QUADTOK: QUADTREE VISUAL TOKENIZER FOR AUTOREGRESSIVE IMAGE GENERATION

Anonymous authors
Paper under double-blind review

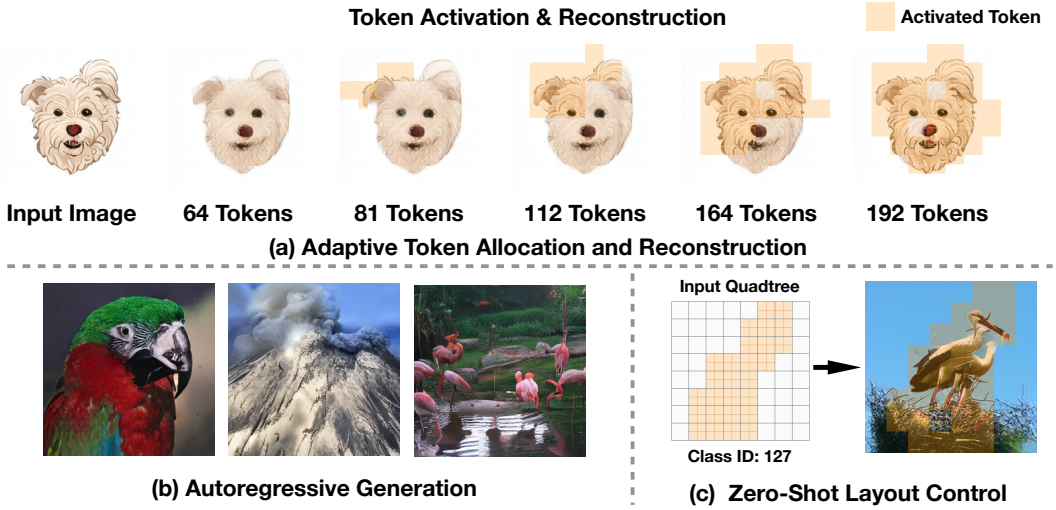


Figure 1: **Adaptive token allocation with QuadTok**: (a) Progressive reconstruction using the QuadTok tokenizer: starting from a coarse reconstruction with 64 tokens, additional tokens progressively improve reconstruction quality. (b) Autoregressive image generation with the QuadTok generator: tokens are generated following the quadtree topology, enabling coarse-to-fine synthesis. (c) Zero-shot spatial layout controlled generation: a user or LLM-specified spatial layout is translated into a quadtree, and the QuadTok generator predicts visual codes conditioned on this topology.

ABSTRACT

We introduce QuadTok, a novel framework for visual tokenization and autoregressive image generation. Compared to traditional approaches using 2D grids or 1D token sequences, we propose a hierarchical quadtree structure, bridging the gap between 2D spatial binding and 1D sequence-level flexibility. The QuadTok tokenizer dynamically allocates representational capacity to visually intricate areas while leaving homogeneous regions at a coarse resolution. Compared with a fixed 256-token grid, our ImageNet-trained tokenizer saves approximately **10%** of tokens on ImageNet and **9%** when transferred zero-shot to COCO dataset, while maintaining comparable reconstruction fidelity. Furthermore, the natural causality introduced by the tree structure seamlessly enables autoregressive image generation. Conditioned on a quadtree topology supplied before generation, our 947M GPT-style generative model achieves a **2.08** gFID on the ImageNet 256×256 benchmark. Additionally, leveraging the strong spatial correlation preserved by the quadtree structure, the QuadTok generator enables zero-shot spatially controlled image generation capabilities.

1 INTRODUCTION

Natural images exhibit highly non-uniform spatial information density: regions containing intricate textures and high-frequency patterns carry significantly more information than smooth or homogeneous areas. Ideally, a visual tokenizer should allocate representational capacity according to this spatial variability. However, classic visual tokenizers (van den Oord et al., 2018; Sun et al., 2024) compress images into rigid 2D grids with a fixed token budget, distributing tokens uniformly across

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.

the image regardless of local complexity. As a result, complex regions are often under-represented while simple regions are over-allocated with redundant tokens.

To overcome the rigidity of grid-based tokenization, recent work has explored 1D visual tokenizers (Shen et al., 2025; Yu et al., 2024b; Duggal et al., 2024; Miwa et al., 2025b; Wang et al., 2026; Bachmann et al., 2025b; Chen et al., 2026). These methods model images as sequential token streams and allow for variable length, offering greater flexibility in adapting the token budget to image complexity. Nevertheless, this flexibility comes at the cost of spatial awareness. Because token allocation occurs along the sequence dimension rather than the image plane, these approaches remain largely spatially agnostic. Consequently, they often lack explicit correspondence between tokens and image patches, which can degrade pixel-level fidelity and make them less suitable for downstream tasks that require spatially grounded representations and precise localization.

In this work, we propose *QuadTok*, a novel quadtree-based visual tokenizer that enables adaptive token allocation while preserving explicit spatial correspondence. Instead of relying on a rigid grid or a purely sequential representation, QuadTok represents images using a hierarchical quadtree structure that dynamically subdivides spatial regions based on local complexity. This design bridges the gap between existing paradigms: it retains the spatial grounding of traditional 2D tokenizers while introducing the sequence-level flexibility of 1D token representations. However, one key challenge remains: *How can we select which regions to refine based on their reconstruction benefit?* To tackle this, we propose *Region-Wise Complexity Guidance*, a data-driven strategy to select a content-adaptive tree for each image based on reconstruction benefit. Furthermore, we identify that maintaining the hierarchical causality and explicit spatial relationships between tree nodes is crucial during the tokenization process. To enforce this structural integrity, we introduce the *Kinship Causal Mask*. This mechanism dictates a strict coarse-to-fine information flow along the quadtree hierarchy, allowing the global image structure to be established before localized refinement occurs within spatially bounded regions.

Empowered by our *Region-Wise Complexity Guidance*, the QuadTok tokenizer achieves an rFID of 1.46 and a PSNR of 20.37 on ImageNet with 230 tokens on average, saving approximately 10% relative to a fixed 256-token grid. Transferred zero-shot to COCO, the same tokenizer uses 232 tokens on average, saving approximately 9%. Furthermore, leveraging the causality inherent in the quadtree structure, the QuadTok 947M autoregressive generator achieves a gFID of 2.08, compared with 2.15 for the 1.4B RandAR-XXL model (Pang et al., 2025). We also show a consistent scaling behavior on our generator across different model scales. Furthermore, guided by the explicit quadtree topology, QuadTok generator enables novel zero-shot capabilities such as spatially controllable generation.

Our contributions are summarized as follows:

- We propose *QuadTok*, a quadtree-based visual tokenizer that replaces fixed-grid representations with spatially adaptive token allocation. Our approach preserves explicit spatial correspondence while enabling flexible token budgets driven by local visual complexity.
- We introduce two key mechanisms to support adaptive tokenization: (i) *Region-Wise Complexity Guidance* strategy selecting regional refinement from reconstruction benefit; (ii) a *Kinship Causal Mask* enforcing hierarchical dependencies among tree nodes during tokenization.
- Leveraging the hierarchical causality of quadtrees, QuadTok achieves competitive image generation quality while enabling zero-shot spatially controllable image synthesis conditioned on quadtree layouts supplied before generation.

2 Related Work

2.1 Visual Tokenization

Visual tokenizers compress images into representations for generative modeling. VQ-VAE (van den Oord et al., 2018) learns discrete codebooks, and VQ-GAN (Esser et al., 2021) adds adversarial training to improve perceptual reconstruction. Later work improves codebook design and tokenizer architectures (Lee et al., 2022; Mentzer et al., 2024; Yu et al., 2024a; Zhao et al., 2024). Beyond fixed spatial grids, TiTok (Yu et al., 2024b) represents images as compact 1D sequences. FlexTok (Bachmann et al., 2025a) and One-D-Piece (Miwa et al., 2025b) support variable sequence lengths, while GigaTok (Xiong et al., 2025) uses semantic regularization to improve reconstruction and downstream generation as tokenizers scale. QuadTok focuses on assigning discrete tokens to regions within an image while retaining an explicit spatial correspondence.

Content-adaptive tokenization. CAT (Shen et al., 2025) adapts representation capacity to image complexity by using caption-derived scores to select an image-wide compression ratio in a nested continuous VAE. Its variable-length latents support image synthesis with a diffusion transformer. QuadTok instead allocates discrete tokens at different spatial resolutions within each image, preserving explicit correspondence between tokens and regions. During reconstruction, image content determines which regions to refine; during generation, an autoregressive model predicts visual codes under a topology supplied before sampling. We compare with CAT's published ImageNet 512×512 generation results in Tab. 7, alongside other high-resolution generators.

2.2 Autoregressive Image Generation

Autoregressive image models differ in both their latent representation and prediction unit (Lee et al., 2022; Li et al., 2024; Shan et al., 2026). LlamaGen (Sun et al., 2024) predicts discrete grid tokens sequentially, RandAR (Pang et al., 2025) changes their generation order, and VAR (Tian et al., 2024) predicts successive resolution scales. DPAR (Srivastava et al., 2025) uses an entropy model to group discrete tokens into variable-length patches, reducing the sequence processed by its global transformer. We compare with DPAR as a related approach to spatial token aggregation. QuadTok represents regions directly as quadtree nodes and generates their discrete codes in breadth-first order conditioned on the given tree.

Continuous-token methods provide another approach to autoregressive synthesis. xAR (Ren et al., 2025a) generalizes the prediction unit to entities such as cells and uses flow matching with noisy contexts. FlowAR (Ren et al., 2025b) combines scale-wise autoregression with flow matching. QuadTok models variable-length discrete sequences with explicit regional correspondence, allowing generation to be conditioned on a supplied spatial topology.

2.3 Quadtree Representations in Vision

Quadtrees recursively partition two-dimensional space and have been used for image processing, compression, and adaptive visual computation (Wu, 2018; Tang et al., 2022; Ronen et al., 2023; Zhao et al., 2025). QuadTok incorporates this hierarchy into a generative visual tokenizer. Its topology-aware attention preserves structured interactions among tokens, and its discrete representation supports image reconstruction and autoregressive synthesis under a supplied spatial topology.

3 Method

3.1 QuadTok Tokenizer

We introduce QuadTok, a visual tokenizer that represents images using an adaptive quadtree structure. Unlike conventional methods that rely on fixed grids, QuadTok dynamically allocates tokens across regions while maintaining explicit correspondence between tokens and the image patches.

3.1.1 Quadtree Representation

QuadTok represents an image using a hierarchical quadtree decomposition, where each node corresponds to a spatial region and is treated as a visual token. A node is defined by the attribute tuple (l, p), where l denotes the tree level and p the spatial index. Level $l = 1$ represents coarse regions of the image, while level $l = 2$ corresponds to higher-resolution subdivisions. Each parent node may expand into four child nodes covering its spatial quadrants. Our default configuration uses two levels; we also evaluate a three-level extension in Secs. 4.1 and 4.2.

To interface this hierarchical structure with Transformer architecture, the quadtree is serialized into a 1D sequence using breadth-first search (BFS) traversal. This ordering first lists all coarse tokens and then their corresponding higher-resolution child tokens, naturally aligning with coarse-to-fine generation.

3.1.2 Image Encoding

Given an input image I, we first extract dense visual features using a Vision Transformer(ViT) (Dosovitskiy et al., 2021) encoder $F = \mathrm{Encoder}(I)$ where $F \in \mathbb{R}^{N \times D}$ denotes grid-based patch features.

Each quadtree node is represented by a *learnable instruction token* that encodes its spatial location and tree level. For node i with tree attributes (l_i, p_i), the instruction embedding is $q_i = \mathrm{Emb}_{l_i}(p_i)$

The sequence of instruction tokens $Q = [q_1, \cdots, q_M]$ is concatenated with the dense image features and processed by a Transformer module named *Aggregator*: $[F' \parallel Z] = \mathrm{Aggregator}([F \parallel Q])$

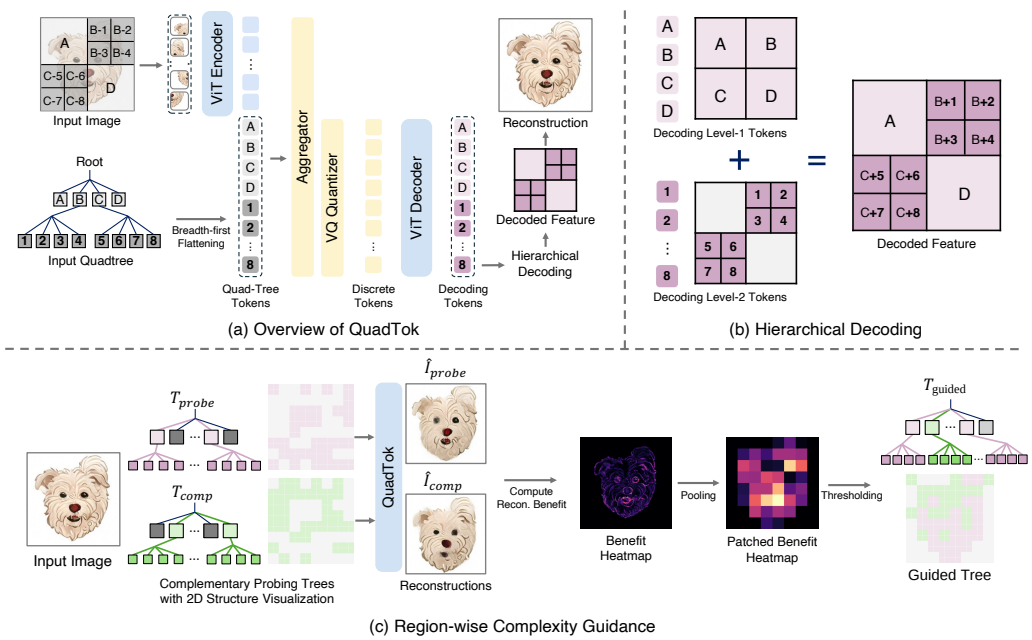


Figure 2: **(a) Overview of QuadTok.** An image is serialized into a 1D token sequence from a quadtree via breadth-first traversal. ViT features are aggregated into quadtree tokens, quantized, and decoded hierarchically. **(b) Hierarchical decoding.** Coarse tokens recover global structure; fine tokens refine local regions. **(c) Region-wise Complexity Guidance.** Complementary probing trees estimate per-region reconstruction benefit to build a content-adaptive quadtree.

Through self-attention, each region-indexed instruction token reads from the dense, globally contextualized ViT feature map. The resulting embeddings $Z = [z_1, \cdots, z_M]$ form continuous representations of quadtree nodes associated with their respective spatial regions.

3.1.3 Vector Quantization

To obtain discrete tokens compatible with autoregressive modeling, the node embeddings are quantized using a learned codebook $\mathcal{C} = \{e_k\}_{k=1}^{K}$: $\text{Quant}(z_i) = \arg\min_{e_k \in \mathcal{C}} \|z_i - e_k\|_2$
The resulting discrete token sequence constitutes the image representation.

3.1.4 Hierarchical Decoding

To reconstruct the image from the discrete token sequence, we reverse the tokenization process through a coarse-to-fine hierarchical decoding procedure.
The discrete tokens are first mapped back to continuous embeddings using the codebook and combined with their corresponding structural instruction embeddings. For token i associated with node (l_i, p_i), we have $z_i = e_{y_i} + q_i$ where e_{y_i} denotes the embedding of the discrete token index and q_i is the node instruction embedding encoding the tree level and spatial position. A Transformer-based *Decoder* then processes the embedding sequence as $\tilde{Z} = [\tilde{z}_1, \cdots, \tilde{z}_M]$.
Unlike standard grid-based decoders that directly reshape the output sequence into a dense spatial grid, QuadTok reconstructs the image according to the quadtree hierarchy.
Each processed node feature $\tilde{z}_i$ is first projected into a spatial patch using a level-specific projection module: $C_{l_i}(p_i) = U_{l_i}(\tilde{z}_i)$ where $U_l(\cdot)$ denotes the unpatching operator for level l, producing a spatial feature patch corresponding to region p. The patches from the same level are placed at their corresponding spatial locations on a canvas to form a residual feature map C_l. We then aggregate the feature map level-by-level as $H_{l+1} = F_{l+1}^{\text{up}}(H_l) + C_{l+1}$ where $H_0 = 0$, and F_{l+1}^{up} is a learned upsampling module for level $l + 1$.
This process starts with a coarse global representation using level-1 tokens and then progressively refines local details using higher resolution tokens. Finally, the highest-resolution feature map (H_2 in two-level configuration) is projected to RGB space, yielding the final reconstruction $\hat{I} \in \mathbb{R}^{H \times W \times 3}$.

3.1.5 TREE-STRUCTURE SAMPLING

Although the tokenizer can operate on arbitrary quadtree structures, training with a fixed fully expanded tree would cause the model to rely on a deterministic token layout. To improve robustness to variable tree structures, we employ a probabilistic tree sampling strategy during training.

Instead of always using a fully expanded quadtree, we randomly sample valid subtrees for each training iteration. All level-1 nodes are always retained to preserve the global image layout, while each parent node independently expands into its four child nodes with probability γ. When a parent node is not expanded, its spatial region remains represented solely by the coarse-level token.

This stochastic sampling exposes the tokenizer to diverse tree configurations and sequence lengths during training. As a result, the model learns to produce consistent representations under both coarse and refined token layouts, enabling it to robustly process the adaptive tree structures used during complexity-guided tokenization (to be introduced in Sec. 3.3).

3.2 KINSHIP CAUSAL ATTENTION

Flattening the quadtree into a 1D sequence removes the explicit hierarchical constraints of the original structure. Without additional restrictions, standard self-attention would allow tokens to interact globally, potentially breaking the spatial correspondence between tokens and their associated image regions. To preserve the hierarchical topology, we introduce *Kinship Causal Attention*, a topology-aware masking strategy that constrains attention according to the quadtree structure.

During encoding, the *Aggregator* Transformer processes the concatenated sequence $[F \parallel Q]$ of ViT feature tokens F and quadtree instruction tokens Q. Attention is asymmetric: ViT tokens attend bidirectionally among themselves, and quadtree tokens attend to all ViT tokens, but not vice versa. Tree tokens thus draw on globally contextualized image features, while their region and level embeddings specify the regions they represent.

During decoding, the *Decoder* Transformer processes the quantized visual tokens together with their corresponding quadtree instruction tokens. In both the encoding and decoding stages, the self-attention interactions *among quadtree tokens* are governed by our proposed *Kinship Causal Mask*, which enforces a strict coarse-to-fine dependency consistent with the quadtree hierarchy.

This mask is used within the tokenizer, whereas the autoregressive generator uses a **standard causal mask** over the BFS token sequence. For the two-level tokenizer, the Kinship Causal Mask follows three structural rules:

- **Parent $\leftrightarrow$ Parent.** Coarse-level tokens attend to each other bidirectionally to establish the global image layout.
- **Child $\rightarrow$ Parent.** Fine-level tokens attend to all parent tokens to obtain global structural context, while parent tokens are prevented from attending to child tokens.
- **Child $\leftrightarrow$ Sibling.** Fine-level tokens attend only to other child tokens that share the same parent node, restricting refinement within each spatial region.

Fig. 3 illustrates the two-level pattern. The same rule extends to deeper trees: a node can attend to all nodes at coarser levels and to siblings sharing its immediate parent at the same level, but not to finer-level nodes. The coarsest level retains full bidirectional attention. Thus, level-3 nodes attend to all level-1 and level-2 nodes and bidirectionally to their level-3 siblings.

These constraints impose a coarse-to-fine flow and restrict same-level interactions among tree tokens to siblings. The ablation in Sec. 4.4.1 evaluates their effect on reconstruction and generation.

3.3 REGION-WISE COMPLEXITY GUIDED TOKENIZATION

While the tokenizer can process arbitrary tree structures, we must determine which regions should be refined for a given image. Intuitively, regions containing complex textures require finer-level tokens, whereas homogeneous regions can remain coarse. We measure regional complexity using *reconstruction benefit*. A region is considered complex if reconstructing it with finer-level tokens significantly improves perceptual fidelity. To estimate this efficiently, we perform a probing procedure. We randomly expand half of the coarse regions to form a probing tree and construct its complementary tree where the remaining regions are expanded. This balanced design allows each region to appear once in an expanded state and once in a collapsed state, enabling efficient estimation of the regional reconstruction benefit for all regions simultaneously. The image is reconstructed under both tree structures, producing two outputs. We compute a spatial LPIPS (Zhang et al., 2018) loss map between each reconstruction and the original image I. For a region r, let

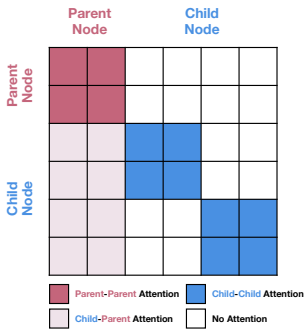


Figure 3: **Kinship causal mask.** Child, parent, and sibling attention preserves hierarchical spatial correspondence.

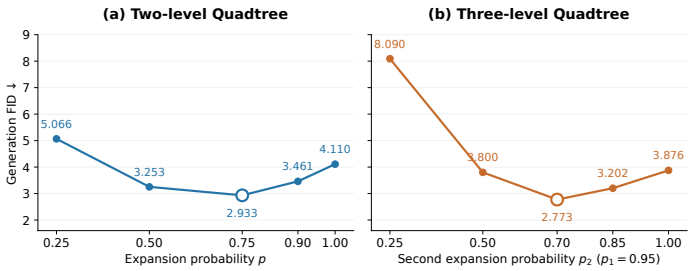


Figure 4: **Generation quality versus Quadtree expansion.** Two-level p (left) and three-level p_2 with $p_1 = 0.95$ (right). Open markers indicate the lowest evaluated FID.

$E_r^{\text{collapsed}}$ and E_r^{expanded} denote the loss maps from the probing reconstructions in which that region is collapsed and expanded, respectively. Its estimated regional reconstruction benefit is $b_r = \text{AvgPool}_r\left(E_r^{\text{collapsed}} - E_r^{\text{expanded}}\right)$. Here, AvgPool_r averages the loss difference over the spatial extent of region r. A positive b_r indicates that expansion reduces reconstruction error relative to the input image.

Regions with $b_r > \tau$ are expanded into four child nodes, where τ controls the refinement threshold. We then tokenize the image using the selected content-adaptive tree.

This strategy allocates tokens to visually complex regions while keeping simple regions at coarse resolution, enabling efficient adaptive tokenization.

3.4 Quadtree-Based Autoregressive Image Generation

Given a quadtree structure T supplied before generation, we synthesize its visual token sequence $Y = [y_1, \cdots, y_M]$ autoregressively: $p(Y|c, T) = \prod_{i=1}^{M} p(y_i|y_{<i}, c, t_{\leq i})$ where c denotes the conditioning signal (*e.g.* a class label) and $t_i = (l_i, p_i)$ represents the tree attributes of node i. The topology is selected before sampling, but node instructions are presented in BFS order. At step i, the model can use only the current and preceding node instructions, not future nodes.

We parameterize this distribution using a LLaMA (Touvron et al., 2023)-style decoder-only Transformer with a standard causal attention mask. Each visual-token prediction depends only on preceding visual codes and the supplied conditioning information; the tokenizer's bidirectional parent and sibling interactions are not used in the generator. Structural instruction embeddings encode the quadtree geometry, which specifies both the tree level and spatial position of each token, enabling the model to condition generation on spatial hierarchy.

During inference, tokens are generated sequentially in BFS order. Coarse tokens appear first, establishing the global structure, after which finer tokens progressively refine local details and textures.

The generator predicts visual codes under the supplied topology, enabling layout-controlled synthesis from user-defined or algorithmically constructed trees.

4 Experiments

4.1 QuadTok Image Reconstruction

Implementation Details. We train the QuadTok tokenizer on the ImageNet-1K (Russakovsky et al., 2015) training set at 256×256 resolution, following LlamaGen's data augmentation pipeline (Sun et al., 2024). The encoder, aggregator, and decoder use Transformer backbones with the structural constraints described in Sec. 3. Architecture and training details are in Appendix B.

Evaluation Protocol. We evaluate on the ImageNet-1K and MS-COCO (Lin et al., 2015) validation sets, applying the ImageNet-trained tokenizer to COCO without fine-tuning. We report PSNR, rFID (Heusel et al., 2018), and mean token count.

Results. Tab. 1 shows the results. Compared with LlamaGen at downsampling factor 16, content-adaptive allocation achieves lower rFID with fewer tokens, at the cost of lower PSNR. This tradeoff persists under zero-shot transfer to COCO (Tab. 1).

Table 1: **Image reconstruction on ImageNet-1K and COCO at 256×256.** #Tokens reports dataset-specific mean sequence lengths for QuadTok. f denotes the LlamaGen downsampling factor. High-resolution results are provided in Sec. D. – denotes unavailable values.

Type	Tokenizer	Codebook	ImageNet-1K			COCO		
			#Tokens	rFID↓	PSNR↑	#Tokens	rFID↓	PSNR↑
2D	Taming VQ-GAN (Esser et al., 2021)	16384	256	4.98	19.40	256	19.29	19.57
	MaskGIT VQ-GAN (Chang et al., 2022)	1024	256	2.28	–	–	–	–
	LlamaGen ($f = 16$) (Sun et al., 2024)	16384	256	2.19	20.79	256	8.11	20.42
	LlamaGen ($f = 8$) (Sun et al., 2024)	16384	1024	0.59	24.45	1024	4.19	24.20
1D	TiTok-S (Yu et al., 2024b)	4096	128	1.71	17.80	128	9.22	17.27
	FlexTok (Bachmann et al., 2025a)	64000	256	1.08	17.70	–	–	–
	One-D-Piece-L (Miwa et al., 2025a)	4096	256	1.08	19.04	256	8.04	17.94
	GigaTok-S-S (Xiong et al., 2025)	16384	256	1.01	20.74	–	–	–
	GigaTok-S-B (Xiong et al., 2025)	16384	256	0.89	20.93	–	–	–
	GigaTok-B-L (Xiong et al., 2025)	16384	256	0.81	21.21	–	–	–
	GigaTok-XL-XXL (Xiong et al., 2025)	16384	256	0.79	21.65	–	–	–
Quadtree	QuadTok	16384	230	1.46	20.37	232	7.88	19.30
	– 3-level	16384	989	0.770	23.31	1012	3.92	23.83

Reconstruction with a deeper quadtree. To evaluate the extension beyond two levels, we add a 32×32 token grid to the default $8 \times 8 \rightarrow 16 \times 16$ hierarchy at the same image resolution. The three-level tokenizer retains the shared codebook and Kinship Mask rules, with each expanded node contributing four children. The reconstruction improvements in Tab. 1 demonstrate that the representation supports finer spatial subdivision on both datasets. Depth and token budget increase together in this experiment. High-resolution (512×512) reconstruction is reported in Appendix D.

4.2 QuadTok Generation

Implementation Details. QuadTok uses a decoder-only autoregressive generator following LlamaGen (Sun et al., 2024). The two-level generator family comprises 111M (B), 343M (L), 697M (XL), and 947M (XXL) models, and the three-level generator contains 344M parameters. Generation predicts visual codes conditioned on a topology specified before sampling. For the unconditional topology setting, we sample random trees independently of the image content to be generated. Generator training and sampling details are provided in Appendix C.

4.2.1 ImageNet Generation Benchmark.

We evaluate class-conditional ImageNet generation using gFID (Heusel et al., 2018), Inception Score (IS) (Salimans et al., 2016), and precision/recall (Kynkäänniemi et al., 2019). In Tab. 2, QuadTok improves gFID over native-256 LlamaGen at comparable scales and stays competitive with larger discrete-token generators, with continuous-token xAR and FlowAR as additional references. These are cross-setting comparisons; our design choices are isolated in the ablations (Sec. 4.4). At reduced token budgets, QuadTok also achieves lower generation latency than LlamaGen configured for 256×256 images at matched model scales; the timing protocol is provided in Appendix E. We further validate QuadTok at native 512×512; these high-resolution reconstruction and generation results are provided in Appendix D.

Generation with a deeper quadtree. We evaluate generation under supplied three-level topologies using random trees formed by recursively expanding nodes into four children. The results in Tab. 2, together with Sec. 4.1, validate the representation for both reconstruction and generation beyond two levels. Fig. 4 examines the two hierarchies at different expansion probabilities, using the same CFG schedule and holding other sampling settings fixed within each sweep. Both curves show that increasing expansion initially improves gFID, but further expansion reverses the trend. The best evaluated quality therefore occurs before maximal expansion.

4.3 Zero-Shot Spatial Layout Control with QuadTok

Layout interface and demonstrations. A user or LLM (Yang et al., 2025; Bai et al., 2025) specifies a subject region, which is converted into a quadtree by expanding cells inside it and keeping coarse nodes elsewhere. The class label sets the category, and the topology is fixed before generation. We reuse the ImageNet-trained generator and tokenizer, both frozen, without layout-specific training. Fig. 1(c) and Fig. 5 show how this steers subjects toward the expanded regions.

Quantitative evaluation. We evaluate four equally sized target half-planes (left, right, top, bottom) to test both horizontal and vertical steering. Evaluation stays within the ImageNet classes,

Table 2: **Class-conditional generation on ImageNet evaluated at** 256×256**.** These are system-level comparisons with different training and sampling settings. Size denotes generator parameter count. ‡: native-256 LlamaGen results. xAR and FlowAR use continuous 2D latents.

Tokens	Model	Size	gFID↓	IS↑	Prec.↑	Rec.↑
1D	TiTok-S-128–MaskGIT (Yu et al., 2024b)	287M	1.97	281.80	–	–
	TiTok-S-128–AR* (Yu et al., 2024b)	775M	4.90	191.71	0.77	0.56
	One-D-Piece-L* (Miwa et al., 2025b)	775M	2.99	235.10	0.81	0.59
	FlexTok d18-d28 (Bachmann et al., 2025a)	1.33B	1.86	–	–	–
	GigaTok-B-L (Xiong et al., 2025)	1.4B	2.03	238.52	0.80	0.63
	GigaTok-XL-XXL (Xiong et al., 2025)	1.4B	1.98	256.76	0.81	0.62
2D	VAR-d20 (Tian et al., 2024)	600M	2.57	302.60	0.83	0.56
	VAR-d30 (Tian et al., 2024)	2.0B	1.92	323.10	0.82	0.59
	xAR-L (Ren et al., 2025a)	608M	1.28	292.50	0.82	0.62
	xAR-H (Ren et al., 2025a)	1.1B	1.24	301.60	0.83	0.64
	FlowAR-L (Ren et al., 2025b)	589M	1.90	281.40	0.83	0.57
	FlowAR-H (Ren et al., 2025b)	1.9B	1.65	296.50	0.83	0.60
	RandAR-L (Pang et al., 2025)	343M	2.55	288.82	0.81	0.58
	RandAR-XL (Pang et al., 2025)	775M	2.22	314.21	0.80	0.60
	RandAR-XXL (Pang et al., 2025)	1.4B	2.15	321.97	0.79	0.62
	LlamaGen-B‡ (Srivastava et al., 2025)	111M	5.46	193.61	0.84	0.46
	LlamaGen-L‡ (Srivastava et al., 2025)	343M	3.80	248.30	0.83	0.52
	LlamaGen-XL‡ (Srivastava et al., 2025)	775M	3.39	227.10	0.81	0.54
	DPAR-B (Srivastava et al., 2025)	120M	3.98	250.62	0.83	0.49
	DPAR-L (Srivastava et al., 2025)	352M	2.93	269.34	0.81	0.56
	DPAR-XL (Srivastava et al., 2025)	789M	2.67	281.65	0.82	0.56
Quadtree	QuadTok-B	111M	4.41	205.61	0.85	0.46
	QuadTok-L	343M	2.93	226.69	0.82	0.54
	– 3-level-L	344M	2.77	231.70	0.82	0.55
	QuadTok-XL	697M	2.23	262.38	0.807	0.60
	QuadTok-XXL	947M	2.08	273.35	0.82	0.59



Figure 5: **Zero-shot spatial layout control.** Highlighted regions specify where the input quadtree is refined. Generated subjects align with these regions using the frozen ImageNet-trained model.

so it tests zero-shot control of the existing generator. Each prescribed topology is compared with random topologies at the same token budget, which isolates the effect of spatial allocation. These experiments use predefined layouts. Details are in Appendix F.

Localization metrics. We measure target-region hit rates using the intended-class Grad-CAM center of mass, the Grounding DINO box center, and the centroid of a box-prompted SAM 2 mask. A hit requires the estimated subject location to fall in the requested half-plane. All generated images contribute to the denominator, with missed detections counted as failures for box and mask metrics.

Results. Prescribed layouts improve all three localization hit rates across the four directions (Tab. 3), demonstrating that spatial token allocation can steer subject placement under a fixed token budget. The corresponding FID-50cls changes range from -0.15 to $+0.12$ points relative to the matched random-topology baseline.

Table 3: **Zero-shot spatial control with a frozen ImageNet-trained QuadTok generator.** Prescribed vs. random topologies at 192 tokens, 1,000 images each. Grad-CAM, Box, and Mask: % of subjects in the target half-plane (↑). FID-50cls (↓): against 2,500 class-matched real images.

Target	Topology	Grad-CAM	Box	Mask	FID-50cls
Left	Random	29.6	47.5	51.1	35.03
	Prescribed	**58.3** (+28.7)	**51.8** (+4.3)	**59.0** (+7.9)	35.15 (+0.12)
Right	Random	70.4	51.0	48.5	35.03
	Prescribed	**87.9** (+17.5)	**58.1** (+7.1)	**55.7** (+7.2)	34.88 (-0.15)
Top	Random	33.9	40.7	42.9	35.03
	Prescribed	**55.4** (+21.5)	**47.0** (+6.3)	**47.8** (+4.9)	34.97 (-0.06)
Bottom	Random	66.1	58.9	56.7	35.03
	Prescribed	**83.6** (+17.5)	**64.6** (+5.7)	**62.5** (+5.8)	35.05 (+0.02)

Table 4: **Ablation of the *Kinship Causal Mask*.**

Kinship Mask	rFID ↓	PSNR ↑	gFID ↓
✗	1.41	19.16	3.17
✓	1.46	20.37	2.23

Table 5: **Effect of *region-wise complexity guidance* on ImageNet reconstruction.**

Quadtree	#Tokens ↓	rFID ↓	PSNR ↑
Random	253	1.601	19.40
Full	320	1.50	20.39
Guided	**230**	**1.46**	**20.37**

4.4 Ablation Studies

4.4.1 Effectiveness of the Kinship Causal Mask

We ablate the *Kinship Causal Mask* (Sec. 3.2) by training a baseline QuadTok with unconstrained bidirectional attention over all flattened tree tokens, and study its effect on spatial correlation, reconstruction, and generation (Tab. 4).

Spatial correlation. To probe token–region binding, we perturb a single fine-level token (l=2) with a random codebook index and compare the perturbed and unperturbed reconstructions. With the mask, the resulting difference remains localized to the corresponding spatial patch; without it, the difference spreads across the image. This spatial correspondence supports the zero-shot spatial control in Sec. 4.3.

Reconstruction and generation. This spatial isolation is also reflected quantitatively (Tab. 4). Without the mask, the tokenizer degenerates toward a sequential 1D code: rFID is slightly lower (1.46→1.41), but PSNR drops by over a point (20.37→19.16), trading spatial fidelity for perceptual realism. The gap widens downstream: with the same 697M generator and CFG, removing the mask raises gFID from 2.23 to 3.17, showing that spatially structured tokens matter for AR generation.

4.4.2 Region-wise Complexity Guidance Strategy.

Does guidance help? We compare quadtree constructions on ImageNet-1K reconstruction (Tab. 5). A uniformly expanded (*Full*) tree reconstructs well but is token-heavy (320), and a *Random* tree of similar size is worse (rFID 1.60 at 253 tokens). Guided by *Region-wise Complexity Guidance* (Sec. 3.3), QuadTok concentrates tokens on visually complex regions and matches *Full* fidelity (rFID 1.46, PSNR 20.37) with 28% fewer tokens (230). We adopt LPIPS-based guidance with τ=0.05 as the default; the full probing-metric and threshold ablation is provided in Appendix G.

5 Conclusion

We present QuadTok, a quadtree-based tokenizer that enables spatially adaptive, variable-length visual representations. By organizing tokens in a quadtree structure, QuadTok allocates tokens according to local complexity while preserving spatial correspondence between tokens and image regions which bridges adaptive tokenization and structured spatial representations. Experiments show that QuadTok achieves strong reconstruction and generation performance while enabling zero-shot spatial layout control. While this work focuses on class-conditional generation, QuadTok is compatible with multimodal pipelines and text-to-image generation, which we leave for future work.

AI USE STATEMENT

We used large language models to assist with drafting and polishing the manuscript. The authors reviewed and revised all AI-assisted text, checked the descriptions of methods and experimental results against the underlying work, and take responsibility for the final content of the paper, including its claims and conclusions. As mentioned in the paper, we use LLM to plan the quadtree structure and show it as figures.

ETHICS STATEMENT

Our experiments use the publicly available ImageNet-1K and MS-COCO datasets; we do not collect new personal data or conduct human-subject studies. As with other image generators, outputs may reflect biases in the training data and could be used to create misleading visual content. We encourage responsible use of the released artifacts and adherence to the datasets' terms of use.

REPRODUCIBILITY STATEMENT

The model design, experimental setup, and evaluation metrics are described in Secs. 3 and 4. The appendix provides tokenizer and generator training details (Appendices B and C), high-resolution and spatial-layout evaluation protocols (Appendices D.1 and F), and efficiency measurement details (Appendix E). We will release the source code and pretrained model weights to support independent reproduction of the reported results.

REFERENCES

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *ICML*, 2025a. 2, 7, 8

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length, 2025b. URL https://arxiv.org/abs/2502.13967. 2

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025. 7

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022. 7, 16

Zeyuan Chen, Kai Zhang, Zhuowen Tu, and Yuanjun Xiong. Soft tail-dropping for adaptive visual tokenization. In *ECCV*, 2026. 2

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, 2021. URL https://arxiv.org/abs/2010.11929. 3

Shivam Duggal, Phillip Isola, Antonio Torralba, and William T Freeman. Adaptive length image tokenization via recurrent allocation. In *First Workshop on Scalable Optimization for Efficient and Adaptive Foundation Models*, 2024. 2

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 2, 7

Alex Henry, Prudhvi Raj Dachapally, Shubham Pawar, and Yuxuan Chen. Query-key normalization for transformers, 2020. URL https://arxiv.org/abs/2010.04245. 15

Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer, 2024. URL https://arxiv.org/abs/2403.13298. 14, 15

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018. URL https://arxiv.org/abs/1706.08500. 6, 7

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022a. 15

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022b. URL https://arxiv.org/abs/2207.12598. 15

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 15

Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019. 7

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 11513–11522. IEEE, 2022. 2, 3

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024. 3

Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015. URL https://arxiv.org/abs/1405.0312. 6

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. In *ICLR*, 2024. 2

Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-d-piece: Image tokenizer meets quality-controllable compression. *arXiv preprint arXiv:2501.10064*, 2025a. 7, 15

Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-d-piece: Image tokenizer meets quality-controllable compression, 2025b. URL https://arxiv.org/abs/2501.10064. 2, 8

Ziqi Pang, Tianyuan Zhang, Fujun Luan, Yunze Man, Hao Tan, Kai Zhang, William T Freeman, and Yu-Xiong Wang. Randar: Decoder-only autoregressive visual generation in random orders. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 45–55, 2025. 2, 3, 8

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 16

Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Beyond next-token: Next-x prediction for autoregressive visual generation. In *ICCV*, 2025a. 3, 8

Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Flowar: Scale-wise autoregressive image generation meets flow matching. In *ICML*, 2025b. URL https://arxiv.org/abs/2412.15205. 3, 8

Tomer Ronen, Omer Levy, and Avram Golbert. Vision transformers with mixed-resolution tokenization, 2023. URL https://arxiv.org/abs/2304.00287. 3

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge, 2015. URL https://arxiv.org/abs/1409.0575. 6

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. URL https://arxiv.org/abs/1606.03498. 7

Xiaojun Shan, Haoyu Shen, Yucheng Mao, Xiang Zhang, Abhay Anand, Bingnan Li, Haiyang Xu, and Zhuowen Tu. Cyclegen: Cycle-consistent layout prediction and image generation in vision foundation models. *arXiv preprint arXiv:2603.14957*, 2026. 3

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020. 15

Junhong Shen, Kushal Tirumala, Michihiro Yasunaga, Ishan Misra, Luke Zettlemoyer, Lili Yu, and Chunting Zhou. Cat: Content-adaptive image tokenization. *arXiv preprint arXiv:2501.03120*, 2025. 2, 3, 16

Divyansh Srivastava, Akshay Mehra, Pranav Maneriker, Debopam Sanyal, Vishnu Raj, Vijay Kamarshi, Fan Du, and Joshua Kimball. Dpar: Dynamic patchification for efficient autoregressive visual generation, 2025. URL https://arxiv.org/abs/2512.21867. 3, 8

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864. 14

Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024. 1, 3, 6, 7, 14, 15, 16

Shitao Tang, Jiahui Zhang, Siyu Zhu, and Ping Tan. Quadtree attention for vision transformers, 2022. URL https://arxiv.org/abs/2201.02767. 3

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. *Advances in neural information processing systems*, 37:84839–84865, 2024. 3, 8, 16

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 6

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning, 2018. URL https://arxiv.org/abs/1711.00937. 1, 2

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 14

Yikai Wang, Zhouxia Wang, Zhonghua Wu, Qingyi Tao, Kang Liao, and Chen Change Loy. Next visual granularity generation, 2026. URL https://arxiv.org/abs/2508.12811. 2

Jun Wu. Continuous optimization of adaptive quadtree structures. *Computer-Aided Design*, 102:72–82, September 2018. ISSN 0010-4485. doi: 10.1016/j.cad.2018.04.008. URL http://dx.doi.org/10.1016/j.cad.2018.04.008. 3

Tianwei Xiong, Jun Hao Liew, Zilong Huang, Jiashi Feng, and Xihui Liu. Gigatok: Scaling visual tokenizers to 3 billion parameters for autoregressive image generation. In *ICCV*, 2025. URL https://arxiv.org/abs/2504.08736. 2, 7, 8

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 7

Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language model beats diffusion–tokenizer is key to visual generation. In *ICLR*, 2024a. 2

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation, 2024b. URL https://arxiv.org/abs/2406.07550. 2, 7, 8, 16

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric, 2018. URL https://arxiv.org/abs/1801.03924. 5, 18

Yue Zhao, Yuanjun Xiong, and Philipp Krähenbühl. Image and video tokenization with binary spherical quantization. *arXiv preprint arXiv:2406.07548*, 2024. 2

Yue Zhao, Fuzhao Xue, Scott Reed, Linxi Fan, Yuke Zhu, Jan Kautz, Zhiding Yu, Philipp Krähenbühl, and De-An Huang. Qlip: Text-aligned visual tokenization unifies auto-regressive multimodal understanding and generation, 2025. URL https://arxiv.org/abs/2502.05178. 3

A Additional Qualitative Examples

Figs. 6 to 9 show class-conditional generation at 256×256 resolution. Each grid contains 25 samples from one ImageNet class; we include the complete grids without selecting individual images.



Figure 6: Uncurated class-conditional generation on ImageNet-256. Left: macaw (class 88). Right: golden retriever (class 207). Each grid contains 25 samples.



Figure 7: Uncurated class-conditional generation on ImageNet-256. Left: Arctic fox (class 279). Right: otter (class 360). Each grid contains 25 samples.

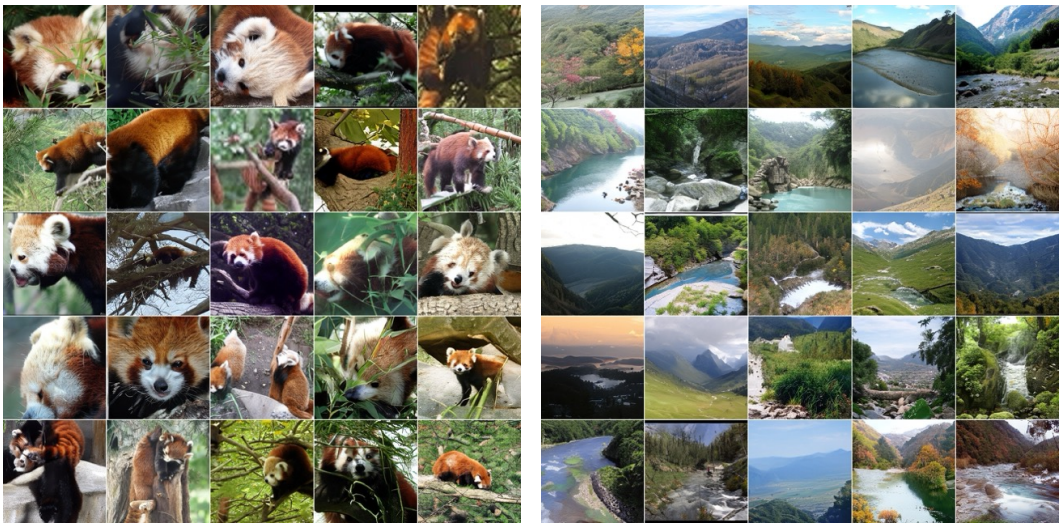


Figure 8: Uncurated class-conditional generation on ImageNet-256. Left: lesser panda (class 387). Right: valley (class 979). Each grid contains 25 samples.

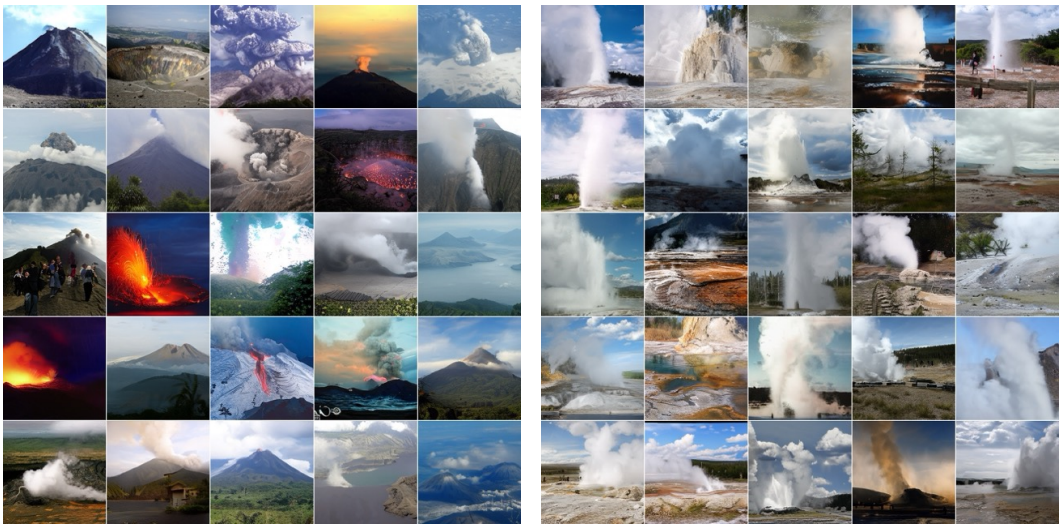


Figure 9: Uncurated class-conditional generation on ImageNet-256. Left: volcano (class 980). Right: geyser (class 974). Each grid contains 25 samples.

B QuadTok Tokenizer Training Details

We train the QuadTok tokenizer on the ImageNet-1K training set at a resolution of 256×256. The *Encoder* adopts a Vision Transformer (ViT) backbone comprising 8x Transformer (Vaswani et al., 2017) blocks with a patch size of 16×16 and learnable positional embeddings (Heo et al., 2024; Su et al., 2023). The *Aggregator* and *Decoder* modules follow a similar architecture, each containing 8x Transformer (Vaswani et al., 2017) blocks, but we only apply structural instruction embedding instead using standard position embedding widly using in Visiton Transformer.

The hidden embedding dimension across all modules is fixed at 512. Following LlamaGen (Sun et al., 2024), we employ a shared codebook across all tree levels with vocabulary size $V = 16384$ and latent dimensionality $d = 8$. During training, probabilistic tree sampling is applied with child node expansion probability $\gamma = 0.5$, which balances computational efficiency and reconstruction fidelity. The tokenizer contains approximately 120M parameters.

For the quadtree structure, we use two depth levels with spatial token grids of 8×8 and 16×16. The unpatched spatial footprint of each token is set to $p_1 = p_2 = 8$. The hidden embedding dimension is fixed to 512 across all modules. Following LlamaGen (Sun et al., 2024), we use a

shared codebook across tree levels with vocabulary size $V = 16384$ and latent dimension $d = 8$. During training, probabilistic tree sampling is applied with child expansion probability $\gamma = 0.5$, balancing computational efficiency and reconstruction fidelity. In total, the tokenizer contains approximately 120M parameters.

To optimize the tokenizer, we employ a combination of reconstruction and adversarial objectives. The overall training objective $\mathcal{L}_{total}$ is formulated as a weighted sum of four distinct loss components:

$$\mathcal{L}_{total} = \lambda_{MSE}\mathcal{L}_{MSE} + \lambda_{LPIPS}\mathcal{L}_{LPIPS} + \lambda_{VQ}\mathcal{L}_{VQ} + \lambda_{GAN}\mathcal{L}_{GAN}$$

where $\mathcal{L}_{MSE}$ denotes the Mean Squared Error for exact pixel-level fidelity, and $\mathcal{L}_{LPIPS}$ is the Learned Perceptual Image Patch Similarity loss, which preserves high-frequency visual details and structural integrity. $\mathcal{L}_{VQ}$ represents the Vector Quantization commitment loss necessary to align the continuous visual representations with the discrete codebook. To further enhance the perceptual realism and sharpness of the reconstructed images, we incorporate an adversarial GAN loss, denoted by $\mathcal{L}_{GAN}$. The hyperparameters λ_{MSE}, λ_{LPIPS}, λ_{VQ}, and λ_{GAN} control the relative contribution of each corresponding objective. To ensure initial training stability and proper codebook convergence, the adversarial weight λ_{GAN} is strictly set to 0 initially and is only activated after the first 200,000 training steps. For each loss weight, we apply the same setting with (Miwa et al., 2025a)

C QuadTok Generator Training Details

The architecture of the QuadTok generator follows the decoder-only Autoregressive(AR) generation paradigm adopted in LlamaGen (Sun et al., 2024). We incorporate several modern architectural components, including Query-Key (QK) Normalization (Henry et al., 2020), SwiGLU activation functions (Shazeer, 2020), and Rotary Position Embeddings (RoPE) (Heo et al., 2024). We scale the two-level generator across four model sizes: 111M (B), 343M (L), 697M (XL), and 947M (XXL) parameters. The three-level generator contains 344M parameters. All models are trained for 300 epochs (approximately 360K iterations) using a global batch size of 1024, without applying exponential moving average (EMA) to the model weights. Optimization is performed using the AdamW (Kingma & Ba, 2014) optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.95$, and weight decay 0.05. The learning rate is set to 4×10^{-4} for the first 250 epochs without warm-up, followed by a linear decay to 1×10^{-5} over the final 50 epochs. To mitigate overfitting and enable Classifier-Free Guidance (Ho & Salimans, 2022b;a) (CFG), the class conditioning embeddings are randomly dropped with probability 0.1 during training.

During inference, we employ a standard Key-Value (KV) cache to accelerate autoregressive decoding, together with a dynamic power-cosine CFG scheduler to improve sample fidelity. A key characteristic of our framework is that the quadtree topology must be predefined before generation.

Quadtree structure during inference. For class-conditional ImageNet evaluation, we sample quadtree topologies independently of image content using a probabilistic expansion policy. Each sampled topology specifies the regions to refine, and its node instructions are presented in breadth-first order as the generator predicts visual codes.

D High-Resolution Reconstruction and Generation

We evaluate QuadTok on ImageNet at 512×512 to test whether spatially adaptive tokenization and topology-conditioned generation extend to higher resolution. We retain two quadtree levels and enlarge the spatial grids to 16×16 and 32×32, separating increased image resolution from the deeper-tree experiment.

Reconstruction. Region-wise Complexity Guidance allocates tokens according to the reconstruction benefit of refining each region. Tab. 6 compares this allocation with fixed 2D grids and compact 1D tokenizers, demonstrating high-resolution reconstruction with fewer tokens than the 1,024-token grid baselines.

Generation. The 344M generator synthesizes native-512×512 images under random topologies supplied before generation. We evaluate 50,000 class-conditional samples using the ImageNet-512 ADM reference. In Tab. 7, QuadTok achieves lower reported gFID than DiT-XL/2 and CAT with a smaller generator, supporting the extension to high-resolution synthesis. These are system-level comparisons with different tokenizers and training and sampling settings.

Table 6: **ImageNet-1K reconstruction with 512×512 inputs.** #Tokens is the mean for QuadTok. †: LlamaGen resizes reconstructions to 256×256 for evaluation. MaskGIT and TiTok values follow Table 2 of the TiTok paper. – denotes unreported values.

Type	Tokenizer	#Tokens	Codebook size	rFID ↓	PSNR ↑
2D	MaskGIT VQ-GAN (Chang et al., 2022)	1024	1024	1.97	–
	LlamaGen† (Sun et al., 2024)	1024	16384	0.70	23.03
1D	TiTok-L-64 (Yu et al., 2024b)	64	4096	1.77	–
	TiTok-B-128 (Yu et al., 2024b)	128	4096	1.52	–
Quadtree	**QuadTok**	987	16384	0.84	24.96

Table 7: **Class-conditional ImageNet generation at 512×512.** Size denotes generator parameters. CAT uses a continuous VAE with DiT ($c = 8$, original Table 7); TiTok uses MaskGIT with 64 sampling steps. VAR uses next-scale prediction, while QuadTok uses causal autoregressive decoding. Results are system-level comparisons; – denotes unreported metrics.

Model	Size	gFID ↓	IS ↑	Precision ↑	Recall ↑
DiT-XL/2 (Peebles & Xie, 2023)	675M	3.04	240.82	0.84	0.54
CAT (Shen et al., 2025)	≈431M	4.38	181.03	0.76	0.48
TiTok-B-128 (Yu et al., 2024b)-MaskGIT	177M	2.13	261.2	–	–
VAR-*d*36-s (Tian et al., 2024)	2.3B	2.63	303.2	–	–
QuadTok	344M	2.734	268.01	0.835	0.537

D.1 Experimental details

The 512 model uses two token levels on 16×16 and 32×32 grids. All 256 coarse tokens are retained; expanding a coarse node adds four fine tokens, giving a maximum of 1,280 tokens.

Tokenizer configuration and reconstruction. The tokenizer configuration uses a ViT-small encoder, a ViT-large decoder, a patch size of 16, and a shared 16,384-entry codebook with 8-dimensional codes. The training recipe uses a coarse-node expansion probability of 0.73, 512×512 random crops, and random horizontal flips. AdamW uses learning rate 10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, and weight decay 10^{-4}. The cosine schedule has 10,000 warm-up steps and final learning rate 10^{-5}. The configured training budget is 350,000 steps with global batch size 512, BF16 precision, EMA, and gradient clipping at 1.0. The reconstruction, perceptual, VQ, and adversarial loss weights are 1.0, 1.1, 1.0, and 0.1, respectively; adversarial training starts at step 200,000.

Generator configuration and evaluation. We evaluate the 344M generator checkpoint at 400,000 steps, conditioned on a topology supplied before generation with a 16×16 coarse grid and expansion probability 0.75. Sampling uses linear classifier-free guidance with a final scale of 16, temperature 1.0, and BF16 precision. We generate 50,000 native-512×512 images, with 50 samples per ImageNet class, and evaluate them against the ImageNet-512 ADM reference. The mean generation sequence length is 1,021.28 tokens.

Table 8: **Generation speedup relative to LlamaGen.** The LlamaGen timing baseline uses randomly initialized generator weights in the 256×256 configuration and generates 256 tokens. At $p = 0.25$ and 0.50, ranges span batch sizes 1/32 and CFG scales 1/2. The $p = 0.75$ column reports the near-equal-token comparison. Ratios measure autoregressive generation, excluding image decoding; values above 1 indicate faster generation. The 775M comparison is timing-only (Sec. E).

Parameters Approx. tokens	$p = 0.25$ 128	$p = 0.50$ 192	$p = 0.75$ 255
111M	1.81–2.40×	1.14–1.58×	1.20×
343M	1.62–1.94×	1.09–1.20×	1.09×
775M	1.57–1.91×	1.15–1.21×	1.10×

E Efficiency Measurement Details

Generation efficiency. Tab. 8 compares QuadTok with LlamaGen at matched model scales on an A100-80GB GPU, using BF16, KV caching, and matched batch sizes and CFG modes. At the XL scale, LlamaGen's training hyperparameters were unstable in our setting, motivating the 697M backbone used for quality evaluation. Timing instead uses a 775M QuadTok model to match LlamaGen-XL. Lower token budgets yield larger speedups, while the near-equal-token gains reflect implementation differences. The comparison measures runtime at specified budgets; Fig. 4 examines the associated quality tradeoff.

Generation protocol. Each matched-size pair runs on one NVIDIA A100-SXM4-80GB GPU with PyTorch 2.7.1 and CUDA 12.8. Both implementations use BF16, a KV cache, full-vocabulary multinomial sampling, and the same batch size and CFG execution mode. We measure 50 batches after 10 warmups. Generation latency covers autoregressive token sampling; generation-plus-decoding latency additionally includes the tokenizer decoder. Checkpoint loading and disk/network I/O are excluded. Reported times are per batch, not per image. The two-level random topology retains 64 coarse nodes and adds four children for each independently expanded coarse node, giving an expected $64 + 256p$ tokens per image.

The benchmark uses 111M, 343M, and 775M generators. For latency measurements, we instantiate the corresponding LlamaGen architectures with randomly initialized generator weights and the 256×256 configuration, using 256 autoregressive decoding steps. This benchmark measures runtime under the specified architecture and sampling settings; image quality is evaluated separately in Tab. 2. Tab. 8 summarizes generation speedups at $p = 0.25$, 0.50, and 0.75, while Tab. 9 reports absolute latencies for all three settings at batch size 32 and CFG scale 2.

Table 9: **Absolute generation latency at batch size 32 and CFG scale 2.** Means over 50 timed batches, in seconds per batch. Gen. denotes autoregressive sampling; + decode includes image decoding. Results cover $p = 0.25$, 0.50, and 0.75 under the protocol in Appendix E. Speedup is the ratio of LlamaGen to QuadTok generation latency.

Scale	Method	Tokens	Gen. (s)	+ decode (s)	Speedup
111M	LlamaGen	256.0	5.440	5.551	1.00×
111M	QuadTok ($p = 0.25$)	126.6	2.524	2.560	2.16×
111M	QuadTok ($p = 0.50$)	189.2	3.528	3.570	1.54×
111M	QuadTok ($p = 0.75$)	255.0	4.533	4.586	1.20×
343M	LlamaGen	256.0	8.404	8.512	1.00×
343M	QuadTok ($p = 0.25$)	126.6	4.656	4.692	1.80×
343M	QuadTok ($p = 0.50$)	189.2	7.504	7.550	1.12×
343M	QuadTok ($p = 0.75$)	255.0	7.710	7.763	1.09×
775M	LlamaGen	256.0	12.612	12.720	1.00×
775M	QuadTok ($p = 0.25$)	126.6	6.587	6.620	1.91×
775M	QuadTok ($p = 0.50$)	189.2	10.643	10.688	1.18×
775M	QuadTok ($p = 0.75$)	255.0	11.465	11.520	1.10×

Pretokenization latency. Adaptive allocation adds image-dependent search to pretokenization. Tab. 10 measures latency from GPU image tensors to discrete codes, including tree search and LPIPS-based selection but excluding data loading, transfers, and reconstruction decoding. This pipeline is slower than fixed-token encoding, but its cost is incurred only during data preparation

Table 10: **Pretokenization latency at 256×256.** Mean milliseconds per image on one A100-80GB GPU. Token counts for QuadTok are measured on the timed batches (32 / 128); fixed encoders use a constant budget. Timing includes adaptive selection for QuadTok and ends at discrete codes, before reconstruction decoding.

Tokenizer	Tokens/image	Batch 32	Batch 128
LlamaGen VQ-16	256	2.927	2.885
TiTok L-32	32	1.480	1.352
TiTok B-64	64	0.586	0.512
TiTok S-128	128	0.305	0.277
QuadTok	234.3 / 233.2	8.795	6.186

and can be amortized by caching codes for generator training; generation under a supplied topology requires no image-dependent search.

Pretokenization protocol. We use a fixed ImageNet validation prefix and time five warmups followed by 30 repeats at batch sizes 32 and 128. Inputs are preloaded 256×256 image tensors on the GPU; outputs are discrete token indices on the GPU. Dataset decoding, host-to-device transfer, disk I/O, and final image reconstruction decoding are excluded. The optimized QuadTok path includes adaptive-tree search, LPIPS-based selection, and code extraction. LlamaGen VQ-16 and the released TiTok L-32, B-64, and S-128 variants use fixed-token encoders. The comparison measures actual preprocessing pipelines, not matched encoder architectures. At batch size 128, QuadTok processes 161.66 images/s versus 346.58 for LlamaGen VQ-16; the additional cost supports image-dependent regional allocation. The measured mean token counts differ slightly from dataset-wide reconstruction averages because the timing uses a fixed validation prefix.

F Spatial Layout Evaluation Details

We evaluate the frozen two-level ImageNet-trained generator and tokenizer using CFG scale 6, a power-cosine guidance schedule with exponent 1, and sampling temperature 1. Each condition contains 20 images for each of 50 fixed ImageNet classes, giving 1,000 images per condition.

The four prescribed layouts correspond to the left, right, top, and bottom half-planes. Each topology retains all nodes of an 8×8 coarse grid and expands the 32 cells in the target half-plane, adding four children per cell for a total of 192 visual tokens. The matched random baseline uniformly selects exactly 32 coarse cells for expansion independently for each image. Its generated image set is shared across the four directional evaluations.

Grad-CAM uses the intended class in ResNet-50. Box localization uses the highest-confidence intended-object box from Grounding DINO, and SAM 2 receives that box as its mask prompt. The Grad-CAM center of mass, box center, or mask centroid is tested against the requested half-plane. Box and mask scores count missed detections as failures and use all generated images as the denominator.

FID-50cls compares Inception-v3 features of the 1,000 generated images in each condition with the same 2,500 ImageNet validation images from the 50 evaluated classes. Its value is 35.03 for the random-topology baseline and 35.15, 34.88, 34.97, and 35.05 for the prescribed left, right, top, and bottom layouts, respectively.

G Complexity Guidance: Metric and Threshold

We ablate the probing signal and its expansion threshold τ (Tab. 11), comparing MSE, LPIPS (Zhang et al., 2018), and their linear combination ($\mathcal{L}_{\mathrm{MSE}} \times 10 + \mathcal{L}_{\mathrm{LPIPS}}$). At a matched $\sim$230-token budget, LPIPS aligns allocation with perceptual quality better than MSE or the combination (rFID 1.46 vs. 1.67 and 1.58); sweeping τ shows 0.05 best balances token count and fidelity. We therefore adopt LPIPS with τ=0.05 as the default.

H Limitations and Future Work

While QuadTok demonstrates strong potential in dynamic token allocation and structured autoregressive generation, our current study has two primary limitations.

Scale and Modality Constraints. First, due to computational resource limitations, our empirical evaluations are currently confined to class-conditional image generation on the ImageNet dataset.

Table 11: **Ablation of complexity guidance metric and threshold τ.** We compare MSE, LPIPS, and their linear combination as probing metrics; for LPIPS we sweep the expansion threshold τ, and report MSE and the combination at a matched $\sim$230-token budget on ImageNet-1K. LPIPS at τ=0.05 achieves the best perceptual quality (rFID 1.46) at 230 tokens, and is adopted as the default.

Guidance Metric	Threshold (τ)	#Tokens ↓	rFID ↓	PSNR ↑
$\mathcal{L}_{\text{MSE}}$	0.0005	230	1.67	20.52
$\mathcal{L}_{\text{LPIPS}}$	0.03	267	1.46	20.24
	0.05	**230**	**1.46**	**20.37**
	0.06	195	1.75	19.99
Combined ($\mathcal{L}_{\text{MSE}} \times 10 + \mathcal{L}_{\text{LPIPS}}$)	0.06	230	1.58	20.37

We have not yet scaled the framework to massive, open-domain datasets (e.g., LAION) nor extended the architecture to encompass complex text-to-image synthesis paradigms. Scaling the model capacity and extending the cross-attention mechanisms for open-vocabulary text conditioning require significantly more training compute and represent a critical next step.

Dataset Bias in Layout Control. Second, the inherent bias of the ImageNet dataset towards object-centric, single-subject compositions restricts the upper bound of our zero-shot spatial layout control. Although QuadTok effectively localizes primary subjects and separates them from the background based on topological priors, synthesizing highly complex, multi-object scenes with intricate relational layouts remains challenging. The model's structural prior is naturally limited by the compositional diversity of its training data.

We leave the scaling of QuadTok to large-scale text-to-image generation and its application to scene-level dataset for complex, multi-object layout control to future work.