Top-1 refinement as one-shot global Top-K selection. **(c)** MAT acts before the downstream model (Dosovitskiy et al., 2021; Kirillov et al., 2023; Hatamizadeh et al., 2022), in contrast to in-network token pruning and merging.

each step and thus proceeding sequentially. Threshold-based methods such as AP (Zhang et al., 2024) and APT (Choudhury et al., 2026) process regions in parallel within each level, but still traverse the hierarchy level by level and require score- and data-dependent threshold tuning, resulting in input-dependent token counts. When run on the GPU, sequential Top-1 refinement can become the bottleneck of the pipeline: it takes 53–80% of each training step on our 2D benchmarks, more than the model itself, because its K dependent rounds must run sequentially, each dominated by kernel-launch overhead (Section 4.4).

To remove this bottleneck, we aim to eliminate the sequential dependency of Top-1 refinement while preserving its tuning-free selection. Selection over a hierarchy with ancestor–descendant precedence also arises in multiresolution rendering (Duchaineau et al., 1997; Pajarola, 1998; Lindstrom & Pascucci, 2002) and heap selection (Frederickson, 1993; Kaplan et al., 2019). In multiresolution rendering, refinement errors are propagated upward, e.g., through a max reduction, to avoid explicitly tracking dependencies (Pajarola, 1998; Lindstrom & Pascucci, 2002; Karis et al., 2021). This produces scores that are *monotone along the hierarchy*: letting $C(r)$ denote the score of region r and $\mathrm{Anc}(r)$ its ancestors,

$$a \in \mathrm{Anc}(r) \implies C(a) \geq C(r).$$

Such a hierarchy satisfies the heap order property, and classical heap selection shows that K repeated selections of the highest-scoring node on the current frontier produce exactly the global Top-K nodes (Frederickson, 1993; Kaplan et al., 2019). Thus, under monotone scores and consistent tie breaking, K sequential Top-1 refinements are exactly equivalent to one global Top-K selection: $K \times \mathrm{Top\text{-}1} \equiv \mathrm{Global\ Top\text{-}K}$. Importantly, monotone scores already arise in adaptive tokenization: AP and SHF use additive Canny edge counts over regions (Zhang et al., 2024; 2025a), so a parent necessarily scores no lower than its children, yet neither method exploits this property. We therefore introduce Monotone Adaptive Tokenization (MAT), which uses monotone scores to reformulate sequential Top-1 refinement as one-shot global Top-K selection: (1) compute scores over the hierarchy and make them monotone when necessary, (2) perform one global Top-K selection, and (3) extract the selected regions. MAT removes sequential refinement dependencies while preserving the exact Top-1 allocation and ancestor–descendant precedence, avoids threshold tuning, retains explicit token-budget control, and maps naturally to GPU-friendly operations such as Top-K. For non-monotone importance criteria, MAT constructs monotone scores through simple bottom-up sum or max reductions, following the same error-propagation idea used in multiresolution rendering

(Pajarola, 1998; Lindstrom & Pascucci, 2002; Karis et al., 2021). The formulation applies naturally to quadtrees in 2D, octrees in 3D, and hierarchical representations more generally.

We evaluate MAT across high-resolution 2D images, large-scale 3D point clouds, and multichannel spatiotemporal weather data. MAT consistently accelerates training and inference while preserving downstream quality and substantially reducing memory relative to dense 3D baselines. Table 1 shows consistent end-to-end gains across all three modalities, with 1.18–$3.73\times$ higher training throughput and 1.66–$2.56\times$ higher inference throughput over sequential refinement or dense baselines while preserving downstream quality. Across the tested budgets and attention backends, speedups reach up to $6.3\times$ in training and $19.2\times$ in inference on $8\text{K} \times 8\text{K}$ inputs.

Our contributions are threefold: ❶ We connect adaptive tokenization to monotone refinement in multiresolution rendering and classical heap selection, showing that, under monotone scores, K sequential Top-1 refinements can be reformulated exactly as one global Top-K selection. ❷ We develop MAT, which implements one-shot global Top-K selection with GPU-friendly operations and supports non-monotone importance criteria through monotone score adaptation. ❸ We validate MAT across 2D high-resolution image segmentation, large-scale 3D point-cloud processing, and multichannel weather downscaling, demonstrating consistent end-to-end efficiency gains while preserving downstream quality.

2 RELATED WORK

High-resolution and high-dimensional data often exhibit strongly non-uniform information density, motivating adaptive allocation of representation and computation according to local information content (Berger & Oliger, 1984; Berger & Colella, 1989; Samet, 1984; Zhang et al., 2024; Havtorn et al., 2023). Existing approaches introduce such adaptivity at different stages: through the model architecture, during network execution, or directly in the input representation.

Hierarchical and multiscale architectures. PVT (Wang et al., 2021), Swin (Liu et al., 2021), and MViT (Fan et al., 2021) improve multiscale processing through pyramidal, windowed, and hierarchical designs, while QuadTree Attention (Tang et al., 2022) organizes attention coarse-to-fine with a quadtree. These methods modify the model architecture or attention pattern and are orthogonal to MAT, which adapts the input representation before model processing.

In-network token pruning and merging. TokenLearner (Ryoo et al., 2021) learns a compact set of adaptive tokens, while DynamicViT (Rao et al., 2021), EViT (Liang et al., 2022), A-ViT (Yin et al., 2022), and ATS (Fayyaz et al., 2022) progressively select, reorganize, halt, or sample tokens during network execution. ToMe (Bolya et al., 2023) instead shortens the sequence by merging similar tokens across layers. These methods adapt the token sequence within the model, whereas MAT adapts the input representation before model processing.

Adaptive tokenization. QuadFormer (Ronen et al., 2023) refines a quadtree one region at a time using saliency- or blur-based scores, AP (Zhang et al., 2024) splits regions whose Canny edge count exceeds a threshold, and APT (Choudhury et al., 2026) thresholds patch entropy across scales. SHF (Zhang et al., 2025a) builds on edge-based patching with multilevel structural information for high-resolution medical segmentation, MSViT (Havtorn et al., 2023) learns local coarse-or-fine token scales, and GPSToken (Zhang et al., 2025b) constructs irregular Gaussian tokens through iterative spatial partitioning. These methods differ mainly in how regions are scored; MAT is complementary, as it changes only how refinements are selected and can adopt their criteria directly or after score adaptation (Section 4.5).

Heap selection and multiresolution rendering. Classical heap selection algorithms reduce the sequential complexity of selecting the K best nodes of a heap-ordered tree (Frederickson, 1993; Kaplan et al., 2019), and explorable heap selection addresses settings where scores are revealed only through traversal (Karp et al., 1986; Borst et al., 2025). MAT instead exploits the fact that candidate scores in adaptive tokenization can be evaluated without following the refinement traversal, replacing sequential refinement with GPU-friendly operations such as global Top-K. In multiresolution rendering, monotone errors, often obtained by saturation, encode hierarchical dependencies in the

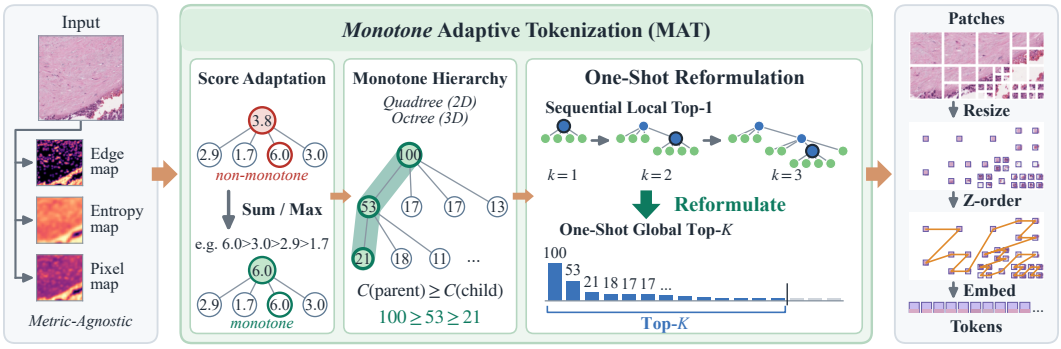

Figure 2: **Overview of Monotone Adaptive Tokenization (MAT).** Task-specific importance scores are adapted, when necessary, to be monotone along the hierarchy, enabling sequential Top-1 refinement to be reformulated exactly as one-shot global Top-K selection. Score construction, Top-K selection, and region extraction all map to GPU-friendly operations.

scores and enable greedy or threshold-based level-of-detail selection (Duchaineau et al., 1997; Pajarola, 1998; Lindstrom & Pascucci, 2002; Karis et al., 2021). MAT adopts the principle of monotone scores but uses the resulting heap order for one-shot global Top-K selection, preserving the exact Top-1 allocation without threshold tuning.

3 METHODS

3.1 HIERARCHICAL REFINEMENT WITH ANCESTOR PRECEDENCE

Let $\mathcal{T}$ denote the set of refinable nodes in a rooted hierarchy up to a prescribed maximum depth, where each node represents a candidate region, and let $\mathrm{Child}(r)$ and $\mathrm{Anc}(r)$ denote the children and ancestors of region r. Refining r replaces it with its children, so a tokenization is determined by the set $S \subseteq \mathcal{T}$ of refined regions. Since a region becomes available only after its parent has been refined, every valid refinement set is ancestor closed:

$$r \in S \implies \mathrm{Anc}(r) \subseteq S. \tag{1}$$

Given an importance score $C(r)$ for each region, sequential Top-1 refinement grows S one region at a time. It maintains the frontier

$$F(S) = \{\, r \in \mathcal{T} \setminus S : \mathrm{Anc}(r) \subseteq S \,\}$$

of unrefined regions whose ancestors have all been refined, selects $r^\star = \arg\max_{r \in F(S)} C(r)$, adds it to S, and updates the frontier by removing $r^\star$ and inserting its refinable children. Precedence is thus enforced through the evolving frontier. We next show that, when scores are monotone, the same constraint is already encoded in the global score ordering.

3.2 MONOTONE SCORES AND HEAP ORDER

We say that scores are *monotone along the hierarchy* if

$$a \in \mathrm{Anc}(r) \implies C(a) \geq C(r). \tag{2}$$

Under this condition, the hierarchy satisfies the heap order property with respect to C: no region scores higher than any of its ancestors. We fix a total order $r_1 \succ r_2 \succ \cdots$ on $\mathcal{T}$ that sorts regions by decreasing score and places ancestors before descendants among equal scores. The following results are standard consequences of the heap order property underlying classical heap selection (Frederickson, 1993; Kaplan et al., 2019), stated here in the notation of adaptive refinement.

Lemma 1 (Top-K closure). Under equation 2 and the ordering above, every prefix $\{r_1, \ldots, r_K\}$ is ancestor closed.

Proof. Let r be among the first K regions and $a \in \mathrm{Anc}(r)$. By equation 2, $C(a) \geq C(r)$, so a precedes r either by score or, for equal scores, as an ancestor; hence a is also among the first K regions. □

Proposition 1 (Exact equivalence of greedy refinement and global selection). Under equation 2, with ties in sequential Top-1 broken by $\succ$, K sequential Top-1 refinements select exactly $\{r_1, \ldots, r_K\}$:

$$K \times \text{Top-1} \equiv \text{Global Top-K}.$$

Proof. Suppose that after k steps sequential Top-1 has selected $S_k = \{r_1, \ldots, r_k\}$, which holds trivially for $k = 0$. By Lemma 1, all ancestors of r_{k+1} lie in S_k, so $r_{k+1} \in F(S_k)$. Every region ranked ahead of r_{k+1} already belongs to S_k, so r_{k+1} is the highest-ranked region on the frontier and is selected at step $k + 1$. Induction over k completes the proof. □

Monotone scores therefore encode ancestor precedence in the global ordering: the refinement is no longer defined by a sequential trajectory, but by the K highest-ranked regions.

3.3 Constructing Monotone Scores

Some importance criteria are monotone by construction. Additive regional quantities, such as the Canny edge counts used by AP and SHF (Zhang et al., 2024; 2025a), satisfy equation 2 directly: computed on a fixed edge map, the count of a parent equals the sum of the counts of its children. Other importance criteria need not satisfy equation 2. For these, let $c(r)$ denote the base importance score at the finest level. We initialize $C_{\mathrm{sum}}(r) = C_{\max}(r) = c(r)$ at this level and propagate scores upward using

$$C_{\mathrm{sum}}(r) = \sum_{u \in \mathrm{Child}(r)} C_{\mathrm{sum}}(u), \qquad C_{\max}(r) = \max_{u \in \mathrm{Child}(r)} C_{\max}(u).$$

For nonnegative base scores, sum reduction is monotone and naturally extends additive regional quantities such as edge counts. Max reduction is monotone by construction and follows the upward error propagation used to obtain monotone refinement errors in multiresolution rendering (Pajarola, 1998; Lindstrom & Pascucci, 2002; Karis et al., 2021). Proposition 1 applies to the adapted scores: MAT exactly matches sequential Top-1 refinement under the same adapted scores, but need not preserve the trajectory induced by the original non-monotone criterion. Section 4.5 shows that downstream accuracy remains comparable under this adaptation.

3.4 Monotone Adaptive Tokenization

Given an importance criterion, MAT (Figure 2) maps adaptive tokenization to three GPU-friendly operations: (1) construction of a monotone score for every candidate region, using the adaptation in Section 3.3 when necessary, (2) one global Top-K selection of the highest-ranked regions, and (3) extraction of the resulting adaptive tokens. By Proposition 1 and Lemma 1, the selected set equals that of K sequential Top-1 refinements and is ancestor closed, so no frontier needs to be maintained. The selected regions are refined, and the output tokens are the unrefined regions whose ancestors are all refined. With a fixed branching factor b and R initial regions, each refinement replaces one region with b children, so K directly determines the token count:

$$L = R + (b - 1)K.$$

We use quadtrees ($b = 4$) for 2D data and octrees ($b = 8$) for 3D data (Samet, 1984; Meagher, 1982); Proposition 1 also holds for general rooted hierarchies, although with variable branching factors K alone no longer determines L (Appendix A.1.3).

In terms of complexity, let $N = |\mathcal{T}|$ and $D = O(\log_b N)$, with $O(1)$ cost per score evaluation as under bottom-up reductions. Sequential Top-1 requires $O(bK \log(bK))$ work, but its K dependent selections yield $\Omega(K)$ span. MAT requires $O(N)$ work, with $O(D)$ span for score construction. The remaining global Top-K can be implemented efficiently using parallel GPU selection primitives such as radix Top-K (Zhang et al., 2023). Thus, MAT removes the K dependent refinement chain, making its runtime much less sensitive to the token budget in practice (Section 4.4).

Table 1: End-to-end training and inference efficiency across 2D high-resolution images, 3D point clouds, and multichannel scientific data, measured on a single NVIDIA GB200 GPU. Top-1 is iterative refinement (Ronen et al., 2023), Batch-M refines m patches per round, APT (Choudhury et al., 2026) refines every cell above a calibrated threshold, and MAT is ours. DynamicViT (Rao et al., 2021) and ToMe (Bolya et al., 2023) perform in-network token reduction at matched mean token counts, and Uniform Voxel retains the dense 3D voxel representation. Speedups are relative to the first row of each dataset block. ERA5 accuracies are means over three seeds.

Dataset (Metric)	Tokenization Method	Token Handling	Token Budget	Training			Inference			Task Quality
				Samples/s	Speedup	GPU Mem.(GiB)	Samples/s	Speedup	GPU Mem.(GiB)	
PAIP $4K \times 4K$ Dice	Top-1	Adaptive	8,191	5.46	1.00×	15.1	4.43	1.00×	4.1	80.75
	Batch-M ($m = 4$)	Adaptive	8,191	9.15	1.68×	15.1	6.76	1.53×	4.1	79.79
	Batch-M ($m = 16$)	Adaptive	8,191	15.71	2.88×	15.1	9.71	2.19×	4.1	79.55
	Batch-M ($m = 64$)	Adaptive	8,191	19.00	3.48×	15.1	10.89	2.46×	4.1	79.10
	APT	Adaptive	7,123	12.80	2.35×	26.1	8.64	1.95×	4.7	79.89
	DynamicViT (ρ=0.544)	Pruning	8,191	8.69	1.59×	29.1	10.24	2.31×	5.3	75.53
	ToMe (r=1,504)	Merging	8,191	16.41	3.01×	17.6	9.90	2.23×	5.1	77.34
	MAT (Ours)	Adaptive	8,191	**20.37**	**3.73**×	15.1	**11.34**	**2.56**×	4.1	80.77
S8D $8K \times 8K$ Dice	Top-1	Adaptive	8,191	3.80	1.00×	18.7	4.13	1.00×	7.4	95.49
	Batch-M ($m = 4$)	Adaptive	8,191	5.23	1.38×	18.7	6.04	1.46×	7.4	95.43
	Batch-M ($m = 16$)	Adaptive	8,191	6.65	1.75×	18.7	8.13	1.97×	7.4	95.39
	Batch-M ($m = 64$)	Adaptive	8,191	7.09	1.87×	18.7	8.89	2.15×	7.4	95.27
	APT	Adaptive	7,890	7.13	1.88×	20.5	9.00	2.18×	7.4	95.33
	DynamicViT (ρ=0.544)	Pruning	8,191	4.99	1.31×	33.3	**9.24**	**2.24**×	8.3	95.97
	ToMe (r=1,504)	Merging	8,191	6.85	1.80×	21.8	8.94	2.16×	8.1	**96.01**
	MAT (Ours)	Adaptive	8,191	**7.27**	**1.92**×	18.7	9.18	2.22×	7.4	95.51
JpeakForest F1	Vanilla ForestFormer3D	-	154.65M	1.38	1.00×	28.74	0.133	1.00×	14.80	83.01
	Uniform Voxel	-	154.65M	1.73	1.25×	62.29	0.172	1.29×	50.45	83.10
	Top-1	Adaptive	29.34M	0.565	0.41×	13.68	0.016	0.12×	13.60	83.20
	Batch-M (m=1,400)	Adaptive	29.32M	2.04	1.48×	**11.37**	**0.365**	**2.74**×	**10.88**	82.97
	APT	Adaptive	29.34M	2.02	1.47×	17.61	0.351	2.64×	14.86	82.81
	MAT (Ours)	Adaptive	29.34M	**2.05**	**1.48**×	13.68	0.354	2.66×	13.61	**83.23**
ERA5 $100 \times R^2$	Vanilla ORBIT-2 (AP)	Adaptive	127	21.8	1.00×	1.41	29.9	1.00×	1.22	78.21
	Top-1	Adaptive	128	539.5	24.78×	1.43	824.6	27.56×	1.25	78.20
	Batch-M ($m = 4$)	Adaptive	128	614.1	28.20×	1.43	1220.4	40.79×	1.25	78.26
	Batch-M ($m = 16$)	Adaptive	128	598.9	27.50×	1.43	1254.0	41.91×	1.25	77.64
	Batch-M ($m = 64$)	Adaptive	128	612.0	28.10×	1.43	1229.7	41.10×	1.25	78.26
	APT	Adaptive	125	627.6	28.82×	1.44	1037.5	34.68×	1.26	78.26
	MAT (Ours)	Adaptive	128	**643.0**	**29.53**×	1.43	**1372.4**	**45.87**×	1.25	**78.27**

4 EVALUATION

4.1 BASELINES, IMPLEMENTATION AND DATASETS

We compare MAT with three preprocessing adaptive tokenization methods, **Top-1** (Ronen et al., 2023), **Batch-M**, and **APT** (Choudhury et al., 2026), and two in-network token reduction methods, **DynamicViT** (Rao et al., 2021) and **ToMe** (Bolya et al., 2023). Top-1 performs sequential refinement, Batch-M refines the top m leaves per round, and APT uses threshold-based refinement with input-dependent token counts; DynamicViT and ToMe reduce dense token sequences through pruning and merging, respectively. For PAIP and S8D, we use UNETR (Hatamizadeh et al., 2022) with a ViT-B backbone (Dosovitskiy et al., 2021); uniform tokenization at a comparable patch size is reported to exceed GPU memory (Zhang et al., 2024), so we compare only adaptive and in-network baselines. The 3D experiments use ForestFormer3D (Xiang et al., 2025), including Uniform Voxel as the dense baseline, while ERA5 uses the Res-Slim-ViT architecture from ORBIT-2 (Wang et al., 2025). Detailed training and tokenization settings are provided in Appendix A.2.

PAIP (Kim et al., 2021; PAIP 2019 Challenge, 2019) is a high-resolution **2D pathology** dataset for liver cancer segmentation, with 50 annotated training whole-slide images tiled into 3,061 training and 778 validation $4K \times 4K$ RGB crops; Canny edge count is used as the importance score (Canny, 1986). **S8D** is a high-resolution **2D scientific imaging** dataset of $8K \times 8K$ X-ray CT slices from SPring-8 (SPring-8, 2026), containing 3,080 slices from 43 volumes, split into 2,430 training and 650 validation slices for five-class phase segmentation; we again use Canny edge count. **JpeakForest** is a large-scale **3D point-cloud** dataset of ten UAV laser-scanning scenes containing 7.0 billion points in total, used for semantic and per-tree instance segmentation (Xiang et al., 2025; Puliti et al., 2023); mean channel variance is used as the octree importance score. **ERA5** (Hersbach et al., 2020) provides **multichannel spatiotemporal scientific fields** through the ORBIT-2 surrogate weather model (Wang et al., 2025), with 8,760 hourly samples from 2020 split chronologically into $80/10/10$ for training, validation, and testing; each sample maps 23 atmospheric variables to

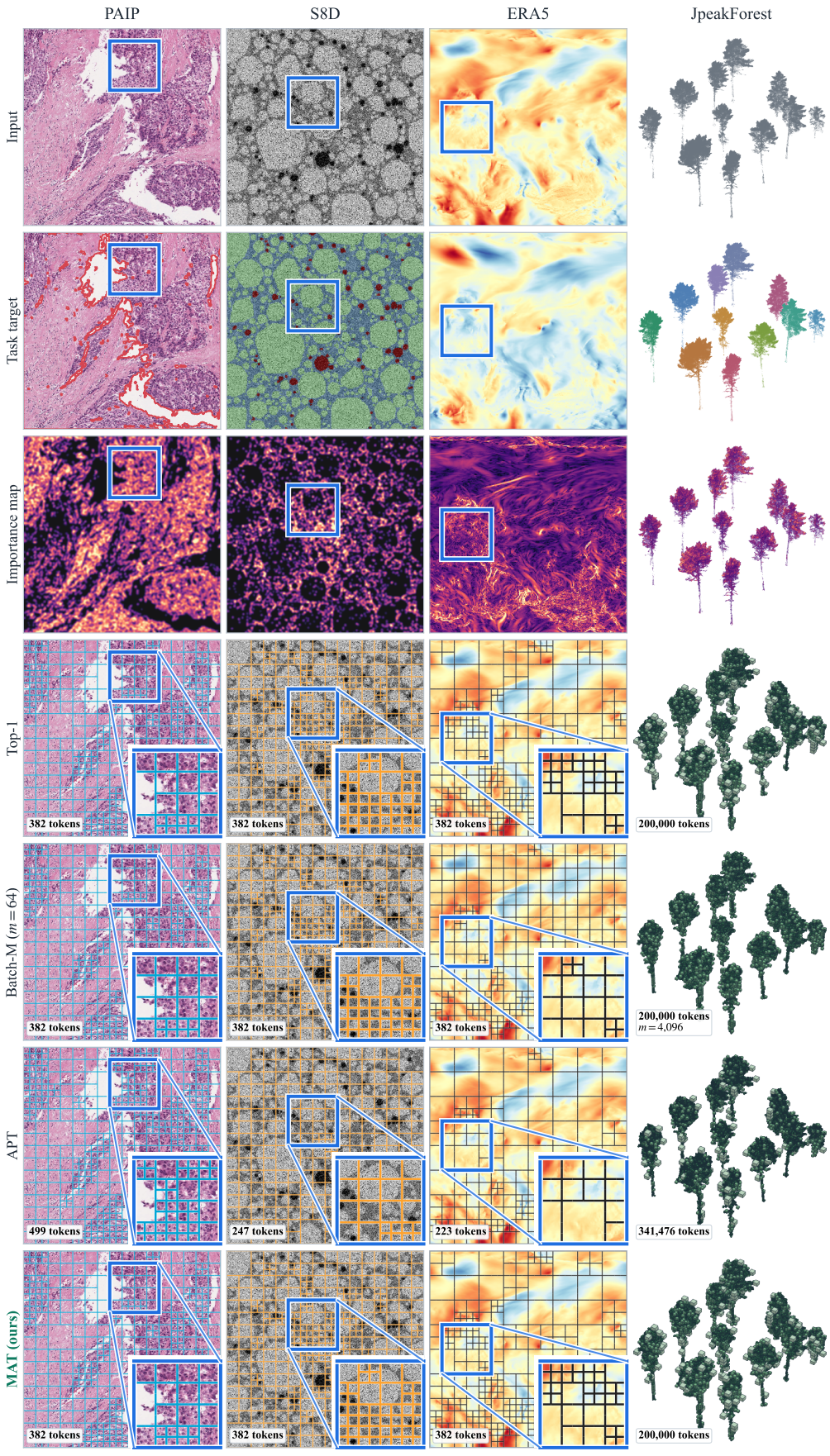

Figure 3: Adaptive tokenization under four refinement rules across high-resolution 2D images (PAIP and S8D), multichannel scientific data (ERA5), and a 3D point cloud (JpeakForest).

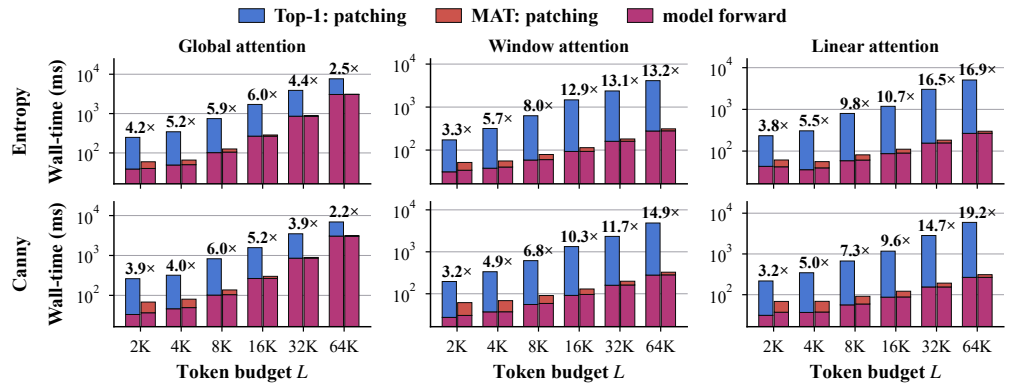

Figure 4: End-to-end inference wall-time breakdown at $8\text{K} \times 8\text{K}$ on a single GPU with batch size 1. Bars separate adaptive tokenization from model forward. Top-1 denotes the sequential refinement strategy of QuadFormer (Ronen et al., 2023). Numbers above the bars indicate MAT's end-to-end speedup over Top-1.

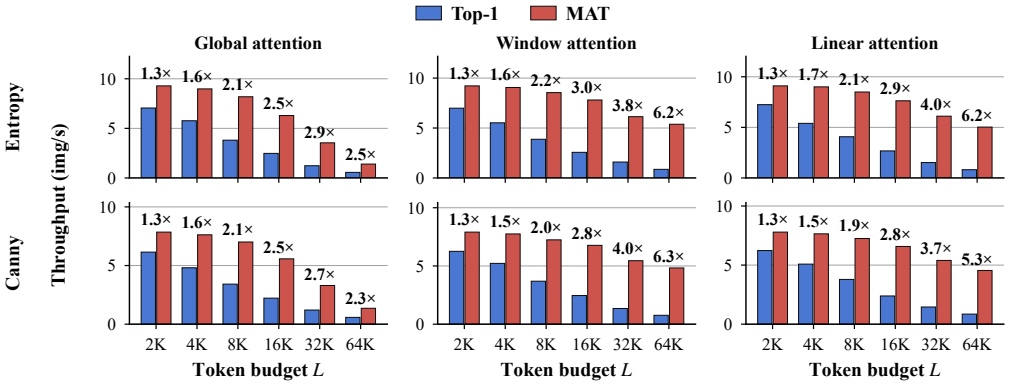

Figure 5: End-to-end training throughput at $8\text{K} \times 8\text{K}$ on a single GPU with batch size 4. Top-1 denotes the sequential refinement strategy of QuadFormer (Ronen et al., 2023). Numbers above the bars indicate MAT's speedup over Top-1.

higher-resolution weather targets, with importance scores derived from the aggregated multivariable representation.

4.2 END-TO-END TRAINING AND INFERENCE EFFICIENCY

Table 1 evaluates MAT across 2D images, 3D point clouds, and multichannel scientific data. On PAIP and S8D, MAT reproduces the allocation of Top-1 without its sequential loop, improving training and inference throughput with essentially unchanged downstream quality. On JpeakForest, implementation optimizations provide $1.25\times/1.29\times$ training/inference speedups, while octree-based MAT adds a further $1.18\times/2.06\times$ with $4.6\times$ lower training memory and comparable quality, outperforming APT in task quality at the same token budget (83.23 vs. 82.81). On ERA5, GPU batching accelerates Top-1 by $24.8\times/27.6\times$ over the original ORBIT-2 tokenizer, and MAT further improves throughput by $1.19\times/1.66\times$.

4.3 VISUALIZATION OF ADAPTIVE TOKENIZATION

Figure 3 shows token allocations produced by different refinement rules using the same importance map. At a matched budget, MAT reproduces the Top-1 partition exactly across all four datasets, including the fine-scale refinements highlighted in the magnified regions. This is consistent with our

Table 2: ImageNet-1K classification accuracy (%) (Deng et al., 2009) with ViT-B. *Original* uses iterative adaptive patching with the original scores; *Adapted* uses max-based score adaptation with MAT.

		Feature		Entropy		Pixel Blur	
	Budget	Original	Adapted	Original	Adapted	Original	Adapted
Pre-train	193	81.92	82.22	81.83	81.76	81.88	81.80
	127	81.45	81.42	80.42	80.79	80.17	80.19
	64	78.26	78.27	75.96	76.17	76.89	76.70
Fine-tune	196	84.08	84.05	83.89	83.97	83.86	83.84
	148	83.65	83.62	83.02	83.21	83.11	83.21
	100	82.84	82.76	81.75	81.84	82.09	81.99
	64	81.41	81.49	79.60	80.12	80.29	80.30

equivalence result: MAT changes how the split nodes are selected computationally, but not which nodes are selected. Batch-M deviates from this allocation by committing multiple refinements in each round before newly created children can compete. As a result, some coarse regions are refined earlier than under Top-1, causing large patches that Top-1 would otherwise preserve to be unnecessarily fragmented. APT exhibits a different behavior: refinement is controlled by a fixed threshold rather than an explicit token budget (Choudhury et al., 2026). Its output length therefore depends strongly on the threshold and the input, requiring threshold calibration and producing different token counts across datasets and samples. MAT avoids both effects, preserving Top-1's allocation and explicit budget without its sequential refinement loop.

4.4 PERFORMANCE SCALING OF MONOTONE ADAPTIVE TOKENIZATION

Figures 4 and 5 examine how the benefit of MAT scales with the token budget and attention backend on $8\text{K} \times 8\text{K}$ inputs, for two scoring metrics and six budgets. Top-1 requires $K = \lfloor (L-1)/3 \rfloor$ sequential rounds, so its tokenization cost grows with L; on PAIP and S8D it accounts for 53–80% of each training step, compared with 13–30% for MAT (Appendix A.3). With a single global Top-K, MAT's tokenization cost is largely insensitive to L; at inference, tokenization shrinks from a dominant to a small fraction of the wall time at large budgets, yielding speedups of up to $19.2\times$.

Training shows the same trend, although the backward pass reduces the relative share of tokenization. The speedup also depends on the attention backend, reaching up to $2.9\times$, $6.3\times$, and $6.2\times$ with global (Vaswani et al., 2017; Dao et al., 2022), window (Liu et al., 2021), and linear attention (Katharopoulos et al., 2020), respectively, as cheaper model computation makes tokenization a larger fraction of each step.

4.5 EFFECT OF SCORE ADAPTATION

Table 2 evaluates score adaptation based on three non-monotone importance metrics: feature saliency (Ronen et al., 2023), color entropy (Choudhury et al., 2026), and pixel blur (Ronen et al., 2023). Across all three metrics, MAT with adapted scores achieves performance comparable to iterative adaptive patching with the original scores, in both pre-trained and fine-tuned settings. This shows that score adaptation extends MAT to non-monotone importance criteria without noticeably compromising downstream accuracy.

5 CONCLUSION

We introduced Monotone Adaptive Tokenization (MAT) for adaptive tokenization over a hierarchy. Building on the heap order property of monotone scores, MAT turns sequential Top-1 refinement into one-shot global Top-K selection with GPU-friendly operations, while supporting diverse importance metrics through simple score adaptation and general rooted hierarchies. Across 2D images, 3D point clouds, and weather data, MAT consistently accelerates training and inference while preserving downstream quality, with speedups reaching $6.3\times$ in training and $19.2\times$ in inference across the tested budgets and attention backends. By removing sequential refinement as a bottleneck, MAT makes adaptive tokenization practical where it is most needed and turns it into a scalable representation paradigm for the next generation of high-resolution and high-dimensional data.

AI USE STATEMENT

In this work, we used generative AI tools for code generation, for running experiments and organizing experimental data (e.g., collecting metrics from experiment logs into tables), for literature search, for drafting parts of the paper, and as an editorial aid for writing and grammar. We have not used generative AI tools for research ideation: the research questions, the experimental design, the analyses and all claims are the authors' own. We have reviewed all AI-assisted work; generated code was reviewed by the authors and its outputs verified against the results reported here, references found with AI assistance were checked against the original publications, and all AI-assisted text was checked and revised by the authors. We take full responsibility for the final content of this work.

REFERENCES

Marsha J. Berger and Phillip Colella. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics*, 82(1):64–84, 1989. doi: 10.1016/0021-9991(89)90035-1.

Marsha J. Berger and Joseph Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *Journal of Computational Physics*, 53(3):484–512, 1984. doi: 10.1016/0021-9991(84)90073-1.

Kaifeng Bi, Lingxi Xie, Hengheng Zhang, Xin Chen, Xiaotao Gu, and Qi Tian. Accurate medium-range global weather forecasting with 3D neural networks. *Nature*, 619(7970):533–538, 2023. doi: 10.1038/s41586-023-06185-3.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Sander Borst, Daniel Dadush, Sophie Huiberts, and Danish Kashaev. A nearly optimal randomized algorithm for explorable heap selection. *Mathematical Programming*, 210:75–96, 2025. doi: 10.1007/s10107-024-02145-5.

Neil Burgess, Jelena Milanovic, Nigel Stephens, Konstantinos Monachopoulos, and David Mansell. Bfloat16 processing for neural networks. In *Proceedings of the IEEE Symposium on Computer Arithmetic (ARITH)*, pp. 88–91, 2019.

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.

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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 16144–16155, 2022.

Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, László A. Jeni, and Kris M. Kitani. Accelerating vision transformers with adaptive patch sizes. In *International Conference on Learning Representations (ICLR)*, 2026.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 16344–16359, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations (ICLR)*, 2021.

Mark Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. ROAMing terrain: Real-time optimally adapting meshes. In *Proceedings Visualization '97*, pp. 81–88. IEEE, 1997. doi: 10.1109/VISUAL.1997.663860.

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 (ICCV)*, pp. 6824–6835, 2021.

Mohsen Fayyaz, Soroush Abbasi Koohpayegani, Farnoush Rezaei Jafari, Sunando Sengupta, Hamid Reza Vaezi Joze, Eric Sommerlade, Hamed Pirsiavash, and Jürgen Gall. Adaptive token sampling for efficient vision transformers. In *European Conference on Computer Vision (ECCV)*, volume 13671 of *Lecture Notes in Computer Science*, pp. 396–414. Springer, 2022. doi: 10.1007/978-3-031-20083-0_24.

Raphael A. Finkel and Jon Louis Bentley. Quad trees: A data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974. doi: 10.1007/BF00288933.

Greg N. Frederickson. An optimal algorithm for selection in a min-heap. *Information and Computation*, 104(2):197–214, 1993. doi: 10.1006/inco.1993.1030.

Irene Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12): 905–910, 1982. doi: 10.1145/358728.358741.

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 (WACV)*, pp. 574–584, 2022.

Jakob Drachmann Havtorn, Amélie Royer, Tijmen Blankevoort, and Babak Ehteshami Bejnordi. MSViT: Dynamic mixed-scale tokenization for vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 838–848, 2023. doi: 10.1109/ICCVW60793.2023.00091.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, et al. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146 (730):1999–2049, 2020. doi: 10.1002/qj.3803.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021. doi: 10.1109/TBDATA.2019.2921572.

Haim Kaplan, László Kozma, Or Zamir, and Uri Zwick. Selection from heaps, row-sorted matrices, and X+Y using soft heaps. In *2nd Symposium on Simplicity in Algorithms (SOSA 2019)*, volume 69 of *Open Access Series in Informatics*, pp. 5:1–5:21, 2019. doi: 10.4230/OASIcs.SOSA.2019.5.

Brian Karis, Rune Stubbe, and Graham Wihlidal. A deep dive into Nanite virtualized geometry. ACM SIGGRAPH 2021, Advances in Real-Time Rendering in Games Course, 2021.

Richard M. Karp, Michael E. Saks, and Avi Wigderson. On a search problem related to branch-and-bound procedures. In *27th Annual Symposium on Foundations of Computer Science*, pp. 19–28. IEEE, 1986. doi: 10.1109/SFCS.1986.34.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165, 2020.

Yoo Jung Kim, Hyungjoon Jang, Kyoungbun Lee, Seongkeun Park, Sung-Gyu Min, Choyeon Hong, et al. PAIP 2019: Liver cancer segmentation challenge. *Medical Image Analysis*, 67:101854, 2021. doi: 10.1016/j.media.2020.101854.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, 2023. doi: 10.1109/ICCV51070.2023.00371.

Youwei Liang, Chongjian Ge, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations (ICLR)*, 2022.

Peter Lindstrom and Valerio Pascucci. Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):239–254, 2002. doi: 10.1109/TVCG.2002.1021577.

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 (ICCV)*, pp. 10012–10022, 2021.

Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. In *International Conference on Learning Representations (ICLR)*, 2017.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

Donald Meagher. Geometric modeling using octree encoding. *Computer Graphics and Image Processing*, 19(2):129–147, 1982. doi: 10.1016/0146-664X(82)90104-6.

Guy M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. Technical report, International Business Machines Company, Ottawa, Canada, 1966.

PAIP 2019 Challenge. PAIP 2019: Liver cancer segmentation challenge – dataset. `https://paip2019.grand-challenge.org/Dataset/`, 2019. Accessed: September 2026.

Renato Pajarola. Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings Visualization '98*, pp. 19–26. IEEE, 1998. doi: 10.1109/VISUAL.1998.745280.

Stefano Puliti, Grant Pearse, Peter Surový, Luke Wallace, Markus Hollaus, Maciej Wielgosz, and Rasmus Astrup. FOR-instance: A UAV laser scanning benchmark dataset for semantic and instance segmentation of individual trees. arXiv preprint arXiv:2309.01279, 2023.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 13937–13949, 2021.

Tomer Ronen, Omer Levy, and Avram Golbert. Vision transformers with mixed-resolution tokenization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 4613–4622, 2023.

Michael S. Ryoo, A. J. Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. TokenLearner: Adaptive space-time tokenization for videos. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, pp. 12786–12797, 2021.

Hanan Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys*, 16 (2):187–260, 1984. doi: 10.1145/356924.356930.

Anil Shanbhag, Holger Pirk, and Samuel Madden. Efficient Top-K query processing on massively parallel hardware. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pp. 1557–1570, 2018. doi: 10.1145/3183713.3183735.

SPring-8. SPring-8: Large synchrotron radiation facility. `https://www.spring8.or.jp/en/`, 2026. Accessed: September 2026.

Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. QuadTree attention for vision transformers. In *International Conference on Learning Representations (ICLR)*, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pp. 5998–6008, 2017.

Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 568–578, 2021.

Xiao Wang, Jong-Youl Choi, Takuya Kurihaya, Isaac Lyngaas, Hong-Jun Yoon, Xi Xiao, David Pugmire, Ming Fan, Nasik Muhammad Nafi, Aristeidis Tsaris, Ashwin M. Aji, Maliha Hossain, Mohamed Wahib, Dali Wang, Peter Thornton, Prasanna Balaprakash, Moetasim Ashfaq, and Dan Lu. ORBIT-2: Scaling exascale vision foundation models for weather and climate downscaling. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 86–98. ACM, 2025. doi: 10.1145/3712285.3771989.

Binbin Xiang, Maciej Wielgosz, Stefano Puliti, Kamil Král, Martin Krůček, Azim Missarov, and Rasmus Astrup. ForestFormer3D: A unified framework for end-to-end segmentation of forest LiDAR 3D point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 24717–24727, 2025.

Hongxu Yin, Arash Vahdat, Jose M. Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10809–10818, 2022.

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 *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pp. 1–16. IEEE, 2024. doi: 10.1109/SC41406.2024.00082.

Enzhi Zhang, Peng Chen, Rui Zhong, Du Wu, Jun Igarashi, Isaac Lyngaas, Xiao Wang, Masaharu Munetomo, and Mohamed Wahib. SHF: Symmetrical hierarchical forest with pretrained vision transformer encoder for high-resolution medical segmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025a.

Jingrong Zhang, Akira Naruse, Xipeng Li, and Yong Wang. Parallel Top-K algorithms on GPU: A comprehensive study and new methods. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC '23)*. ACM, 2023. doi: 10.1145/3581784.3607062.

Zhengqiang Zhang, Rongyuan Wu, Lingchen Sun, and Lei Zhang. GPSToken: Gaussian parameterized spatially-adaptive tokenization for image representation and generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 38, 2025b.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 16259–16268, 2021. doi: 10.1109/ICCV48922.2021.01595.

A APPENDIX

A.1 ADDITIONAL DETAILS OF MONOTONE ADAPTIVE TOKENIZATION

This section complements Sections 3.1–3.4 with the connection to heap selection and multiresolution rendering, a detailed proof of the exact reformulation, general hierarchies and token counts, properties of score adaptation, and parallel execution.

A.1.1 CONNECTION TO HEAP SELECTION AND MULTIRESOLUTION RENDERING

Under equation 2, the hierarchy $\mathcal{T}$ satisfies the heap order property with respect to C. Sequential Top-1 refinement then coincides with the best-first procedure of heap selection: it keeps the frontier $F(S)$ in a priority queue, repeatedly extracts the highest-scoring region, and inserts its refinable children (Kaplan et al., 2019). Classical work improves the $O(K \log K)$ cost of this procedure for bounded branching factors to time linear in K (Frederickson, 1993; Kaplan et al., 2019). MAT does not rely on these algorithms; it uses only the fact that their output is the set of K highest-ranked regions (Lemma 1 and Proposition 1).

Max reduction corresponds to error saturation in multiresolution rendering, where parent errors are raised to bound those of their descendants so that refinement decisions respect hierarchical dependencies (Pajarola, 1998; Lindstrom & Pascucci, 2002; Karis et al., 2021). The dependency structures differ, however: crack-free meshes impose additional dependencies between neighboring regions, so errors are propagated over a dependency graph rather than a tree, whereas adaptive tokenization involves only ancestor precedence. Sum reduction has no direct counterpart in this setting; it arises naturally for nonnegative additive quantities such as edge counts.

A.1.2 EXACT EQUIVALENCE AND TIE BREAKING

We restate the ordering used in Section 3.2 and give the proof of Proposition 1 in more detail. Let $\succ$ be a deterministic total order on $\mathcal{T}$ that sorts regions by decreasing score, places ancestors before descendants among equal scores, and resolves remaining ties by a fixed node identifier. Thus,

$$C(a) > C(r) \implies a \succ r, \qquad C(a) = C(r),\ a \in \mathrm{Anc}(r) \implies a \succ r.$$

Sequential Top-1 refinement breaks ties on the frontier by the same order.

Lemma 1 (Top-K closure). Under equation 2, every prefix of $\succ$ is ancestor closed.

Proof. Let r lie in the first K positions of $\succ$ and let $a \in \mathrm{Anc}(r)$. By equation 2, $C(a) \geq C(r)$. If the inequality is strict, a precedes r by score; if the scores are equal, the tie-breaking rule places a before r. Hence every ancestor of r also lies in the first K positions. □

Proposition 1 (Exact equivalence of greedy refinement and global selection). Under equation 2 and the ordering above, K sequential Top-1 refinements select exactly the first K regions of $\succ$.

Proof. Let $r_1 \succ r_2 \succ \cdots$ denote the global ordering. We show by induction that after k steps the selected set is $S_k = \{r_1, \ldots, r_k\}$. The claim holds trivially for $k = 0$. Assume it holds after k steps. By Lemma 1, the prefix $\{r_1, \ldots, r_{k+1}\}$ is ancestor closed, so every ancestor of r_{k+1} belongs to S_k and $r_{k+1} \in F(S_k)$. If another region $u \in F(S_k)$ satisfied $u \succ r_{k+1}$, it would be among the first k regions and hence in S_k, a contradiction. Thus r_{k+1} is the highest-ranked region on the frontier and is selected at step $k + 1$. □

Therefore, $K \times \text{Top-1} \equiv \text{Global Top-K}$. The result concerns the selected refinement set, not merely the token count: both procedures refine exactly the same regions and therefore produce the same adaptive partition.

A.1.3 General Hierarchies and Token Budgets

The reformulation does not depend on the branching factor. Let $b(r) = |\operatorname{Child}(r)|$. Refining r removes one token and introduces $b(r)$ new ones, so starting from R initial regions, a refinement set S produces

$$L = R + \sum_{r \in S} \big(b(r) - 1\big)$$

tokens. For a fixed branching factor b and $K = |S|$, this reduces to $L = R + (b-1)K$, i.e., $L = R + 3K$ for quadtrees and $L = R + 7K$ for octrees.

Proposition 1 depends only on the ancestor relation and equation 2, not on the branching factor or spatial dimensionality, so it holds for general rooted hierarchies. With $R > 1$ initial regions, $\mathcal{T}$ is a forest of R trees, to which Lemma 1 and Proposition 1 apply unchanged. With variable branching factors, global Top-K still reproduces K sequential refinement decisions exactly, although K alone no longer determines L.

A.1.4 Properties of Score Adaptation

Let $c(r)$ denote the base score at the finest level. We initialize $C_{\text{sum}}(r) = C_{\max}(r) = c(r)$ at this level and, for every internal region r, define

$$C_{\text{sum}}(r) = \sum_{u \in \operatorname{Child}(r)} C_{\text{sum}}(u), \qquad C_{\max}(r) = \max_{u \in \operatorname{Child}(r)} C_{\max}(u).$$

Sum reduction. For nonnegative base scores, $C_{\text{sum}}(r)$ equals the total base score of the finest-level regions contained in r, so

$$C_{\text{sum}}(r) \geq C_{\text{sum}}(u), \qquad \forall u \in \operatorname{Child}(r).$$

By transitivity, equation 2 holds between every ancestor and descendant.

Max reduction. By construction,

$$C_{\max}(r) \geq C_{\max}(u), \qquad \forall u \in \operatorname{Child}(r),$$

so max reduction is likewise monotone along the full hierarchy.

Both reductions retain the spatial localization of the base metric. Proposition 1 applies to the adapted scores: MAT exactly matches sequential Top-1 refinement under the same adapted scores, but need not reproduce the ordering induced by the original non-monotone scores.

A.1.5 Parallel Execution and Implementation

The computational advantage of MAT comes from removing the dependency chain induced by the evolving frontier.

The dependency depth of sequential refinement is analyzed in Section 3.4. In MAT, scores for naturally monotone metrics are computed independently for all candidate regions, and adapted scores are obtained by a bottom-up reduction over the D levels of the hierarchy, parallel within each level. For N candidate regions, score construction requires $O(N)$ work, and D is typically much smaller than K. A single parallel Top-K then determines all refinement decisions, and the selected regions are extracted independently. The critical path is therefore set by the hierarchy depth and global selection rather than by K serial refinement steps.

GPU representation. We store candidate regions in contiguous arrays organized by hierarchy level and spatial order. Parent–child relationships follow from deterministic index mappings, avoiding pointer-based tree traversal. For 2D and 3D inputs, spatial indices can be represented in Morton order (Morton, 1966; Gargantini, 1982), which preserves locality while providing efficient parent–child indexing.

Scores are constructed in parallel within each level; for adapted metrics, sum or max reductions proceed from fine to coarse levels. The resulting scores are passed to a GPU Top-K primitive to

Algorithm 1: Monotone Adaptive Tokenization

Require: hierarchy $\mathcal{T}$, importance metric ϕ, refinement budget K
1: Compute importance scores with ϕ
2: **if** scores are not monotone along the hierarchy **then**
3: Adapt scores by bottom-up sum or max reduction
4: **end if**
5: $S \leftarrow \mathrm{GlobalTopK}(\mathcal{T}, K)$
6: Extract the tokens induced by S in parallel
7: Resize, order, and embed the tokens
8: **return** adaptive token sequence

determine the refinement set. Because Lemma 1 guarantees that the selected set is ancestor closed, no post-selection feasibility correction or frontier reconstruction is required. The selected refinement decisions are then converted into adaptive regions in parallel, and patch extraction, resizing, spatial ordering, and embedding proceed without the sequential refinement loop of Top-1 methods. Our implementation fuses lightweight hierarchy operations where possible and uses standard parallel selection primitives for global Top-K (Shanbhag et al., 2018; Johnson et al., 2021).

A.2 IMPLEMENTATION DETAILS

Baseline implementation. We follow the baseline definitions in the main text. Since APT (Choudhury et al., 2026) does not enforce an exact token budget, we calibrate its threshold to approximately match the target token count and report the measured mean. DynamicViT (Rao et al., 2021) and ToMe (Bolya et al., 2023) are configured to match MAT's mean sequence length. Top-1 (Ronen et al., 2023) and MAT share the same GPU tokenization pipeline and differ only in the rule for selecting split nodes.

Timing protocol. All throughput, latency, and memory measurements use a single NVIDIA GB200 GPU, and peak memory is the maximum tensor memory allocated by PyTorch. For PAIP and S8D, training and inference use a batch size of 4 with 16 data-loading workers; training runs in BF16 mixed precision, and inference runs the forward pass in FP32. We discard the first 40 steps as warm-up and report the mean over the next 30 steps, measured without additional synchronization so that data loading, host-to-device transfer, and computation overlap as in normal training. Each timed step spans fetching a batch through the optimizer update (or through the forward pass at inference), and thus includes the tokenization stage and the model. The tokenization stage comprises host-to-device transfer, training-time augmentation, edge detection, cost-pyramid construction, selection, leaf generation, and resampling; for PAIP it also includes JPEG decoding on the GPU, whereas S8D images are decoded by the data-loading workers outside this stage. The per-stage decomposition in Table 3 is measured in a separate pass that synchronizes at every stage boundary. Figures 4 and 5 are separate scaling experiments on $8\mathrm{K} \times 8\mathrm{K}$ inputs, with a batch size of 1 for inference and 4 for training. For JpeakForest, training uses a batch size of 2; each variant is trained for two epochs, and we report the second epoch, treating the first as warm-up. Inference is timed scene by scene with CUDA synchronization around the complete prediction call, after the points have been loaded and transferred to the GPU, so it includes voxelization, octree construction and selection, the model forward pass, and post-processing. For ERA5, training and inference use a batch size of 32. Training throughput is measured over the full 100-epoch run after 10 untimed warm-up steps, and includes data loading, normalization, host-to-device transfer, tokenization, and the model update. Inference runs 3 untimed warm-up batches and then times every batch of the test split between CUDA synchronizations, starting once the batch is resident on the GPU; it includes tokenization and the model forward pass, but excludes data loading and host-to-device transfer.

2D experiments. PAIP and S8D use UNETR (Hatamizadeh et al., 2022) with a ViT-B backbone (Dosovitskiy et al., 2021). PAIP is split by slide into 40 training slides (3,061 crops) and 10 validation slides (778 crops). S8D is split by volume into 34 training volumes (2,430 slices) and 9 validation volumes (650 slices). PAIP uses 32×32 base patches with a 16×16 minimum leaf, whereas S8D uses 64×64 for both; both target $L = 8{,}191$ tokens. Models are trained for 300

epochs with AdamW (Loshchilov & Hutter, 2019), an initial learning rate of 10^{-4}, weight decay 0.05, cosine decay (Loshchilov & Hutter, 2017) to 10^{-6}, a global batch size of 16, and BF16 precision (Burgess et al., 2019). S8D inputs are 16-bit single-channel images; we retain the high byte and replicate it to three channels. For both datasets, Canny edge detection uses a Gaussian blur with kernel size 5 and hysteresis thresholds of $100/200$, and hierarchical scores are obtained by summing edge counts bottom-up.

3D forest experiments. We use ForestFormer3D (Xiang et al., 2025) and apply MAT to its decoder representation through an octree. The model is trained on the FOR-instanceV2 dataset (Puliti et al., 2023), which contains approximately 600 million labeled points, and evaluated on the substantially larger JpeakForest dataset with 7.0 billion points. Each cell is scored by the mean channel variance of its features. These scores are made monotone along the octree with max reduction (Section 3.3). Training runs in FP32 for all variants. At inference, Vanilla follows the FP32 path of the released code and processes one region per forward pass, whereas Uniform Voxel and the adaptive variants use FP16 mixed precision with an FP16 decoder and process eight regions per forward pass. The gap between Vanilla and Uniform Voxel therefore includes these precision and batching differences, while Uniform Voxel and the adaptive variants share the same implementation and differ only in tokenization.

ERA5 experiments. We use the Res-Slim-ViT architecture from ORBIT-2 (Wang et al., 2025). The original ORBIT-2 tokenizer produces 127 tokens, whereas our two-root quadtree produces 128. Importance scores are computed from the aggregated variable embedding after per-sample min-max normalization. All variants are trained for 100 epochs with AdamW, a learning rate of 2×10^{-3}, and a batch size of 32. Model weights and inputs are held in BF16 for both training and inference. We run three seeds with matched initialization across methods and report the mean accuracy.

ImageNet experiments. Table 2 uses ViT-B (Dosovitskiy et al., 2021) at 256×256 input resolution, with adaptive patches of 16×16 to 64×64 pixels resampled to 16×16. *Original* and *Adapted* differ only in the selection rule: sequential Top-1 on the original scores versus MAT on the max-adapted scores. The pre-training rows are trained from random initialization for 300 epochs with a global batch size of 1,024 on four GPUs, AdamW with learning rate 10^{-3}, weight decay 0.05 and $\beta = (0.9, 0.999)$, 5 warm-up epochs followed by cosine decay to 10^{-6}, drop path 0.1, and RandAugment (`rand-m9-mstd0.5-inc1`). The fine-tuning rows start from a dense ViT-B that is initialized from ImageNet-21k weights and trained for 70 epochs on the uniform 256×256 grid, and are fine-tuned for 50 epochs with a global batch size of 4,096 (four GPUs, 256 images per GPU, and four accumulation steps), learning rate 3×10^{-4}, weight decay 0.05, 5 warm-up epochs followed by cosine decay to 3×10^{-6}, drop path 0.2, and AutoAugment. Both settings use random-resized crops (scale 0.08–1, bicubic) with horizontal flips, Mixup ($\alpha = 0.8$) and CutMix ($\alpha = 1.0$), label smoothing 0.1, gradient clipping at 1.0, BF16 mixed precision, and an exponential moving average of the weights (decay 0.99996 for pre-training and 0.9998 for fine-tuning). Evaluation resizes the shorter side to 292 pixels and takes a 256×256 center crop; we report the best top-1 accuracy of the averaged weights over training.

A.3 TOKENIZATION COST OF SEQUENTIAL TOP-1

Table 3 decomposes the training step on a single GB200 GPU with batch size 4. Top-1 tokenization takes 53–80% of each training step (51–75% at inference), longer than the rest of the step combined; since it runs on the GPU, it lies on the critical path of every step. MAT reduces this share to 13–30%, and its remaining cost is dominated by edge detection.

Table 4 isolates the selection stage on the PAIP setting with 21,845 internal nodes. Top-1 time is proportional to K at about $183\,\mu$s per round, whereas one-shot selection stays at 0.46 ms, and both select identical sets at every budget, as guaranteed by Proposition 1. Nsight Systems profiling at $L = 8{,}191$ shows that each round issues 14 kernels with a median execution time of $1.34\,\mu$s, keeping the GPU busy for only $37.5\,\mu$s per round, with no device synchronization or memory copy in the loop; the remaining time is spent on CPU-side dispatch and kernel launch. Eliminating launch overhead, e.g., with CUDA Graphs, would still leave about 102 ms of GPU time at $L = 8{,}191$, compared with 0.15 ms for one-shot selection, and a persistent single-kernel implementation would still require K dependent rounds, each with a global reduction and synchronization.

Table 3: Training-step decomposition. The tokenization stage is defined in the timing protocol of Appendix A.2.

			Top-1			MAT	
Dataset	L	K	Step (ms)	Tok. (ms)	Share	Tok. (ms)	Share
PAIP 4K	4,096	1,365	450	326	73%	47	30%
	8,191	2,730	733	590	80%	46	23%
	16,384	5,461	1,823	1,256	69%	48	13%
S8D 8K	4,096	1,365	777	410	53%	136	26%
	8,191	2,730	1,053	636	60%	131	24%
	16,384	5,461	1,608	1,047	65%	135	20%

Table 4: Selection-stage cost of sequential Top-1 and one-shot selection on PAIP.

L	K	Top-1 (ms)	Per round (μs)	One-shot (ms)	Identical sets
4,096	1,365	252.8	185	0.46	✓
8,191	2,730	501.7	184	0.46	✓
16,384	5,461	996.3	182	0.45	✓

A.4 LAYOUTS PRODUCED BY ADAPTED SCORES

Figure 6 shows the patch layouts that the original and max-adapted scores produce at matched budgets for feature saliency, color entropy, and pixel blur. The layouts are close enough that the difference is hard to see without overlaying them, which is the qualitative counterpart of the accuracy agreement reported in Table 2.

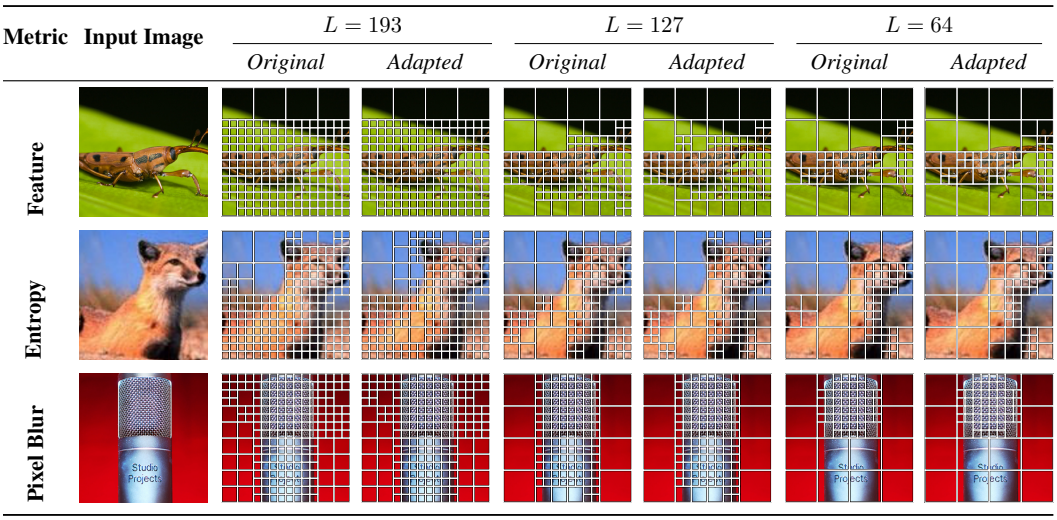

Figure 6: Visual comparison before and after monotone score adaptation. *Original* shows the adaptive patch layout produced by the original non-monotone scoring metric, while *Adapted* shows the layout produced after applying our monotone score adaptation.