CHARM: COMPRESSIBLE HILBERT ORDER AUTOREGRESSIVE MODEL FOR IMAGE GENERATION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we present CHARM, a **C**ompressible **H**ilbert Order **A**uto**R**egressive **M**odel that jointly considers spatial locality and inference efficiency in generation-order design. CHARM adopts the Hilbert order, a recursively constructed traversal that keeps contiguous intervals in the generation sequence spatially concentrated. By introducing the notion of β-Hölder locality, we prove that the Hilbert order achieves the optimal exponent $\beta = 1/2$, which in turn guarantees high local-context coverage. In contrast, raster and diagonal zigzag orders admit only the trivial exponent $\beta = 0$, leaving their worst-case local-context coverage without any non-trivial guarantee. We further show that the Hilbert order naturally induces meaningful spatial blocks for compression. To show this, we introduce a power-of-four partition for any sequence whose intervals always correspond to square regions at multiple scales under Hilbert order. Using learned summary tokens to represent each such region with a single KV cache entry, CHARM reduces the attention computation to $\mathcal{O}(n^2 \log n)$ and KV cache space to $\mathcal{O}(\log n)$. On class-conditional ImageNet generation, our recursively compressed 1.5B model achieves 1.64 FID at 256 resolution and 1.87 FID at 512 resolution. Across model scales and resolutions, CHARM improves throughput by up to $19.75\times$ over matched uncompressed baselines, achieving a favorable balance between generation quality and efficiency. Additional experiments show that the same design extends to continuous-latent autoregression.

1 INTRODUCTION

Autoregressive modeling is the de facto framework for language generation and has recently emerged as a competitive paradigm for image generation alongside diffusion models (Sohl-Dickstein et al., 2015; Song & Ermon, 2019; Ho et al., 2020; Song et al., 2020a;b; Nichol & Dhariwal, 2021). In autoregressive image generation, an image is first partitioned into patches, and the model is trained to predict or generate each patch conditioned on previous ones (van den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021; Ramesh et al., 2021; Sun et al., 2024; Yu et al., 2024b).

Applying the autoregressive paradigm to image generation faces two challenges. First, generation is computationally expensive: with a standard KV cache, autoregressive decoding over an $n \times n$ grid requires $\mathcal{O}(n^4)$ attention computation and $\mathcal{O}(n^2)$ peak KV cache space, sharply limiting generation efficiency at high resolutions. Second, the generation order, which determines how much neighboring patches around the current location have already been generated and are available as context, serves as an important inductive bias for image generation. Many classical order strategies suffer from inherent limitations. For example, raster order is widely regarded as suboptimal (Tian et al., 2024; He et al., 2025), as it introduces abrupt spatial jumps at row boundaries.

In this paper, we present CHARM, a **C**ompressible **H**ilbert Order **A**uto**R**egressive **M**odel for image, which jointly considers spatial locality and inference efficiency in generation-order design. CHARM generates image patches according to the Hilbert order, a recursively constructed traversal that forms a continuous path over the two-dimensional image grid. Its key property is that contiguous segments of the generation sequence remain spatially concentrated. First, we formalize this property using two notions: local-context coverage and β-Hölder locality. Local-context coverage quantifies how many neighboring patches around the current location have already been generated, and β-Hölder locality yields an explicit lower bound on this quantity. We then prove that Hilbert order achieves the optimal

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.

exponent $\beta = 1/2$, whereas the commonly used raster and diagonal zigzag orders satisfy only the trivial case $\beta = 0$. This establishes the Hilbert order as a principled approach to preserving spatially relevant context when flattening an image into a one-dimensional generation sequence, rather than treating the generation order as a purely empirical design choice.

Another key advantage of CHARM is that the recursive structure of the Hilbert order naturally induces a compressible representation of spatial context in images. We first introduce a power-of-four partitioning algorithm that applies to sequences of arbitrary length and can be maintained efficiently on the fly. We then show that, under the Hilbert order, the resulting partition intervals always correspond to square image regions at multiple spatial scales. These square regions provide natural spatial units for representing image context: larger-scale regions typically correspond to earlier generated tokens that are more likely to lie farther from the current position and can therefore be represented by coarser summaries, whereas smaller-scale regions correspond to more recently generated neighboring tokens and are retained at finer resolution to preserve detailed local information. Building on these advantages, we associate each region, rather than each individual token, with a single KV cache entry and develop an effective method that uses additional summary tokens to learn and update these entries. We prove that this algorithm reduces the attention computation from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^2 \log n)$, while reducing the peak KV cache memory from $\mathcal{O}(n^2)$ to $\mathcal{O}(\log n)$.

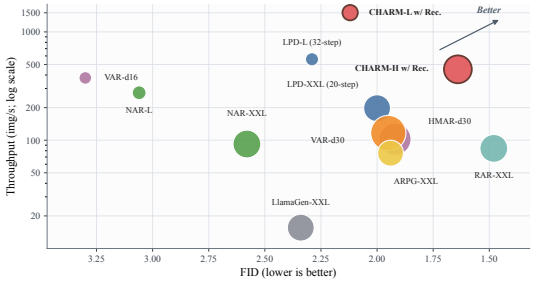


Figure 1: Generation quality versus throughput for L- and H-scale discrete-token models on ImageNet-256. Lower FID is placed farther to the right, throughput is shown on a *logarithmic scale*, and circle area is proportional to model parameter count. Throughput is measured with classifier-free guidance on a single NVIDIA B200.

We evaluate CHARM on class-conditional ImageNet generation with a discrete VQ tokenizer at 256 and 512 resolutions. Our uncompressed 1.5B-parameter model achieves 1.56 and 1.78 FID, while recursive compression retains FIDs of 1.64 and 1.87 and improves throughput by $5.37\times$ to $19.75\times$ across model scales and resolutions. At both resolutions, CHARM provides better generation quality and higher throughput than prior state-of-the-art parallel AR methods (Zhang et al., 2025; Li et al., 2025a) and VAR (Tian et al., 2024). Controlled ablations show that Hilbert order outperforms other fixed scan orders, compression better preserves quality at larger model scales, and CHARM extends to continuous-latent autoregression.

2 Related Work

Autoregressive image generation. Autoregressive image models generate an image as a sequence of patches, with each patch conditioned on previously generated patches (Van Den Oord et al., 2016; van den Oord et al., 2016; Parmar et al., 2018; Chen et al., 2020). Because directly modeling patch pixels is difficult, VQ-VAE and VQGAN encode image patches as discrete image tokens (van den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021; Yu et al., 2021; Zheng et al., 2022). These tokenized representations support both autoregressive generation (Ramesh et al., 2021; Yu et al., 2022) and masked-token generation (Chang et al., 2022; Weber et al., 2024). More recent decoder-only autoregressive models show that standard next-token prediction scales effectively to discrete image generation (Sun et al., 2024; Yu et al., 2024a). Continuous autoregressive models instead represent each patch as a continuous image token, typically use a diffusion prediction head at each generation step, and are trained with diffusion loss (Li et al., 2024; Fan et al., 2025; Yu et al., 2025; Hang et al., 2025; Ke & Xue, 2026).

Generation order and efficient decoding. Because image patches are arranged in two dimensions but generated as a one-dimensional sequence, the generation order determines which neighboring patches have already been generated and are available as context. Prior work explores several order strategies and generation schedules (Liu et al., 2024; Ren et al., 2025). VAR uses next-scale prediction and thereby changes both the generation order and granularity (Tian et al., 2024). Direction-aware and neighboring autoregressive models treat raster order as suboptimal and empirically design

alternative generation orders (Xu et al., 2025; He et al., 2025). Randomized-order models explore random orders during training or generation (Yu et al., 2024b; Pang et al., 2025). Parallel autoregressive methods generate multiple image tokens at each decoding step to improve generation efficiency (Wang et al., 2025; Li et al., 2025a; Zhang et al., 2025; Kumbong et al., 2025). In contrast, CHARM uses a fixed Hilbert order.

Hilbert order and its properties. Classical space-filling curves construct continuous mappings from a one-dimensional interval onto a two-dimensional region, with the Hilbert curve being a canonical example (Hilbert, 1935; Sagan, 2012). Its finite-resolution approximation induces the discrete Hilbert order used to traverse image patches. The Hilbert curve is constructed recursively from transformed copies of its lower-order approximation. This recursive and self-similar construction closely relates it to fractal geometry and naturally organizes image regions across multiple spatial scales (Li et al., 2025b). For autoregressive image generation, these properties are particularly useful: locality concentrates previously generated context around the current patch, while the multiscale structure allows completed regions to be represented by summary tokens and their KV cache entries to be compressed.

3 METHOD

3.1 GENERAL FRAMEWORK

Without loss of generality, let $\mathbf{x}$ be an image of size $D \times D$, partitioned into non-overlapping patches of size $k \times k$. We denote the resulting patches by $\mathbf{x} = \{\mathbf{x}_{i,j}\}$, where $i = 1, \ldots, D/k$ and $j = 1, \ldots, D/k$ index the row and column of each patch, respectively. Let $n = D/k$ be the number of rows/columns; then the total number of patches is n^2. We define a scan order function π as a bijective mapping from a one-dimensional sequence index to two-dimensional patch coordinates.

$$\pi : \{1, \ldots, n^2\} \to \{1, \ldots, n\} \times \{1, \ldots, n\}. \tag{1}$$

This order serves as the generation order in autoregressive generation, i.e., at step t, the model generates the patch $\mathbf{x}_{\pi(t)}$, conditioned on all previously generated patches $\mathbf{x}_{\pi(<t)} = (\mathbf{x}_{\pi(1)}, \ldots, \mathbf{x}_{\pi(t-1)})$. Thus, for a fixed π, the autoregressive model factorizes the image distribution as

$$P_{\mathbf{x}}(\mathbf{x}; \pi) = \prod_{t=1}^{n^2} p_{\mathbf{x}}\left(\mathbf{x}_{\pi(t)} \mid \mathbf{x}_{\pi(<t)}\right). \tag{2}$$

In practice, directly modeling patch pixels is challenging. We therefore follow discrete autoregressive image generation and use a pre-trained vector-quantized (VQ) tokenizer to first map image patches into latent representations (tokens); the autoregressive model is then trained to generate these latent tokens sequentially; and the generated latent tokens are decoded back into the image space (van den Oord et al., 2017; Razavi et al., 2019; Esser et al., 2021; Chang et al., 2022; Sun et al., 2024; Yu et al., 2024b; Tian et al., 2024). Because $\mathbf{z}_{i,j}$ represents patch $\mathbf{x}_{i,j}$, Eq. 2 can equivalently be written as

$$P_{\mathbf{z}}(\mathbf{z}; \pi) = \prod_{t=1}^{n^2} p_{\mathbf{z}}\left(\mathbf{z}_{\pi(t)} \mid \mathbf{z}_{\pi(<t)}\right). \tag{3}$$

The model parameterizes each conditional factor as a categorical distribution over the codebook and is trained with categorical negative log-likelihood (Sun et al., 2024; Yu et al., 2024b; Tian et al., 2024). We use *token* and *patch* interchangeably when unambiguous. Moreover, methods such as raster order and diagonal zigzag order define scan orders for grids of arbitrary n. We refer to these size-agnostic methods as scan-order strategies.

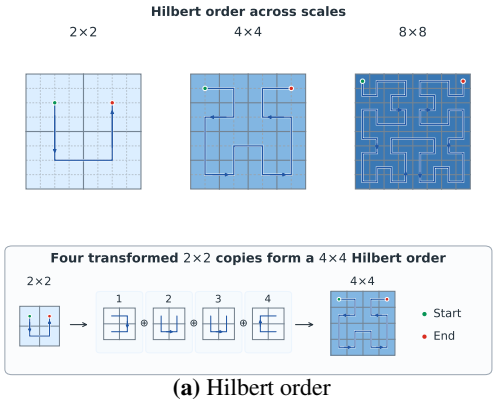
3.2 COVERAGE, LOCALITY AND HILBERT ORDER

From the viewpoint of autoregressive model, we are interested in quantifying how much local spatial context has already been generated when predicting a patch. For any step t, let $S(\pi, t) = \{\pi(1), \ldots, \pi(t-1)\}$ denote the set of previously generated patch positions and let $B(\pi(t), r) = \{u \in \{1, \ldots, n\}^2 : \|u - \pi(t)\|_\infty \le r\}$ denote the neighborhood of radius r, centered at the current patch. We measure the coverage of available local context by

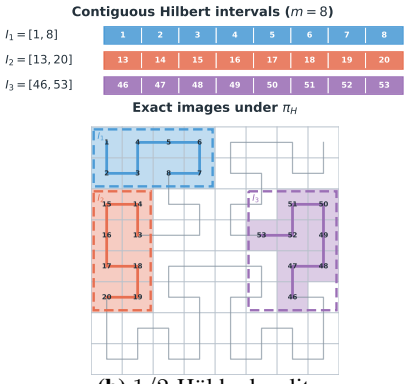
$$\mathrm{Cov}_\pi(t, r) = \frac{|B(\pi(t), r) \cap S(\pi, t)|}{|B(\pi(t), r)|}. \tag{4}$$

A higher coverage indicates that more neighboring patches around the current location have already been generated and are available as context, which generally facilitates more effective autoregressive prediction. We next show that this metric can be naturally controlled by a classical notion of locality. To better characterize locality across different scales, we first normalize the scan order π with respect to the grid size and interpolate the discrete points into a continuous curve in the unit square.

$$\gamma_\pi : [0,1] \to [0,1]^2. \tag{5}$$



(a) Hilbert order



(b) $1/2$-Hölder locality

Figure 2: Generation order and locality in CHARM. (a) Hilbert orders at increasing grid sizes, with each higher-resolution order assembled from transformed lower-resolution copies. (b) Three contiguous intervals of eight tokens and their exact images under an 8×8 Hilbert order. Each colored frame encloses precisely the grid cells visited by its corresponding interval.

Specifically, for each index $i = 1, \ldots, n^2$, let $\pi_i[0]$ and $\pi_i[1]$ denote the two-dimensional output of $\pi(i)$, respectively. We define $\gamma_\pi\left(\frac{i}{n^2}\right) = \left(\frac{\pi_i[0]}{n}, \frac{\pi_i[1]}{n}\right)$, and anchor the left endpoint by $\gamma_\pi(0) = \gamma_\pi\left(\frac{1}{n^2}\right)$. The values of γ_π between these points are obtained by linear interpolation. We measure the locality of a contiguous sequence interval by the spatial diameter of its image. Given an interval $I = [a, b] \subseteq [0, 1]$ of length $|I| = b - a$, let $\gamma_\pi(I) = \{\gamma_\pi(s) : s \in I\}$. We define

$$D_\pi(I) = \operatorname{diam}(\gamma_\pi(I)) = \sup_{u,v \in \gamma_\pi(I)} \|u - v\|_\infty, \tag{6}$$

where $\|\cdot\|_\infty$ is the infinity norm.

Definition 1 (β-Hölder locality)**.** *Let $\beta \in [0, 1]$. A scan order π is said to satisfy the β-Hölder locality condition if there exists a constant C, independent of n, such that*

$$D_\pi(I) \le C|I|^\beta \tag{7}$$

for every interval $I \subseteq [0, 1]$.

The following proposition shows that the Hölder locality condition provides a lower bound for the coverage defined in Equation 4. Therefore, an order function with a stronger Hölder locality condition can lead to higher local-context coverage, which is beneficial for autoregressive learning.

Proposition 1. *Suppose that π satisfies the β-Hölder locality condition with $\beta > 0$ and constant C. For every step t and patch-grid radius r, the local-context coverage satisfies*

$$\mathrm{Cov}_\pi(t, r) \ge \frac{1}{(2r+1)^2} \min\left\{t - 1, \left\lfloor \left(\frac{r}{Cn^{1-2\beta}}\right)^{1/\beta} \right\rfloor\right\}. \tag{8}$$

Proposition 1 establishes an explicit connection between coverage and locality. While one might expect commonly used scan orders to exhibit strong locality properties with large Hölder exponents, we show that typical scan-order strategies, including raster order and zigzag order, fail to satisfy Definition 1 for any non-trivial exponent ($\beta > 0$). They only satisfy the trivial bound with ($\beta = 0$) and ($C = 1$). The formal proofs are provided in Appendix A.2.

To obtain a scan order with non-trivial Hölder locality, we introduce the Hilbert order, whose construction is defined below.

Definition 2 (Hilbert order on a dyadic grid). *For a dyadic-sized $n \times n$ grid with side length $n = 2^L$, the Hilbert order is a bijection $\pi_H^{(L)} : \{1, \dots, 4^L\} \to \{1, \dots, 2^L\}^2$. It is defined recursively as follows. The base case is $\pi_H^{(0)}(1) = (1, 1)$. For each level $l \geq 1$ and each $i = 1, \dots, 4^{l-1}$,*

$$\begin{aligned} \pi_H^{(l)}(i) &= \big(\pi_H^{(l-1)}(i)[1],\, \pi_H^{(l-1)}(i)[0]\big), \\ \pi_H^{(l)}(4^{l-1} + i) &= \big(\pi_H^{(l-1)}(i)[0],\, \pi_H^{(l-1)}(i)[1] + 2^{l-1}\big), \\ \pi_H^{(l)}(2 \cdot 4^{l-1} + i) &= \big(\pi_H^{(l-1)}(i)[0] + 2^{l-1},\, \pi_H^{(l-1)}(i)[1] + 2^{l-1}\big), \\ \pi_H^{(l)}(3 \cdot 4^{l-1} + i) &= \big(2^l + 1 - \pi_H^{(l-1)}(i)[1],\, 2^{l-1} + 1 - \pi_H^{(l-1)}(i)[0]\big). \end{aligned} \tag{9}$$

Different from raster order which scans an image line by line, Hilbert order follows a continuous path that recursively folds into nearby regions. Specifically, it traverses one small quadrant, moves to an adjacent quadrant, and repeats this construction at every scale. Consequently, short contiguous segments in the generation sequence tend to correspond to spatially compact regions in the image. Examples of Hilbert orders at different resolutions are shown in Figure 2(a). An equivalent bit-level algorithm is provided in Appendix A.1.

The Hilbert order makes the recent context more spatially concentrated around the current patch. We next prove that it satisfies the best possible Hölder locality exponent.

Theorem 1 (Optimality of Hilbert order). *The Hilbert scan order over a dyadic grid satisfies the $\beta = 1/2$ Hölder locality condition with $C = 4$. Furthermore, this exponent is optimal: for any constants $C > 0$ and $\beta > 1/2$, and for any scan-order strategy π, there exists a size n and an interval $I \subseteq [0, 1]$ such that $D_\pi(I) > C|I|^\beta$.*

The proof of Theorem 1 is given in Appendix A.2. Combining this result with Proposition 1, and substituting $\beta = 1/2$ and $C = 4$, gives the following coverage bound for Hilbert order:

$$\mathrm{Cov}_{\pi_H}(t, r) \geq \frac{1}{(2r+1)^2} \min\left\{ t - 1, \left\lfloor \frac{r^2}{16} \right\rfloor \right\}. \tag{10}$$

3.3 Compressible Autoregressive Modeling in Hilbert Order

The inefficiency of autoregressive generation primarily stems from the ever-growing KV cache for long sequences; compressing this cache can therefore improve generation efficiency, but often at the cost of degraded generation quality. In this section, we show that the locality property of the Hilbert order naturally enables effective KV cache management. We first construct an explicit partition of the generation sequence at any generation step and demonstrate that it induces a multi-scale hierarchy of spatial quadrants under Hilbert order, providing a natural structure for cache compression.

Definition 3 (Power-4 partition). *For any step t, we construct the interval stack $\mathrm{Partition}(t)$ from left to right. Starting from the first index, we take the longest interval whose length is a power of 4. We then repeat the same procedure on the remaining suffix until all t indices are covered.*

We illustrate the power-4 partitioning procedure with an example at $t = 31$. At initialization, the longest admissible interval has length $4^2 = 16$, leaving $31 - 16 = 15$ tokens unpartitioned. For the remaining 15 tokens, the largest admissible interval size is $4^1 = 4$, yielding three intervals of length 4 and leaving 3 tokens. Finally, the remaining three tokens are partitioned into three singleton intervals of length $4^0 = 1$. Thus, the entire sequence is partitioned into seven intervals:

$$\mathrm{Partition}(31) = \big\langle \underbrace{[1:16]}_{4^2}, \underbrace{[17:20]}_{4^1}, \underbrace{[21:24]}_{4^1}, \underbrace{[25:28]}_{4^1}, \underbrace{[29]}_{4^0}, \underbrace{[30]}_{4^0}, \underbrace{[31]}_{4^0} \big\rangle. \tag{11}$$

Here, $[a : b] := \{a, a+1, \dots, b\}$ denotes a discrete index interval. The following proposition shows that the resulting partition can be maintained efficiently on the fly during autoregressive generation.

Proposition 2 (On-the-fly stack construction of power-4 partition). *The power-4 partition in Definition 3 can be maintained on the fly by a stack. Given $\mathrm{Partition}(t)$, push the new token as a singleton interval $[t + 1]$ onto the stack (the push step). Whenever the top four intervals are consecutive and have the same length, we pop and replace them with their union (the merge step) and repeat until no such four intervals remain. The resulting stack is exactly $\mathrm{Partition}(t + 1)$.*

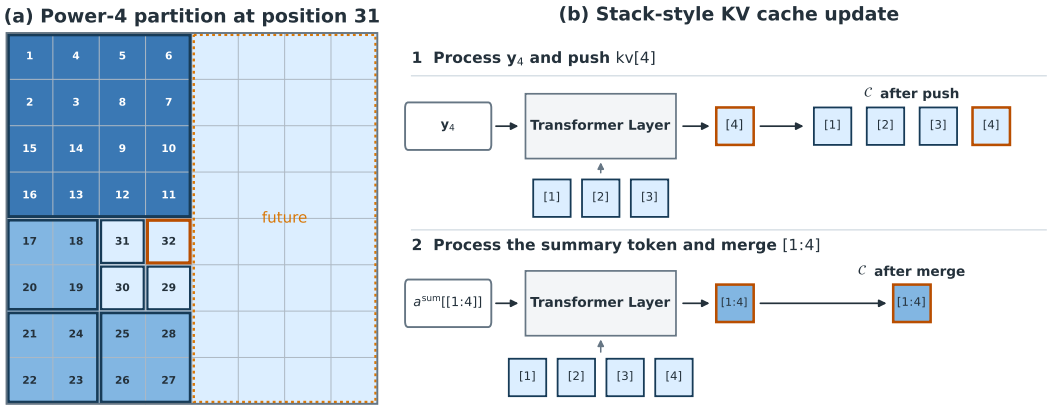


Figure 3: (a) Construction of the power-4 partition at position 31 on an 8×8 image under Hilbert traversal. We can see that every interval corresponds to a square spatial region, and together they form a multiscale partition. Fill colors distinguish interval scales, while black borders indicate interval boundaries. (b) Stack-style KV cache update for the first four image tokens. After processing $\mathbf{y}_4$, we push $\mathrm{kv}[4]$ into the KV cache. The resulting four entries trigger a merge: we feed a summary token into the model to obtain $\mathrm{kv}[[1:4]]$, which replaces the four original KV cache entries.

For example, consider the update from $t = 31$ to $t = 32$. Starting from Equation 11, we push $[32]$. The first merge replaces $[29], [30], [31], [32]$ with $[29:32]$. A second merge replaces $[17:20], [21:24], [25:28], [29:32]$ with $[17:32]$, giving $\mathrm{Partition}(32) = \langle [1:16], [17:32] \rangle$.

Multi-scale structure of the power-4 partition under Hilbert order. The power-4 partition has an exact spatial interpretation under the Hilbert order $\pi_H^{(L)}$. It is easy to prove that Definition 2 gives

$$A = [a:b] \in \mathrm{Partition}(t) \Longrightarrow \{\pi_H^{(L)}(u) : u \in A\} \text{ forms a } \sqrt{|A|} \times \sqrt{|A|} \text{ square.} \tag{12}$$

Figure 3(a) visualizes this multi-scale spatial correspondence. At any step, older information that is spatially farther from the current generation patch is organized into larger-scale units and can be compressed into coarser representations, while recently generated patches near the current patch are retained at a finer scale.

Multi-scale KV cache compression. Unlike conventional Transformers, which associate each token with a separate KV cache entry, we associate each interval $A \in \mathrm{Partition}(t)$ with a single KV cache entry. This reduces the number of cached keys and values, thereby lowering the computational cost of attention. Below, we describe how the KV cache is maintained and analyze the resulting computational cost.

For simplicity and ease of presentation, we describe the method at the level of a single Transformer layer. Specifically, we explain how to maintain the proposed KV cache within one layer during training and inference; the same strategy can be applied independently to all other layers. Under Hilbert order, we write $\mathbf{y}_t := \mathbf{z}_{\pi_H^{(L)}(t)}$ for the image token generated at step t. Without loss of generality, we denote by $\mathrm{kv}[A]$ the key–value pair associated with an interval $A \in \mathrm{Partition}(t)$, and by $\mathcal{C}$ the compressed KV cache maintained with stack-style updates. Its entries are ordered by the partition and jointly read by attention; push and pop below denote only logical updates. For a singleton interval $[t]$, we abbreviate $\mathrm{kv}[[t]]$ as $\mathrm{kv}[t]$. At generation step t, after $\mathbf{y}_t$ is generated, we feed it into the Transformer and obtain $\mathrm{kv}[t]$. We then update the cache according to

$$\mathcal{C}.\mathrm{push}(\mathrm{kv}[t])\,. \tag{13}$$

If the interval update from $\mathrm{Partition}(t-1)$ to $\mathrm{Partition}(t)$ triggers no merge, the KV cache update is complete. Otherwise, when a merge is triggered, we remove the KV entries associated with the intervals to be merged and replace them with a single KV entry that summarizes their union. Specifically, this merged KV entry is obtained by additionally feeding a special summary token into the Transformer and restricting its attention to the KV entries being merged. The key and value representations at the summary token position are then used as $\mathrm{kv}[A]$, i.e.,

$$\mathcal{C}.\mathrm{pop}(4); \quad \mathcal{C}.\mathrm{push}(\mathrm{kv}[A])\,. \tag{14}$$

This compression is performed on the fly. For example, suppose $\mathbf{y}_1, \mathbf{y}_2, \mathbf{y}_3$ have been generated and processed, as illustrated in Figure 3(b). At this point, $\mathrm{Partition}(3) = \langle [1], [2], [3] \rangle$, and the compressed KV cache is $\mathcal{C} = \langle \mathrm{kv}[1], \mathrm{kv}[2], \mathrm{kv}[3] \rangle$. Processing $\mathbf{y}_3$ produces a hidden state, which is passed to the prediction head to predict $\mathbf{y}_4$. After $\mathbf{y}_4$ is generated and processed, the singleton interval $[4]$ is pushed onto the partition stack, yielding $\langle [1], [2], [3], [4] \rangle$. Since these four consecutive intervals have the same length, they trigger a merge, resulting in $\mathrm{Partition}(4) = \langle [1:4] \rangle$. Correspondingly, $\mathrm{kv}[4]$ is first appended to $\mathcal{C}$. As the merge is triggered, we additionally feed a summary token into the model to obtain the summarized entry $\mathrm{kv}[[1:4]]$. The four original KV entries are then replaced by this summarized entry according to Equation 14. The hidden state of this summary token is then used to predict $\mathbf{y}_5$. Algorithm 1 summarizes the online cache update.

Input: grid side n and image tokens $\mathbf{y}_t$ generated online in Hilbert order $\pi_H^{(L)}$.

Initialize the interval stack as empty and set $\mathcal{C} \leftarrow \langle\,\rangle$.
for $t = 1, \ldots, n^2$ **do**
  After $\mathbf{y}_t$ is generated, process its input embedding against $\mathcal{C}$ to obtain $\mathrm{kv}[t]$. Append $[t]$ to the interval stack and execute $\mathcal{C}.\mathrm{push}(\mathrm{kv}[t])$.
  while the top four intervals $A_1, \ldots, A_4$ in the interval stack are mergeable under the partition rule above and $4|A_1| < n^2$ **do**
    Set $A \leftarrow A_1 \sqcup \cdots \sqcup A_4$, append and process the summary token for A, and obtain $\mathrm{kv}[A]$.
    Replace $A_1, \ldots, A_4$ in the interval stack by A, execute $\mathcal{C}.\mathrm{pop}(4)$ followed by $\mathcal{C}.\mathrm{push}(\mathrm{kv}[A])$ as in Equation 14.
  end while
end for

Algorithm 1: Stack-style KV cache update during CHARM inference.

Efficiency. The power-4 partition contains $\mathcal{O}(\log n)$ intervals, so each of the n^2 image tokens attends to $\mathcal{O}(\log n)$ cached entries. Across all scales, CHARM inserts $\mathcal{O}(n^2)$ summary tokens, each of which attends to only four child entries. Accounting for both components gives

$$\mathcal{O}(n^2 \log n) + \mathcal{O}(n^2) = \mathcal{O}(n^2 \log n). \tag{15}$$

Compared with standard AR using a KV cache, CHARM therefore reduces generation complexity from $\mathcal{O}(n^4)$ to $\mathcal{O}(n^2 \log n)$, and the peak per-image KV cache from $\mathcal{O}(n^2)$ to $\mathcal{O}(\log n)$. Detailed derivations are provided in Appendix A.3.

Training and implementation. Training follows the same processing order as inference: image tokens are interleaved with the summary tokens triggered by online cache merges. A fixed sparse attention mask reproduces the visibility pattern induced by the compressed KV cache in Algorithm 1, allowing parallel training. Appendix A.4 specifies the input parameterization, predictor alignment, training-time attention mask, and KV cache implementation.

4 Experiments

We evaluate CHARM on class-conditional ImageNet generation (Deng et al., 2009) to answer two questions: First, *can Hilbert order outperform other scan-order strategies for autoregressive image generation?* Second, *can the resulting spatial structure be exploited to substantially improve practical generation throughput?* We present the main results below. Model, optimization, guidance, and evaluation details are provided in Appendix A.5.

4.1 Main Results

We study both discrete VQ-token and continuous-token autoregressive settings. Throughout this section, *CHARM* denotes fixed Hilbert-order generation without compression, *CHARM w/ L1* uses one-level summary token compression, and *CHARM w/ Rec.* recursively compresses completed regions. Our discrete-token experiments follow the protocol of Yu et al. (2024b): H-size models are used for the main comparison, while L-size models are used for controlled ablations. For continuous tokens, we adopt the decoder-only autoregressive setting of MAR (Li et al., 2024). For efficiency, we report saturated throughput, defined as the maximum generator-only sampling rate achieved on a single GPU; hereafter, we refer to it simply as throughput.

Table 1: **Class-conditional ImageNet-256/512 generation.** We report generation quality together with inference throughput. Throughput is measured with classifier-free guidance on a single NVIDIA B200. Best and second-best values are bold and underlined.

Model configuration		Generation quality				Efficiency
Method	#Params	FID ↓	IS ↑	Precision ↑	Recall ↑	Throughput ↑
ImageNet-256						
VAR-d30 (Tian et al., 2024)	2.0B	1.92	323.1	**0.82**	0.59	102.24
VAR-d30-re (Tian et al., 2024)	2.0B	1.73	**350.2**	**0.82**	0.60	–
HMAR-d30 (Kumbong et al., 2025)	2.4B	1.95	334.5	**0.82**	<u>0.62</u>	116.57
LlamaGen-XXL (Sun et al., 2024)	1.4B	2.34	253.9	0.80	0.59	15.58
NAR-XXL (He et al., 2025)	1.5B	2.58	293.5	**0.82**	0.57	92.21
RAR-XXL (Yu et al., 2024b)	1.5B	**1.48**	326.0	0.80	**0.63**	84.07
RandAR-XXL (Pang et al., 2025)	1.4B	2.15	322.0	0.79	<u>0.62</u>	–
ARPG-XXL (Li et al., 2025a)	1.3B	1.94	<u>339.7</u>	<u>0.81</u>	0.59	75.87
LPD-XXL (20-step) (Zhang et al., 2025)	1.4B	2.00	337.6	0.80	0.60	<u>197.48</u>
CHARM-H (ours)	1.5B	<u>1.56</u>	333.8	**0.82**	0.60	84.07
CHARM-H w/ L1 (ours)	1.5B	1.61	313.0	<u>0.81</u>	0.61	191.34
CHARM-H w/ Rec. (ours)	1.5B	1.64	317.1	<u>0.81</u>	0.59	**451.42**
ImageNet-512						
VAR-d36-s (Tian et al., 2024)	2.3B	2.63	303.2	–	–	20.49
HMAR-d24 (Kumbong et al., 2025)	1.3B	2.99	304.1	–	–	56.27
ARPG-XL (Li et al., 2025a)	719M	2.82	277.5	**0.82**	0.56	35.16
LPD-XL (48-step) (Zhang et al., 2025)	752M	2.10	<u>326.0</u>	0.80	**0.63**	<u>61.44</u>
CHARM-H (ours)	1.5B	**1.78**	311.8	<u>0.81</u>	0.59	7.98
CHARM-H w/ L1 (ours)	1.5B	<u>1.79</u>	**329.6**	0.80	<u>0.61</u>	27.30
CHARM-H w/ Rec. (ours)	1.5B	1.87	311.4	<u>0.81</u>	0.59	**101.70**

Table 1 compares CHARM-H with strong autoregressive image generators on ImageNet at 256 and 512 resolution. Importantly, all CHARM variants at a given resolution differ only in generation order and cache compression. The comparison therefore exposes the quality–efficiency trade-off introduced by CHARM without confounding it with architectural changes.

Competitive generation quality. Without compression, CHARM-H is competitive with the strongest autoregressive baselines at both resolutions. On ImageNet-256, it obtains an FID of 1.56, second only to RAR-XXL's 1.48 among the methods considered, while outperforming both VAR-d30 variants in FID. At the more challenging 512 resolution, CHARM-H achieves an FID of 1.78, the best result in Table 1. These results demonstrate that a fixed locality-preserving Hilbert order provides a strong inductive bias for autoregressive image generation, enabling highly competitive generation quality across resolutions.

Inference efficiency. CHARM substantially improves generation efficiency by reducing the active KV cache during autoregressive decoding. This reduction allows larger batches to fit on the same accelerator and consequently increases generation throughput. Importantly, the advantage grows with image resolution: while the standard KV cache scales with the number of generated image tokens, CHARM retains only a logarithmic number of compressed cache entries (Appendix A.3). As shown in Table 1, this difference translates into increasingly larger throughput gains at higher resolutions, making CHARM particularly effective in the regime where autoregressive generation is otherwise most memory-intensive.

More importantly, CHARM trades only a small amount of generation quality for a substantial increase in throughput. Relative to uncompressed CHARM-H, recursive compression changes FID from 1.56 to 1.64 at 256 resolution and from 1.78 to 1.87 at 512—absolute differences of only 0.08 and 0.09, respectively. In return, throughput increases from 84.07 to 451.42 img/s at 256 ($5.37\times$) and from 7.98 to 101.70 img/s at 512 ($12.74\times$). This operating point is also favorable relative to prior efficient autoregressive generators. Compared with LPD, the fastest prior method reported in Table 1, CHARM-H w/ Rec. is $2.29\times$ faster at 256 resolution (451.42 vs. 197.48 img/s) while attaining a lower FID (1.64 vs. 2.00). At 512 resolution, it remains $1.66\times$ faster (101.70 vs. 61.44 img/s) while improving FID from 2.10 to 1.87.

Continuous autoregression. We further examine whether CHARM generalizes beyond discrete-token autoregression. CHARM transfers directly to the decoder-only continuous setting of MAR,

Table 2: **Ablation on autoregressive order.** L-size models; see Appendix C for visualizations.

Scan order	FID ↓	IS ↑
ImageNet-256		
Raster counterpart (Yu et al., 2024b)	2.18	269.7
Spiral-in (Yu et al., 2024b)	2.50	256.1
Spiral-out (Yu et al., 2024b)	2.46	256.6
Z-curve (Yu et al., 2024b)	2.29	262.7
Subsample (Yu et al., 2024b)	2.39	258.0
Alternate (Yu et al., 2024b)	2.48	270.9
Hilbert (CHARM-L)	**1.83**	**302.2**
ImageNet-512		
Raster counterpart	2.79	257.2
Hilbert (CHARM-L)	**2.47**	**299.1**

Table 3: **Ablation on compression strategy.** L-size discrete VQ-token models; throughput is reported in img/s.

Method	FID ↓	Throughput ↑
ImageNet-256		
CHARM-L	**1.83**	146.75
CHARM-L w/ L1	1.96	822.78
CHARM-L w/ Rec.	2.12	**1505.21**
ImageNet-512		
CHARM-L	2.47	18.70
CHARM-L w/ L1	**2.38**	63.79
CHARM-L w/ Rec.	2.77	**369.29**

without relying on a discrete codebook. Hilbert order improves the raster AR baseline from 3.59 to 3.44 FID, confirming that its locality benefit extends to continuous image tokens. Moreover, both one-level and recursive compression remain compatible with a diffusion prediction head, achieving 3.78 and 4.07 FID, respectively. These results demonstrate that CHARM's ordering and compression principles apply across image-token representations and provide a promising basis for more efficient continuous autoregressive generation. Complete results are reported in Table 7.

4.2 Ablation Studies

We next examine the two design choices underlying CHARM: locality-preserving generation order and hierarchical cache compression. Table 2 compares different deterministic scan orders, while Table 3 measures the quality-throughput trade-off introduced by one-level and recursive compression.

Ablation on autoregressive order. Hilbert order consistently provides the strongest generation quality among the evaluated deterministic scan orders. At 256 resolution, it improves FID from 2.18 to 1.83 relative to the raster counterpart and increases IS from 269.7 to 302.2. The improvement persists at 512 resolution, where Hilbert order reduces FID from 2.79 to 2.47 and raises IS from 257.2 to 299.1. These results establish locality-preserving Hilbert traversal as a strong autoregressive modeling prior rather than an incidental implementation choice.

Ablation on compression strategy. Cache compression introduces a moderate quality–efficiency trade-off. At 256 resolution, one-level and recursive compression increase FID from 1.83 to 1.96 and 2.12, respectively, while substantially reducing the active cache and improving the throuput. A similar pattern holds at 512 resolution, one-level compression attains an FID of 2.38, slightly better than the uncompressed model's 2.47. More importantly, the quality cost of compression decreases substantially with model scale. For the H-size models in Table 1, recursive compression changes FID by only 0.08 at 256 resolution and 0.09 at 512 resolution, while increasing throughput by $5.37\times$ and $12.74\times$, respectively. Together, these results indicate that larger models can enable aggressive cache compression with little degradation in generation quality. Crucially, this compression is enabled by the locality-preserving structure of the Hilbert traversal: spatially coherent regions are completed in a structured manner and can therefore be summarized hierarchically once they are no longer needed at full token-level resolution. This synergy between Hilbert ordering and hierarchical compression makes CHARM increasingly favorable as both model capacity and image resolution grow.

5 Conclusion

We presented CHARM, an autoregressive image generation framework built around a Hilbert order that jointly accounts for spatial locality and multiscale compressibility. By recursively summarizing completed spatial regions, CHARM substantially accelerates generation while maintaining high generation quality. Its consistent performance across model sizes, together with the increasing efficiency gains at higher resolutions, further demonstrates favorable scaling behavior. We believe that the broader principle of jointly designing generation order and memory compression can extend beyond class-conditional image generation to text-to-image, and video generation, where long structured sequences make locality-aware compression particularly valuable.

REFERENCES

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11315–11325, 2022.

Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. *arXiv preprint arXiv:2301.00704*, 2023.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pp. 1691–1703. PMLR, 2020.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12873–12883, 2021.

Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. Fluid: Scaling autoregressive text-to-image generative models with continuous tokens. In *International Conference on Learning Representations*, volume 2025, 2025.

Shanghua Gao, Pan Zhou, Ming-Ming Cheng, and Shuicheng Yan. Masked diffusion transformer is a strong image synthesizer. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 23107–23116. IEEE, 2023.

Tiankai Hang, Jianmin Bao, Fangyun Wei, and Dong Chen. Fast autoregressive models for continuous latent generation. *arXiv preprint arXiv:2504.18391*, 2025.

Yefei He, Yuanyu He, Shaoxuan He, Feng Chen, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Neighboring autoregressive modeling for efficient visual generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 19000–19010, 2025.

David Hilbert. Über die stetige abbildung einer linie auf ein flächenstück. In *Dritter Band: Analysis· Grundlagen der Mathematik· Physik Verschiedenes: Nebst Einer Lebensgeschichte*, pp. 1–2. Springer, 1935.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Guolin Ke and Hui Xue. Hyperspherical latents improve continuous-token autoregressive generation. In *International Conference on Learning Representations*, volume 2026, pp. 30726–30749, 2026.

Hermann Kumbong, Xian Liu, Tsung-Yi Lin, Ming-Yu Liu, Xihui Liu, Ziwei Liu, Daniel Y Fu, Christopher Re, and David W Romero. Hmar: Efficient hierarchical masked auto-regressive image generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2535–2544. IEEE, 2025.

Haopeng Li, Jinyue Yang, Guoqi Li, and Huan Wang. Autoregressive image generation with randomized parallel decoding. *arXiv preprint arXiv:2503.10568*, 2025a.

Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *ArXiv*, abs/2406.11838, 2024. URL https://api.semanticscholar.org/CorpusID:270560593.

Tianhong Li, Qinyi Sun, Lijie Fan, and Kaiming He. Fractal generative models. *arXiv preprint arXiv:2502.17437*, 2025b.

Wenze Liu, Le Zhuo, Yi Xin, Sheng Xia, Peng Gao, and Xiangyu Yue. Customize your visual autoregressive recipe with set autoregressive modeling. *arXiv preprint arXiv:2410.10511*, 2024.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PmLR, 2021.

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.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *International conference on machine learning*, pp. 4055–4064. PMLR, 2018.

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pp. 8821–8831. Pmlr, 2021.

Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

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 *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 15781–15791. IEEE, 2025.

Hans Sagan. *Space-filling curves*. Springer Science & Business Media, 2012.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020a.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020b.

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.

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.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, koray kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems*, 29, 2016.

Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pp. 1747–1756. PMLR, 2016.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Yuqing Wang, Shuhuai Ren, Zhijie Lin, Yujin Han, Haoyuan Guo, Zhenheng Yang, Difan Zou, Jiashi Feng, and Xihui Liu. Parallelized autoregressive visual generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12955–12965, 2025.

Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. Maskbit: Embedding-free image generation via bit tokens. *arXiv preprint arXiv:2409.16211*, 2024.

Yijia Xu, Jianzhong Ju, Jian Luan, and Jinshi Cui. Direction-aware diagonal autoregressive image generation. *arXiv preprint arXiv:2503.11129*, 2025.

Hu Yu, Hao Luo, Hangjie Yuan, Yu Rong, Jie Huang, and Feng Zhao. Frequency autoregressive image generation with continuous tokens. *arXiv preprint arXiv:2503.05305*, 2025.

Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved vqgan. *arXiv preprint arXiv:2110.04627*, 2021.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

Lijun Yu, José Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion-tokenizer is key to visual generation. In *International Conference on Learning Representations*, volume 2024, 2024a.

Qihang Yu, Ju He, Xueqing Deng, Xiaohui Shen, and Liang-Chieh Chen. Randomized autoregressive visual generation. *arXiv preprint arXiv:2411.00776*, 2024b.

Zhuoyang Zhang, Luke J Huang, Chengyue Wu, Shang Yang, Kelly Peng, Yao Lu, and Song Han. Locality-aware parallel decoding for efficient autoregressive image generation. *arXiv preprint arXiv:2507.01957*, 2025.

Chuanxia Zheng, Tung-Long Vuong, Jianfei Cai, and Dinh Phung. Movq: Modulating quantized vectors for high-fidelity image generation. *Advances in Neural Information Processing Systems*, 35:23412–23425, 2022.

A Additional Method Details

A.1 Hilbert Order

We give the exact Hilbert order used in the main text. We use row-column coordinates (r, c) on a square grid, with $(0, 0)$ denoting the upper-left cell. For a grid side length $n = 2^L$, define

$$\mathcal{D}_L = \{0, \dots, n-1\}^2. \tag{16}$$

The Hilbert order is a permutation

$$H_L : \{0, \dots, n^2 - 1\} \to \mathcal{D}_L, \qquad n^2 = 4^L. \tag{17}$$

For an index i, initialize $x = y = 0$, $u = i$, and iterate over $s = 1, 2, 4, \dots, 2^{L-1}$. At each scale, set

$$r_x = \left\lfloor \frac{u}{2} \right\rfloor \bmod 2, \qquad r_y = (u \bmod 2) \oplus r_x, \tag{18}$$

where $\oplus$ denotes bitwise xor on bits. If $r_y = 0$, apply the orientation update

$$(x, y) \leftarrow \begin{cases} (y, x), & r_x = 0, \\ (s - 1 - y,\ s - 1 - x), & r_x = 1. \end{cases} \tag{19}$$

Then translate into the selected quadrant,

$$(x, y) \leftarrow (x + s r_x,\ y + s r_y), \qquad u \leftarrow \left\lfloor \frac{u}{4} \right\rfloor. \tag{20}$$

After the final scale, we define

$$H_L(i) = (x, y). \tag{21}$$

The algorithm above produces 0-indexed coordinates $H_L(i) \in \{0, \dots, n-1\}^2$. For the Hilbert scan order used in the main text, we shift to the 1-indexed row–column convention of Section 3.2; the induced discrete traversal is

$$\pi_H^{(L)}(i) = H_L(i-1) + (1,1), \qquad i = 1, \dots, n^2, \tag{22}$$

so that $\pi_H^{(L)}(i) \in \{1, \dots, n\}^2$. Under this convention the first token is always the upper-left cell, $\pi_H^{(L)}(1) = H_L(0) + (1,1) = (1,1)$.

A.2 Locality Proofs

Proof of Proposition 1. Fix a prediction step t and a patch-grid radius r, and define the number of preceding tokens to be covered as

$$m^\star = \min\left\{t-1, \left\lfloor \left(\frac{r}{Cn^{1-2\beta}}\right)^{1/\beta} \right\rfloor \right\}. \tag{23}$$

The claim is immediate if $m^\star = 0$. Otherwise, consider the normalized sequence-coordinate interval

$$I_{t,m^\star} = \left[\frac{t-m^\star}{n^2}, \frac{t}{n^2}\right], \qquad |I_{t,m^\star}| = \frac{m^\star}{n^2}. \tag{24}$$

This interval contains the current token and its $m^\star$ immediate predecessors. Definition 1 gives

$$D_\pi(I_{t,m^\star}) \le C\left(\frac{m^\star}{n^2}\right)^\beta. \tag{25}$$

Because γ_π scales patch-grid coordinates by $1/n$, all $m^\star$ predecessors are within patch-grid distance

$$C(m^\star)^\beta n^{1-2\beta} \le r \tag{26}$$

of $\pi(t)$. Hence $|B(\pi(t), r) \cap S(\pi, t)| \ge m^\star$. Since $|B(\pi(t), r)| \le (2r+1)^2$, substituting Equation 23 yields Equation 8. □

Raster-order counterexample. Raster order is the row-major scan

$$\pi_{\text{ras}}(i) = \left(1 + \left\lfloor \frac{i-1}{n} \right\rfloor, 1 + \big((i-1) \bmod n\big)\right), \qquad i = 1, \dots, n^2. \tag{27}$$

Consider the sequence interval connecting $\pi_{\text{ras}}(n) = (1, n)$ and $\pi_{\text{ras}}(n+1) = (2, 1)$. Its normalized sequence length is $1/n^2$, while its normalized spatial diameter is at least $1 - 1/n$. If raster order satisfied Definition 1 for some $\beta > 0$, then

$$1 - \frac{1}{n} \le C\left(\frac{1}{n^2}\right)^\beta \tag{28}$$

would have to hold for a constant C independent of n, which is impossible as $n \to \infty$. Since every normalized image lies in the unit square, raster order satisfies the trivial $\beta = 0$ bound with $C = 1$. □

Diagonal-zigzag counterexample. Diagonal zigzag order scans anti-diagonals of constant $r + c$ and alternates direction between successive diagonals. For $d = 1, \dots, 2n-1$, let

$$S_d = \{(r, c) : 1 \le r, c \le n,\ r + c = d + 1\}, \qquad \ell_d = |S_d| = \min(d, 2n - d). \tag{29}$$

Let $A_0 = 0$ and $A_d = \sum_{a=1}^{d} \ell_a$. For each i, let d_i be the unique diagonal satisfying $A_{d_i-1} < i \le A_{d_i}$, and let $o_i = i - A_{d_i-1} - 1$. With $r_{\min}(d) = \max(1, d+1-n)$ and $r_{\max}(d) = \min(n, d)$, define

$$\pi_{\text{zig}}(i) = \begin{cases} \big(r_{\min}(d_i) + o_i,\ d_i + 1 - r_{\min}(d_i) - o_i\big), & d_i \text{ is even,} \\ \big(r_{\max}(d_i) - o_i,\ d_i + 1 - r_{\max}(d_i) + o_i\big), & d_i \text{ is odd.} \end{cases} \tag{30}$$

The central anti-diagonal $d = n$ is visited as one contiguous block of n tokens. Its normalized sequence length is $1/n$, while its normalized spatial diameter is $(n-1)/n$. If diagonal zigzag order satisfied Definition 1 for some $\beta > 0$, then

$$\frac{n-1}{n} \leq C \left(\frac{1}{n}\right)^{\beta} \tag{31}$$

would have to hold for a constant C independent of n, which is impossible as $n \to \infty$. Since every normalized image lies in the unit square, diagonal zigzag order satisfies the trivial $\beta = 0$ bound with $C = 1$. □

Proof of Theorem 1. Let $H : [0,1] \to [0,1]^2$ be the continuous Hilbert curve obtained from the same recursive construction. Fix $u < v$, and choose an integer $\ell \geq 0$ such that

$$4^{-(\ell+1)} < v - u \leq 4^{-\ell}. \tag{32}$$

The interval $[u, v]$ intersects at most two consecutive level-ℓ parameter intervals. Their images are two edge-adjacent squares of side length $2^{-\ell}$, whose union has ℓ_∞-diameter at most $2^{1-\ell}$. Therefore,

$$\|H(u) - H(v)\|_\infty \leq 2^{1-\ell} < 4|u - v|^{1/2}. \tag{33}$$

Up to a common spatial translation, the nodes of $\gamma_{\pi_H^{(L)}}$ are the cell centers of the finite Hilbert construction, so the same two-square argument gives Equation 33 at all interpolation nodes. Piecewise-linear interpolation preserves this constant: within one segment the claim follows from linearity, while across segments it follows from convexity of the norm and concavity of the square root. Consequently, for every interval $I \subseteq [0,1]$,

$$D_{\pi_H^{(L)}}(I) \leq 4|I|^{1/2}. \tag{34}$$

Thus Hilbert order satisfies the $1/2$-Hölder locality condition with $C = 4$. □

Proof of the optimality part of Theorem 1. Fix any constants $C > 0$ and $\beta > 1/2$. Choose n large enough so that

$$\frac{\sqrt{n}-1}{n} > C \left(\frac{1}{n}\right)^{\beta}. \tag{35}$$

For any bijective scan order π_n, take any contiguous block of $m = n$ patches and let I be its normalized sequence-coordinate interval, so $|I| = 1/n$. The block contains n distinct grid cells. If its unnormalized ℓ_∞-diameter were smaller than $\sqrt{n}-1$, then those cells would lie in an axis-aligned square with fewer than n cells, impossible. Therefore its normalized diameter satisfies

$$D_{\pi_n}(I) \geq \frac{\sqrt{n}-1}{n} > C \left(\frac{1}{n}\right)^{\beta} = C|I|^{\beta}. \tag{36}$$

Thus for any $C > 0$ and any $\beta > 1/2$, every family of scan orders has a resolution n and an interval I violating the β-Hölder bound. □

A.3 Generation Complexity

We analyze attention-score complexity during sampling. A full self-attention call over a prefix of length m costs $\mathcal{O}(m^2)$, whereas an incremental query with a standard KV cache costs $\mathcal{O}(m)$. We omit constant factors from the number of transformer layers, hidden width, and conditioning tokens. Let the final latent grid have height and width $n = 2^L$, so the number of image tokens is $n^2 = 4^L$. We additionally measure KV cache space by the peak number of cached entries. Each entry contributes one key and one value per Transformer layer, so restoring batch size, model depth, and KV width only multiplies the reported space bounds by model-dependent factors. Temporary attention-score matrices and other activation memory are not counted as KV cache space.

Lemma A.1. *For a standard latent autoregressive model over an $n \times n$ latent grid, generation has attention-score complexity $\mathcal{O}(n^6)$ when the entire prefix is recomputed and $\mathcal{O}(n^4)$ with a standard KV cache. The standard KV cache contains $\mathcal{O}(n^2)$ entries at peak.*

Proof. Without caching, the t-th generation step recomputes self-attention over the full prefix of length t. Therefore,

$$\sum_{t=1}^{n^2} \mathcal{O}(t^2) = \mathcal{O}((n^2)^3) = \mathcal{O}(n^6). \tag{37}$$

With a standard KV cache, only the new query is evaluated against the t cached entries, reducing the cost of step t to $\mathcal{O}(t)$. Hence,

$$\sum_{t=1}^{n^2} \mathcal{O}(t) = \mathcal{O}((n^2)^2) = \mathcal{O}(n^4). \tag{38}$$

Before the final prediction, the cache contains all preceding image token entries. Its peak space is therefore

$$\mathcal{M}_{\mathrm{AR}}^{\mathrm{KV}}(n) = \mathcal{O}(n^2). \tag{39}$$

The uncached implementation allocates no persistent KV cache; this does not include the working memory required to recompute the full prefix. □

Lemma A.2. *For VAR generation with a geometric resolution schedule and final $n \times n$ latent grid, generation has attention-score complexity $\mathcal{O}(n^4)$. A scale-wise KV cache contains $\mathcal{O}(n^2)$ entries at peak.*

Proof. Let the k-th VAR scale have height and width $n_k = a^{k-1}$, where $a > 1$ is a constant and $n_K = n$. At scale k, the transformer attends to all tokens from the previous and current scales, whose total number is

$$M_k = \sum_{j=1}^{k} n_j^2 = \sum_{j=1}^{k} a^{2(j-1)} = \mathcal{O}(a^{2k}). \tag{40}$$

Without caching, the attention-score cost at scale k is $\mathcal{O}(M_k^2) = \mathcal{O}(a^{4k})$. With a scale-wise KV cache, the n_k^2 current-scale queries attend to M_k cached or current entries, which also costs $\mathcal{O}(n_k^2 M_k) = \mathcal{O}(a^{4k})$. Thus caching avoids recomputing previous keys and values but does not change the asymptotic attention-score complexity. Summing over all scales gives

$$\sum_{k=1}^{K} \mathcal{O}(a^{4k}) = \mathcal{O}(a^{4K}) = \mathcal{O}(n^4), \tag{41}$$

because $a^{K-1} = n$ and a is constant. The cache stores entries from the generated scales. Even before generating the final scale, $M_{K-1} = \Theta(n^2)$; retaining all scales gives $M_K = \Theta(n^2)$. Hence the peak KV cache space is

$$\mathcal{M}_{\mathrm{VAR}}^{\mathrm{KV}}(n) = \mathcal{O}(M_K) = \mathcal{O}(n^2). \tag{42}$$

□

Lemma A.3. *For CHARM w/ Rec. generation over an $n \times n$ latent grid, generation has attention-score complexity $\mathcal{O}(n^2 \log n)$. The compressed KV cache contains $\mathcal{O}(\log n)$ entries at peak.*

Proof. For CHARM w/ Rec., each newly generated image token attends to $\mathcal{C}$, whose entries correspond in order to $\mathrm{Partition}(t-1)$, instead of the full generated-token prefix. Completed intervals are represented by their merged cache entries, while the unmerged suffix remains visible at finer resolution. Before predicting token t, write the prefix length $t-1$ in base four,

$$t - 1 = \sum_{l=0}^{L-1} d_l(t-1) 4^l, \qquad d_l(t-1) \in \{0, 1, 2, 3\}. \tag{43}$$

The partition contains exactly $d_l(t-1)$ intervals of length 4^l: singleton intervals represent image tokens, and longer intervals are represented by summary tokens. Hence the number of cached entries before predicting $\mathbf{y}_t$ is

$$|\mathrm{Partition}(t-1)| = \sum_{l=0}^{L-1} d_l(t-1) \le 3L. \tag{44}$$

The upper bound is attained immediately before the final image token prediction, when $t-1 = n^2-1 = 4^L-1$ and every base-four digit equals three. Therefore,

$$\max_{1\le t\le n^2} |\mathrm{Partition}(t-1)| = 3L, \qquad \mathcal{M}^{\mathrm{KV}}_{\text{CHARM-Rec.}}(n) = \mathcal{O}(\log n), \tag{45}$$

This bound also includes the constant number of entries transiently created during a token push or a summary merge. With this compressed KV cache, each newly generated image token is processed against only these entries. The total image token computation is therefore

$$\sum_{t=1}^{n^2} \mathcal{O}(|\mathrm{Partition}(t-1)|+1) \le \mathcal{O}\big(n^2(3L+1)\big) = \mathcal{O}(n^2\log n). \tag{46}$$

It remains to account for summary tokens. Their total number is

$$\sum_{l=1}^{L-1} 4^{L-l} = \frac{4^L-4}{3} = \mathcal{O}(n^2). \tag{47}$$

Each summary token attends only to the four entries it combines, so the total summary computation is $\mathcal{O}(n^2)$. Combining the two terms gives

$$\mathcal{O}(n^2\log n) + \mathcal{O}(n^2) = \mathcal{O}(n^2\log n). \tag{48}$$

This bound assumes the streaming implementation described in Section 3.3, where the cache entries of four completed equal-length intervals are replaced by a single cache entry representing their union. □

L1 compression. With L1 compression, each completed group of four image tokens contributes one L1 cache entry, while at most three singleton entries remain. Let $\mathrm{Partition}_{\mathrm{L1}}(t-1)$ denote this interval list before predicting token t. Then

$$|\mathrm{Partition}_{\mathrm{L1}}(t-1)| = \left\lfloor \frac{t-1}{4} \right\rfloor + \big((t-1) \bmod 4\big) = \Theta(t). \tag{49}$$

Consequently, L1 compression reduces the cache by a constant factor but retains the standard asymptotic bounds: $\mathcal{O}(n^4)$ attention-score computation and $\mathcal{O}(n^2)$ peak KV cache space.

Table 4 summarizes the resulting time and KV cache space bounds.

Table 4: Sampling complexity for an $n \times n$ latent grid. KV cache space counts peak cached entries and suppresses model-dependent factors.

Method	Attention-score time	Peak KV cache space
Standard AR, no cache	$\mathcal{O}(n^6)$	not used
Standard AR, KV cache	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$
VAR, scale-wise KV cache	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$
CHARM w/ L1, compressed KV cache	$\mathcal{O}(n^4)$	$\mathcal{O}(n^2)$
CHARM w/ Rec., compressed KV cache	$\mathcal{O}(n^2\log n)$	$\mathcal{O}(\log n)$

A.4 Training Sequence and Compressed KV Cache Implementation

This section provides the input parameterization, predictor alignment, and training-time attention pattern corresponding to the cache-update procedure in Algorithm 1. CHARM uses shifted autoregressive prediction: the sequence position immediately preceding an image token target provides its predictor hidden state. After an image token is generated, its input is appended to the sequence, followed by the summary tokens retained by Algorithm 1.

Input parameterization. The input embedding of image token $\mathbf{y}_t$ is $E(\mathbf{y}_t) + p^{\mathrm{tok}}_{\pi^{(L)}_H(t)} + s_0$. For the summary token associated with a completed interval A, let $a^{\mathrm{sum}}[A]$ denote its input embedding. We use

$$a^{\mathrm{sum}}[A] = e^{\mathrm{sum}} + p^{\mathrm{sum}}[A] + s_{\log_4|A|}. \tag{50}$$

Here E is an embedding lookup for discrete image tokens and a learned linear projection for continuous image tokens; e^{sum} is a learned summary token embedding. The position embeddings $p^{\mathrm{tok}}_{\pi^{(L)}_H(t)}$ and $p^{\mathrm{sum}}[A]$ identify, respectively, the image token location and the square spatial region associated with A. Finally, s_l is the scale embedding for intervals of length 4^l, with image tokens assigned scale zero.

KV cache entries. At a fixed Transformer layer, $\mathrm{kv}[A]$ denotes the KV cache entry associated with interval A. Following the abbreviation in the main text, $\mathrm{kv}[t]$ denotes $\mathrm{kv}[[t]]$ for a singleton interval $[t]$. The calligraphic symbol $\mathcal{C}$ denotes the retained KV entries at this layer, ordered by $\mathrm{Partition}(t)$. Push and pop describe only logical updates; in implementation, these entries are stored in GPU tensors and jointly accessed by attention. The collection is initialized as empty, and the updates in Equations 13 and 14 are applied independently at every layer.

Before processing image token $\mathbf{y}_t$, the entries of $\mathcal{C}$ correspond exactly to $\mathrm{Partition}(t-1)$, and attention jointly reads all of them. When processing the summary token for $A = A_1 \sqcup \cdots \sqcup A_4$, attention reads the four entries $\mathrm{kv}[A_1], \ldots, \mathrm{kv}[A_4]$. These entries remain in $\mathcal{C}$ until the summary token has been processed; they are then replaced by $\mathrm{kv}[A]$.

Predictor alignment. For image token target $\mathbf{y}_t$, the predictor hidden state h^{pred}_t comes from the immediately preceding position in the training sequence. For $t = 1$, this is the final hidden state of the conditioning sequence. For $t > 1$, the preceding position contains $\mathbf{y}_{t-1}$ when no merge follows that token, and otherwise contains the final summary token processed by the resulting cascaded merges. We denote the distribution produced by the prediction head as $p_\theta(\cdot \mid h^{\mathrm{pred}}_t, \pi^{(L)}_H(t))$, which is conditioned on both the predictor hidden state and the target position. It is categorical for discrete codes and is parameterized by a diffusion head for continuous vectors.

Parallel training. The training sequence interleaves image tokens and summary tokens in the same order in which they are processed during inference. A fixed sparse attention mask implements the visibility rules above, allowing the full sequence to be evaluated in one parallel forward pass. The prediction loss is applied only to image token targets; summary tokens are not prediction targets, although their hidden states can predict the following image token.

A.5 Experimental Details

Discrete VQ-token setting. Discrete experiments use the RAR generation pipeline (Yu et al., 2024b) and standard autoregressive categorical cross entropy over VQ token IDs. Table 5 summarizes the tokenizer, model, and training configuration used by the reported L- and H-size models.

Table 5: Configuration of the discrete VQ-token autoregressive models.

Component	Configuration
Image tokenizer	Frozen MaskGIT-f16 VQGAN (Chang et al., 2022); codebook size 1024
ImageNet-256 representation	16×16 latent grid; 256 VQ tokens
ImageNet-512 representation	32×32 latent grid; 1024 VQ tokens
L-size backbone	Width 1024, 24 layers, 16 attention heads, MLP width 4096
H-size backbone	Width 1408, 40 layers, 16 attention heads, MLP width 6144
Training duration	250k steps for ImageNet-256; 200k steps for ImageNet-512
Batch and hardware	Global batch size 2048 on 32 GPUs
Optimizer	AdamW, betas $(0.9, 0.96)$, weight decay 0.03
Learning-rate schedule	Peak LR 4×10^{-4}, 62.5k warmup steps, cosine decay to 10^{-5}
Training stabilization	Gradient clipping 1.0, bf16 mixed precision, TF32 enabled, no EMA
Classifier-free guidance (Ho & Salimans, 2022)	Class-label dropout 0.1 during training

Continuous image token setting. We further evaluate CHARM under the decoder-only autoregressive setting of MAR (Li et al., 2024), where images are represented by continuous KL-VAE latents and each conditional distribution is parameterized by a DiffLoss head. MAR reports 4.69 FID for its decoder-only raster baseline with CFG scale 3. We find that CFG scale 4.5 performs better in our setting and use it consistently for all results in Table 7.

All models are trained on ImageNet-1K at 256×256 resolution. We use the frozen KL-16 VAE from MAR with a stride of 16, which maps each image to a 16×16 grid of 256 continuous image tokens, each represented by a 16-D KL latent vector. The VAE remains fixed throughout generator training. We precompute posterior moments for both the original and horizontally flipped images and select between them with equal probability during training, corresponding to center-crop and random-horizontal-flip augmentation. Latents are sampled from the selected posterior moments and scaled by 0.2325.

We compare a raster-order baseline with three Hilbert-order variants: CHARM, CHARM w/ L1, and CHARM w/ Rec. Each continuous image token is linearly projected to the Transformer dimension, and the resulting predictor hidden state conditions the MAR cosine DiffLoss head. At inference time, the DiffLoss sampler generates each token conditioned on its corresponding predictor hidden state. Table 6 summarizes the remaining model and optimization details.

Table 6: Configuration of the continuous image token autoregressive models.

Component	Configuration
AR backbone	Width 1024, depth 32, 16 attention heads, MLP ratio 4
Normalization and dropout	LayerNorm $\epsilon = 10^{-6}$; attention/projection dropout $0.1/0.1$
Trainable parameters	424.93M for uncompressed raster- and Hilbert-order models; compressed variants add only summary token parameters
DiffLoss head	MAR cosine DiffLoss; `diffloss_d=3`, `diffloss_w=1024`, `diffusion_batch_mul=4`
Diffusion sampling	100 sampling steps per autoregressive token
Optimizer	AdamW, betas $(0.9, 0.95)$, weight decay 0.02; no decay on bias, normalization, or DiffLoss parameters
Learning-rate schedule	Base LR 10^{-4}, peak LR 8×10^{-4}, 100 warmup epochs, then constant; 400 epochs total
Training stabilization	Gradient clipping 3.0, mixed-precision AMP, EMA rate 0.9999
Batch and hardware	Global batch size 2048; 16 B200 GPUs; per-GPU batch size 128
Classifier-free guidance	Class-label dropout 0.1 during training
Offline evaluation	EMA weights, frozen KL-16 decoder, 50k generated images, and 100 DiffLoss sampling steps

Evaluation and speed benchmarking. Discrete quality evaluation uses 50k generated samples. ImageNet-256 and ImageNet-512 use the corresponding virtual labeled reference statistics from the evaluator. Sampling hyperparameters, including the guidance scale and schedule (Chang et al., 2023; Gao et al., 2023), are either kept at the model-size default or swept separately for FID optimization. The default L-size sampling setting uses temperature 1.0, guidance scale 16.0, and guidance power 2.75; the default H setting uses temperature 1.02, guidance scale 8.0, and guidance power 1.2. For the guided discrete-token throughput experiments in Table 8, we benchmark generator-only computation on a single NVIDIA B200 in bf16, excluding tokenizer decoding and file I/O, and report the maximum throughput for L- and H-size models at both resolutions. Prior-method throughput values are remeasured from their official public implementations under the same hardware, precision, and generator-only measurement boundary, with classifier-free guidance enabled and each method's decoding schedule preserved. For each candidate batch size, we use 100 warm-up and 500 timed model steps. The standard AR baseline uses FlashAttention (Dao et al., 2022) with a standard KV cache, and its generation code is optimized using `torch.compile`, while compressed CHARM models use our compressed KV cache implementation.

A.6 Additional Experimental Results

Table 8 provides an additional throughput comparison across model scales and image resolutions. Figure 4 complements the ImageNet-256 comparison in Figure 1 by visualizing the quality–throughput trade-off at 512 resolution. Table 7 reports the complete continuous image token comparison discussed in the main text.

Table 7: Continuous image token results with the decoder-only autoregressive setting of MAR on ImageNet-256.

Method	#Params	FID ↓	IS ↑
MAR AR baseline (raster) (Li et al., 2024)	425M	3.59	262.9
CHARM-L (Hilbert)	425M	3.44	259.3
CHARM-L w/ L1	425M	3.78	218.4
CHARM-L w/ Rec.	425M	4.07	213.2

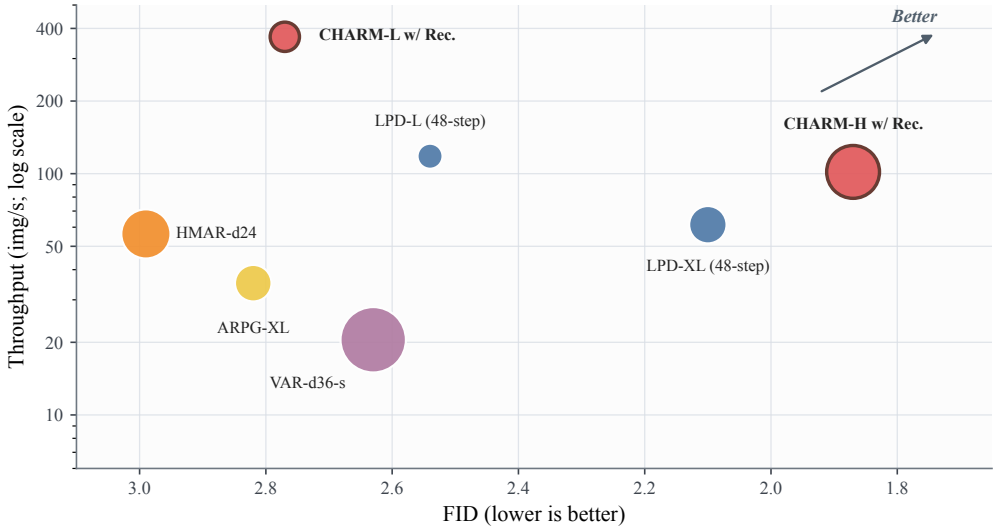


Figure 4: Generation quality versus throughput for L- and H-scale discrete-token models on ImageNet-512. Lower FID is placed farther to the right, throughput is shown on a logarithmic scale, and circle area is proportional to model parameter count. Throughput is measured with classifier-free guidance on a single NVIDIA B200.

Table 8: Detailed throughput of discrete-token generators with classifier-free guidance on a single NVIDIA B200.

L-size			H-size		
Resolution	Method	img/s ↑	Resolution	Method	img/s ↑
256	LPD-L (32-step)	558.96	256	LPD-XXL (20-step)	197.48
	NAR-L	274.34		NAR-XXL	92.21
	VAR-d16	375.83		VAR-d30	102.24
	HMAR-d16	395.47		HMAR-d30	116.57
	CHARM-L	146.75		CHARM-H	84.07
	CHARM-L w/ L1	822.78		CHARM-H w/ L1	191.34
	CHARM-L w/ Rec.	**1505.21**		CHARM-H w/ Rec.	**451.42**
512	LPD-L (48-step)	118.12	512	LPD-XXL (48-step)	36.24
	NAR-L	20.10		NAR-XXL	10.11
	VAR-d16	108.21		VAR-d30	30.41
	HMAR-d16	117.48		HMAR-d24	56.27
	CHARM-L	18.70		CHARM-H	7.98
	CHARM-L w/ L1	63.79		CHARM-H w/ L1	27.30
	CHARM-L w/ Rec.	**369.29**		CHARM-H w/ Rec.	**101.70**

LPD decoding steps follow the configurations reported in the original paper.

B GENERATED SAMPLES

Figures 5 and 6 show samples generated by CHARM-H w/ Rec. at the two evaluated resolutions.



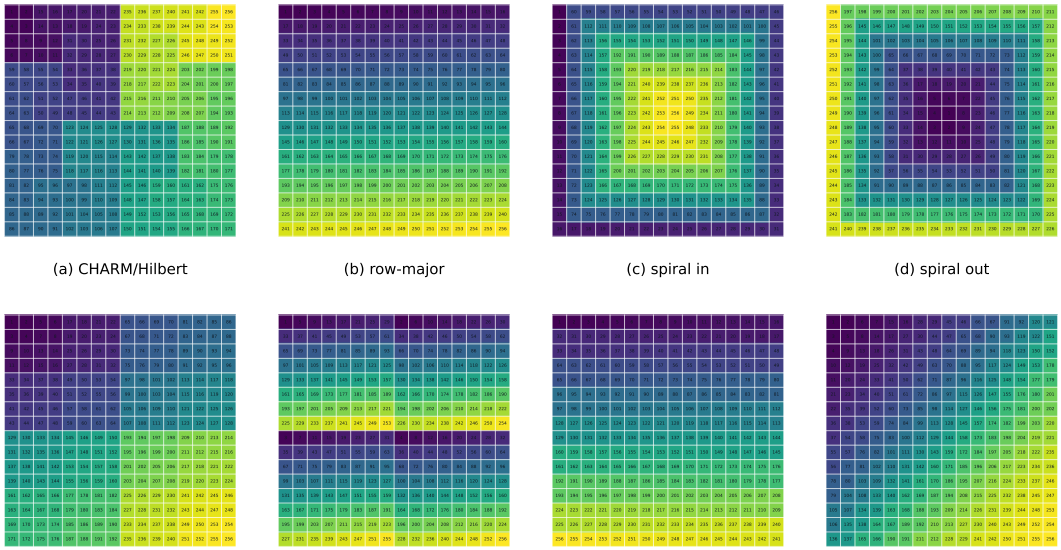
Figure 5: ImageNet samples generated by CHARM-H w/ Rec. in the discrete VQ-token setting at 256×256 resolution.



Figure 6: ImageNet samples generated by CHARM-H w/ Rec. in the discrete VQ-token setting at 512×512 resolution. Each sample is displayed at twice the linear size of those in Figure 5.

C VISUALIZATION OF SCAN ORDERS

Following RAR (Yu et al., 2024b), we visualize a scan order by placing the sequence index in each spatial cell and coloring cells according to their normalized position in the sequence. Figure 7 shows the scan orders discussed in the main text and evaluated in our scan-order ablation.



(a) CHARM/Hilbert (b) row-major (c) spiral in (d) spiral out

(e) z-curve (f) subsample (g) alternate (h) diagonal zigzag

Figure 7: Visualization of scan orders on a 16×16 grid. The color indicates each cell's normalized position in the scan order.

D DISCLOSURE OF AI USAGE

Generative AI tools were used to assist with polishing the manuscript, including checking and correcting grammar and spelling. All AI-assisted text was reviewed by the authors, who take full responsibility for the final content of this work.