# MARché: Fast Masked Autoregressive Image Generation with Cache-Aware Attention

Chaoyi Jiang[*1], Sungwoo Kim[*2], Lei Gao[1], Hossein Entezari Zarch[1], Won Woo Ro[2], and Murali Annavaram[1]

[1] University of Southern California
{chaoyij, leig, entezari, annavara}@usc.edu
[2] Yonsei University
{sungwoo.kim, wro}@yonsei.ac.kr

**Abstract.** Masked autoregressive (MAR) models unify the strengths of masked and autoregressive generation by predicting tokens in a fixed order using bidirectional attention for image generation. Although effective, MAR models incur substantial computational overhead because they recompute attention and feed-forward representations for every token at each decoding step, even though the majority of tokens remain semantically unchanged across steps. We propose a training-free generation framework **MARché** to address this inefficiency through two key components: *cache-aware attention* and *selective KV refresh*. Cache-aware attention partitions tokens into active and cached sets, enabling separate computation paths that allow efficient reuse of previously computed key/value projections without compromising full-context modeling. However, a cached token cannot be used indefinitely without recomputation due to the changing contextual information over multiple steps. MARché recognizes this challenge and applies a technique called selective KV refresh. Selective KV refresh identifies contextually relevant tokens based on attention scores from newly generated tokens and updates only those tokens that require recomputation, while preserving image generation quality. MARché significantly reduces redundant computation in MAR without modifying the underlying architecture. Empirically, MARché achieves up to 1.7× speedup with negligible impact on image quality, offering a scalable and broadly applicable solution for efficient masked transformer generation.

## 1 Introduction

Transformer models have achieved remarkable success in language generation (3; 26; 30; 31; 32; 43; 45), spurring their adoption in visual domains such as image synthesis. Autoregressive image generation models, in particular, generate pixels (6; 25; 35) or tokens (10; 16; 27; 29; 33; 39; 42; 50) sequentially by using causal attention to predict one token at a time conditioned on the past. Although effective, this strictly sequential process is inherently slow and, more importantly, ill-suited for capturing the two-dimensional spatial structure of images.

[*] Equal contribution.

To overcome these limitations, masked generative models (4; 5; 18) have emerged as a powerful alternative, characterized by parallel token generation and bidirectional attention. This leads to improved spatial coherence and significantly faster inference. Recently, masked autoregressive (MAR) models (19) have demonstrated superior performance by combining the expressive power of autoregressive modeling with the efficiency of masked prediction. Unlike prior approaches that rely on vector quantization (11; 34; 44), MAR operates in a continuous token space and employs a diffusion-based loss to model per-token distributions more effectively. Leveraging bidirectional attention within this framework, MAR achieves state-of-the-art image generation performance.

Despite its advantages, MAR, similar to masked generative approaches, still incurs substantial computational overhead. At each decoding step, the model recomputes attention and feed-forward representations for all tokens, even though only a small subset is selected for generation (also referred to as the unmasked tokens). Empirically, we observe that a large number of key/value projections are stable across steps, showing minimal variation. This redundancy leads to inefficient computation that limits the scalability of MAR-based generation.

To address this inefficiency, we propose a novel, training-free generation approach **MARché** – MAR image generation with cache-aware attention. Our key insight is that token representations in MAR exhibit temporal locality: only a small subset of tokens (those currently being generated or contextually influenced by them) need to be updated, while the rest can safely reuse previously computed attention projections.

MARché consists of two core components: *cache-aware attention* and *selective KV refresh*. Cache-aware attention is an efficient mechanism that partitions tokens into *active* and *cached* sets, each following a separate computation path. The active set consists of tokens that are in need of computing their K and V values. The active set includes all the tokens that have been generated in the previous step and the tokens that are selected for generation in the current step. The cached set includes all the remaining tokens that have their K and V values computed in prior steps and will be reused. However, as we show in our empirical evaluations, reusing cached KV values indefinitely is detrimental to the generation quality. Hence, we exploit the structure of MAR to identify several contextually relevant tokens to be dynamically moved from cached set to the active set. We call this dynamic transition of token as the KV refresh component. We explicitly decouple the computation for active and cached tokens in both the attention and feed-forward layers, enabling efficient reuse of stable representations while preserving full bidirectional context.

Selective KV refresh complements cache-aware attention by efficiently identifying which tokens require recomputation. Inspired by previous works that use token-level correlation to manage KV caches (1; 7; 12; 40; 48; 52), we analyze attention scores from generating tokens and select only the most contextually relevant ones, defined by their high attention scores, as *refreshing tokens*. These tokens are then included in the active set. This lightweight mechanism preserves image generation quality while minimizing redundant computation.

Through extensive experiments, we demonstrate that MARché significantly accelerates MAR generation while maintaining high image fidelity. Compared to standard MAR, MARché achieves up to 1.7× speedup while preserving generation quality. Importantly, MARché requires no architectural changes to the transformer and is fully compatible with existing MAR frameworks, making it broadly applicable in practice.

## 2 Related Work

*Image generation with transformers.* Transformer models have been extended to image synthesis by leveraging discrete token representations (11; 34; 44), which enable language-modeling techniques to be applied to visual data. Early approaches (16; 33; 39; 42; 50) adopt autoregressive decoding with causal attention, but struggle to model spatial dependencies. To address this, MaskGIT (5) proposes a masked generative model that predicts all tokens in parallel and refines them iteratively, offering improved spatial consistency and faster generation. Following this, recent masked generative models (4; 17; 18; 46) further improve efficiency and flexibility. Among them, MAR (19) stands out by unifying the strengths of autoregressive and masked modeling: it introduces bidirectional attention into a fixed-order autoregressive framework and eliminates the reliance on vector quantization through a diffusion-based loss. These optimizations allow MAR to achieve state-of-the-art image generation performance with both higher fidelity and faster inference, and form the baseline approach for this paper. Other methods, such as VAR (42), explore hierarchical prediction strategies, but follow a different direction.

*Efficient image generation.* Diffusion models have been widely studied from the perspective of efficiency, with strategies including accelerated sampling (20; 21; 22), representation compression (51; 53; 54), and intermediate result caching (2; 23; 47). In contrast, efficiency in transformer-based generative models has received less attention despite their high-quality outputs. For autoregressive transformers, speculative decoding (15; 41) enables partial parallelism by sampling ahead and verifying predictions, but it is constrained to left-to-right generation and thus orthogonal to our masked setting.

Within masked generative models, several recent approaches have explored efficiency from different perspectives. ENAT (24) accelerates decoding by modeling spatial and temporal dependencies between tokens but requires training a dedicated attention module, while our approach is training-free. LazyMAR (49) speeds up MAR by detecting redundant token updates through hidden feature similarity. Instead of leveraging the KV cache, LazyMAR reuses hidden features. As it adaptively selects tokens based on feature similarity, it departs from MAR's predefined generation order, potentially altering the intended generation trajectory, while our approach preserves the order. MaGNeTS (13) improves generation efficiency by dynamically scaling model size during decoding and caching KV pairs, but its caching is tightly integrated with the model's progressive resizing strategy.

KV caching is proposed to improve the efficiency of auto-regressive models (1; 3; 7; 12; 31; 38; 40; 48; 52). However, the MAR paradigm was not designed to exploit KV caching since the $K$ and $V$ values of a token are repeatedly updated on each step of MAR. MARché makes the key observation that KV projections of many tokens remain stable across steps, and hence KV caching can in fact be adapted to the MAR strategy to improve efficiency. However, the cached KV values have to be occasionally recomputed when the attention scores of cached tokens change. MARché exploits these observations to improve the MAR paradigm without requiring any additional training or architectural modification. Crucially, we decouple the attention mechanism into recomputation and reuse phases, enabling selective KV refresh based on token-level relevance. By leveraging the structural properties and generation semantics of MAR, MARché achieves both computational efficiency and architectural generality in a lightweight, training-free manner.

## 3 Preliminary: Masked Autoregressive Model

MAR models generate images by iteratively predicting a subset of masked tokens based on previously known ones, following a randomly permuted order. This approach generalizes traditional autoregressive decoding by allowing multiple predictions per step while maintaining the autoregressive nature of next-token (or next-set-of-tokens) prediction.

*Formulation.* Let $x = (x_1, x_2, \ldots, x_N)$ denote a sequence of image tokens to be generated. At the start of inference, all tokens are unknown and initialized as masked, i.e., $x_i^{(0)} = \texttt{[MASK]}$ for all $i$, and the initial set of unknown positions is $M^{(0)} = \{1, 2, \ldots, N\}$. A random permutation $\pi$ of the index set $\{1, 2, \ldots, N\}$ defines the generation order, which decides the subset of positions $U^{(t)} \subset M^{(t)}$ to be predicted at each generation step, where $M^{(t)}$ is the current set of masked token positions. $U^{(t)}$ follows this fully randomized order, rather than being chosen adaptively.

The generation proceeds in two stages: encoding and decoding. The encoder $f_{\text{enc}}$ processes only the known tokens, i.e., those at positions $i \notin M^{(t)}$, embedding each with its positional encoding $\text{PE}_i$. This yields contextual features for the visible part of the sequence:

$$h_{\text{enc}}^{(t)} = f_{\text{enc}}(\{x_i^{(t)} + \text{PE}_i \mid i \notin M^{(t)}\}). \tag{1}$$

To form the decoder input, we combine $h_{\text{enc}}^{(t)}$ with embeddings $h_{\text{mask}}^{(t)}$ for the masked tokens (i.e., positions in $M^{(t)}$), aligned with their respective positional encodings. The decoder $f_{\text{dec}}$ attends to the entire sequence and outputs contextual vectors:

$$z^{(t)} = f_{\text{dec}}(\text{concat}(h_{\text{enc}}^{(t)}, h_{\text{mask}}^{(t)}), \text{PE}). \tag{2}$$

For each selected position $i \in U^{(t)}$, a token value is sampled from the conditional distribution modeled by the diffusion sampler, conditioned on the decoder

output $z_i^{(t)}$:

$$x_i^{(t+1)} \sim p_\theta(x_i \mid z_i^{(t)}; \tau), \tag{3}$$

where $\tau$ is a temperature parameter that controls the diversity of the generated samples. The predicted values replace the corresponding masked tokens, and the set of unknown positions is updated:

$$M^{(t+1)} = M^{(t)} \setminus U^{(t)}. \tag{4}$$

This iterative generation process continues, progressively reducing the number of masked positions, until the entire token sequence is completed, i.e., $M^{(T)} = \emptyset$.

*Motivation.* Although MAR models follow an autoregressive decoding paradigm, they use bidirectional attention at each decoding step, which allows *all* tokens, including both generated and yet to be generated ones, to update their contextual representations. This design enables global context exchange, but also implies that the features of *all* tokens are recomputed at every step.

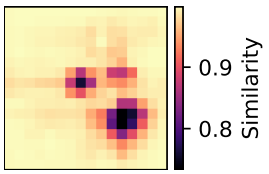


**Fig. 1: Cosine similarity of key projections** between step 4 and 5 (in layer 2) during a single image generation. The x- and y-axes represent token indices.

However, since only a small subset of tokens is newly generated at each step, the contextual information of the majority of tokens varies minimally between steps. To quantify this redundancy, we analyze the cosine similarity of key projections between consecutive steps. As shown in Figure 1, a large number of tokens exhibit consistently high similarity, often exceeding 0.95, indicating that their internal representations remain nearly unchanged. We observe a similar trend for value projections, reinforcing this observation.

These findings suggest that fully recomputing attention representations for all tokens at every step is inefficient. By identifying and reusing stable token representations, we can significantly reduce computational cost.

## 4 MARché: Cache-Aware Attention with Selective KV Refresh

As observed in Section 3, many tokens in MAR generation exhibit stable KV projections throughout steps. This temporal redundancy suggests an opportunity to reduce the computation by caching and reusing KV projections. However, not all tokens are safe to cache, as some exhibit noticeable changes in their contextual representations across steps.

To address this challenge, we introduce two key ideas. (1) *cache-aware attention*, which separates tokens into *active* and *cached* sets, assigning them to distinct computation paths during attention and feed-forward layers (Section 4.1). (2) *Selective KV cache refresh* , augments active tokens by adding *refreshing*

*tokens*. These tokens are not being generated but are contextually influenced by generation targets. To efficiently identify them, we propose a lightweight attention-guided selection strategy, described in Section 4.2.

### 4.1 Cache-aware Attention Mechanism

*Token categorization: active vs. cached.* At each generation step, we categorize tokens into two groups: *active* tokens and *cached* tokens. Active tokens are those whose contextual representations must be updated; cached tokens are considered to have very limited changes to their KV projections and can safely reuse their cached key/value (KV) projections.

The active set includes three types of tokens: (1) *Generating tokens*, which are masked tokens selected for prediction at the current step; (2) *Caching tokens*, which are newly generated tokens from the previous step that have not yet been incorporated into the cache; and (3) *Refreshing tokens*, which are not currently being generated but are contextually influenced by the new predictions and thus require updated projections. Note that these definitions are used consistently throughout the paper and play a central role in the ablation analysis in Section 5.3.

While generating and caching tokens can be identified deterministically, refreshing tokens are selected based on their correlation with generating tokens, as detailed in Section 4.2. All remaining tokens are treated as cached. This categorization forms the foundation of our efficient decoding approach: instead of recomputing all token representations, cache-aware attention assigns computation only to tokens that truly require it.

*Cache-aware attention design.* As illustrated in Figure 2, cache-aware attention achieves efficiency by splitting the attention operation into two computation paths—one for active tokens and one for cached tokens. Active tokens project fresh queries, keys, and values, and compute attention over the union of both active and cached tokens. Cached tokens, by contrast, do not generate new projections and are not used as queries, but their stored key/value projections contribute as context during attention. This separation allows us to reuse representations where possible, without sacrificing the bidirectional nature of the model.

For each active token, attention is computed in two parts. In the first, queries attend to freshly computed keys and values from the active set ($K_A$, $V_A$); in the second, the same queries attend to cached keys and values from previous steps ($K_C$, $V_C$). To avoid data movement overhead from concatenation, we compute these two attention scores independently and merge the results using an *safe online softmax* formulation (8; 9; 37). With $\alpha^{(A)}, \alpha^{(C)}$, and $\ell_i$ from the online softmax, the output vector is computed as a sum of contributions from active and cached tokens, avoiding the need for key/value concatenation. This ensures exact equivalence to standard attention while enabling better memory locality and kernel fusion.

After attention, only active tokens are forwarded through the feed-forward network, while cached tokens are skipped entirely in this stage. Finally, the

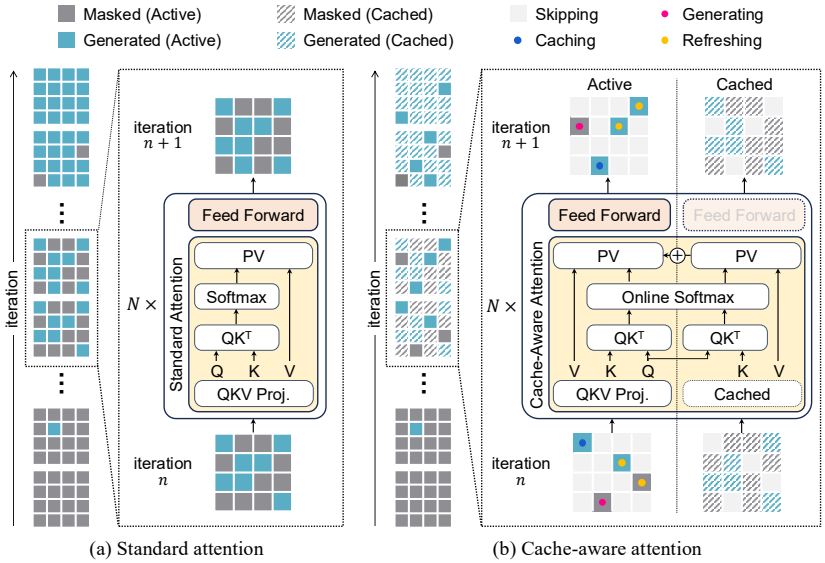


**Fig. 2: Comparison of standard attention and cache-aware attention in MAR generation. (a)** In standard attention, all tokens (masked or generated) are processed at every step, leading to redundant computation. **(b)** In cache-aware attention, only *active tokens* (generating, refreshing, or caching) are recomputed, while *cached tokens* reuse previously computed key/value projections.

key/value projections of active tokens are written back to the cache. These design choices of query-side separation, feed-forward skipping, and selective cache updates enable significant runtime savings without modifying the underlying transformer architecture. The full algorithm is summarized in Algorithm 1. We report a thorough analysis of speedups in Section 5.4.

### 4.2 Selective KV Cache Refresh

*Criteria for selecting refreshing tokens.* As discussed in Figure 1, although most tokens exhibit stable KV representations across steps, a subset of tokens plays a crucial role in maintaining image generation quality—particularly those influenced by newly generated content. Naturally, newly generated tokens must be recomputed. However, tokens that are influenced by the newly unmasked tokens may also need to be refreshed, as their importance can shift dynamically during generation.

To identify such tokens, we draw inspiration from autoregressive language models (1; 7; 12; 40; 48; 52), where attention scores are commonly used to determine which cached key/value entries can be safely discarded or ignored during inference. In contrast, we repurpose attention scores in our setting to do the opposite: to identify which tokens should be actively refreshed. By selecting tokens that receive high attention from newly generated tokens, we can target the refresh process toward contextually important tokens. This strategy enables us

**Algorithm 1** Cache-aware Attention

**Require:** Token embeddings $x^{(t)}$, cache $\mathcal{C}^{(t)}$, generation mask $M^{(t)}$
1: Identify generating tokens $G^{(t)} \subset M^{(t)}$
2: Identify caching tokens $N^{(t)}$ from previous step
3: Identify refreshing tokens $R^{(t)}$ based on correlation with $G^{(t)}$
4: Set active set $A^{(t)} = G^{(t)} \cup R^{(t)} \cup N^{(t)}$, cached set $C^{(t)} = \{1, \ldots, N\} \setminus A^{(t)}$
5: Compute $Q, K, V$ for $i \in A^{(t)}$; retrieve cached $K, V$ for $i \in C^{(t)}$
6: **for all** $i \in A^{(t)}$ **do**
7: Compute attention logits over $A^{(t)}$ and $C^{(t)}$, apply safe online softmax:

$$\alpha^{(A)}, \alpha^{(C)}, \ell_i = \texttt{SafeOnlineSoftmax}(q_i, K_A, K_C)$$

8: Compute output vector: $z_i = \frac{\alpha^{(A)} V_A + \alpha^{(C)} V_C}{\ell_i}$
9: Apply feed-forward: $x_i^{(t+1)} = \mathrm{FFN}(z_i^{(t)})$
10: **end for**
11: Update cache: $\mathcal{C}^{(t+1)}[i] = (K_i, V_i)$ for $i \in A^{(t)}$

to maintain image generation quality with minimal recomputation. We validate its effectiveness in Section 5.3.

*Design of selective KV cache refresh.* Building on our empirical findings, we design a selective KV cache refresh mechanism that dynamically identifies which tokens require recomputation based on their contextual relevance to the generating tokens. As illustrated in Figure 3, we leverage attention scores computed in the second decoder layer (Layer 2). Note that the choice of decoder layer for this selection is configurable, allowing for a trade-off between speedup and image quality, which we discuss further in Section 5.3.

At each generation step, Layers 1 and 2 perform standard full attention, as the active tokens are not yet finalized. During the Layer 2 attention computation, attention scores from generating tokens to all other tokens are computed and aggregated across all attention heads to obtain a global relevance score for each token. We then select the top-$K$ tokens with the highest relevance scores as the refreshing tokens. These refreshing tokens, together with the newly generated tokens and the previously cached tokens, form the active token set for subsequent layers. In these layers, cache-aware attention is applied: representations are recomputed only for active tokens, while the KV projections of non-active (cached) tokens are reused without reprocessing. This targeted refresh strategy enables the model to preserve contextual consistency while minimizing computational overhead.

### 4.3 Implementation Details

*Updating active tokens.* At each generation step, the active token set always includes tokens that are selected for generation in the current step (generating) and tokens selected in the immediately prior step (caching). The attention

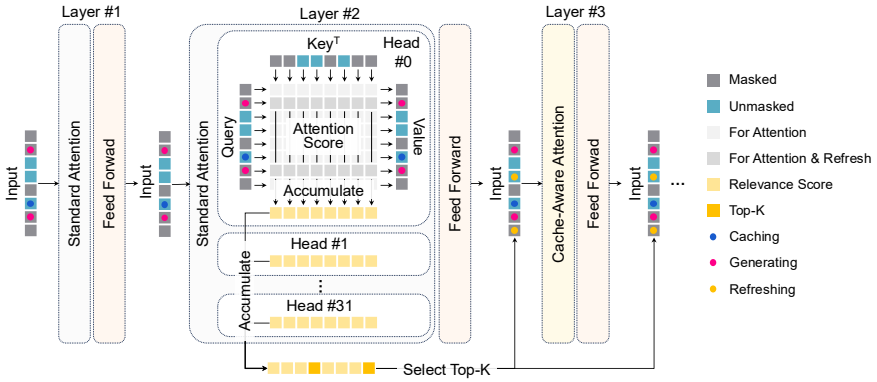


**Fig. 3: KV cache refresh via attention score.** Layers 1 and 2 perform standard full attention. In Layer 2, attention scores are computed from generating tokens to all others, and aggregated across attention heads to produce global relevance scores. The top-$K$ most contextually relevant tokens are selected based on these scores and marked as refreshing tokens. From Layer 3 onward, cache-aware attention is applied.

scores of the first decode layer are then used to select the top-K tokens (refreshing), such that the active set fits a fixed batch size of 64. For example, if there are 5 generating tokens and 9 caching tokens selected, we select the remaining 50 tokens as the refresh tokens based on the attention scores. The number of refreshing tokens is thus adjusted accordingly to fit within this budget.

*Refreshing entire KV Cache.* We refresh the entire KV cache in three specific cases to ensure correctness and stability. First, during the initial step, since no KV projections have been generated yet, we apply full attention across all layers. Second, in every step, the first and second decoder layers perform full attention as the active tokens are not yet finalized. Lastly, to mitigate potential value drift caused by repeated caching, we periodically refresh the entire KV cache every three steps, a process we refer to as periodic full refresh.

## 5 Evaluation

We evaluate MARché along three key dimensions: (1) image quality, to ensure that generation fidelity is preserved; (2) inference latency, to assess computational efficiency; and (3) design choices, to analyze how different components of our method impact performance.

### 5.1 Experimental Setup

We evaluate our proposed MARché framework on the ImageNet dataset at $256 \times 256$ resolution in the class-conditional setting. We adopt three model scales from the original MAR implementation (19): MAR-B (Base), MAR-L (Large),

**Table 1: Comparison of MARché and baseline models on ImageNet 256×256.** MARché achieves up to 1.72× speedup over the original MAR models while maintaining competitive image quality.

| Method | Latency (s/im) ↓ | FID ↓ | IS ↑ | Param | Speedup ↑ |
|---|---|---|---|---|---|
| MaskGIT (5) | 0.440 | 6.18 | 182.1 | 227M | - |
| DiT-XL/2 (28) | 0.787 | 2.27 | 278.2 | 675M | - |
| LlamaGen-XXL (39) | 0.897 | 3.09 | 253.6 | 1.4B | - |
| LlamaGen-3B (39) | 1.011 | 3.05 | 222.3 | 3.1B | - |
| MAR-B (19) | 0.104 | 2.35 | 281.1 | 208M | 1.00 |
| LazyMAR-B (49) | 0.074 | 5.32 | 235.1 | 208M | 1.41 |
| MARché-B | 0.064 | 2.56 | 270.3 | 208M | 1.57 |
| MAR-L (19) | 0.193 | 1.84 | 296.3 | 479M | 1.00 |
| LazyMAR-L (49) | 0.132 | 4.32 | 246.6 | 479M | 1.46 |
| MARché-L | 0.115 | 2.16 | 278.6 | 479M | 1.68 |
| MAR-H (19) | 0.336 | 1.62 | 298.6 | 943M | 1.00 |
| LazyMAR-H (49) | 0.220 | 4.00 | 251.9 | 943M | 1.52 |
| MARché-H | 0.188 | 2.02 | 281.4 | 943M | 1.79 |

and MAR-H (Huge). For each model, we construct a corresponding MARché variant (MARché-B, MARché-L, MARché-H) by applying our cache-aware inference strategy without modifying the model architecture or training procedure.

To assess image quality, we report Fréchet Inception Distance (FID) (14) and Inception Score (IS) (36). Computational efficiency is measured using the latency metric (in seconds per image). Speedup is calculated as the ratio of latency between the original MAR and its corresponding MARché variant at each model scale.

We compare MARché against several strong baselines: MaskGIT (5), a masked generative transformer; DiT (28), a diffusion transformer; LlamaGen (39), an autoregressive model; MAR (19); and LazyMAR (49), an efficiency-oriented MAR baseline. For a fair comparison within the MAR family, we evaluate MAR, LazyMAR, and MARché under identical settings, isolating the effect of inference-time acceleration strategies. All experiments are conducted on a single NVIDIA H100 GPU. We use a batch size of 128 for main evaluations and 256 for ablation studies. We follow the default MAR inference schedule of 64 decoding steps.

### 5.2 Main Results

Table 1 presents a comprehensive comparison of our MARché models against recent high-performing image generation baselines, including MaskGIT (5), DiT (28), and LlamaGen (39), as well as the original MAR models across three different scales.

Across all model sizes, MARché achieves substantial reductions in inference latency while maintaining competitive generation quality. For instance, MARché-

B reduces latency from 0.104s to 0.064s per image, achieving a 1.57× speedup over MAR-B, with negligible change in FID (from 2.35 to 2.56) and Inception Score (from 281.1 to 270.3). Similar trends are observed for larger models: MARché-L and MARché-H show 1.68× and 1.79× speedups, respectively, while keeping FID and IS close to their corresponding MAR baselines.

Compared to LazyMAR, MARché achieves a more favorable speed-quality trade-off across all scales. While LazyMAR provides moderate speedups (e.g., 1.41×–1.52×), it incurs a substantial degradation in generation quality (FID increases to 5.32/4.32/4.00 for B/L/H). In contrast, MARché delivers larger speedups (1.57×–1.79×) while preserving fidelity more effectively, as reflected by FID and IS remaining much closer to the original MAR models.

Beyond the MAR family, MARché is consistently faster than other masked or autoregressive baselines such as MaskGIT and LlamaGen under our evaluation setting, while achieving substantially better FID/IS than these methods. Although DiT attains slightly lower FID in some cases, it does so with significantly higher latency, highlighting the efficiency advantage of MARché.

Overall, MARché offers a compelling trade-off between generation speed and quality, outperforming baseline masked and autoregressive models in both inference efficiency and scalability. Importantly, these gains are achieved without modifying the original architecture or retraining, demonstrating the practical applicability of our approach.

**Table 2: Effect of token selection strategies for constructing the active set.** The results illustrate generation performance with respect to the inclusion of *generating tokens* and *caching tokens*, as defined in Section 4.1.

| Strategy | FID ↓ |
|---|---|
| MARché | **2.56** |
| MARché w/o *caching tokens* | 2.70 |
| MARché w/o *generating tokens* | 504.82 |
| Random selection | 564.61 |

**Table 3: Effect of refreshing token selection strategy.** Our MARché approach, which selects tokens receiving high attention from generating tokens, outperforms both low-attention and random selection strategies.

| Strategy | FID ↓ |
|---|---|
| MARché | **2.56** |
| MARché w/ low attention scores | 3.80 |
| MARché w/ random selection | 3.01 |

### 5.3 Ablation Study

*Ablation on token types in active set construction.* We validate the design of our active token selection strategy in MARché by comparing four different approaches in Table 2. The compared strategies are: (1) the full MARché method; (2) MARché without *caching tokens*; (3) MARché without *generating tokens*; (4) random selection of tokens for recomputation.

As shown in Table 2, the full MARché configuration achieves the best performance with an FID of 2.56. Including caching tokens results in some performance

improvement, and more importantly, that inclusion costs no additional computational burden. In contrast, excluding generating tokens leads to a substantial degradation in quality, confirming their essential role in preserving semantic consistency during decoding. Finally, using randomly selected tokens for recomputation yields the worst performance, highlighting the importance of guided, attention-based selection.

These results demonstrate that generating tokens are indispensable for accurate generation, while incorporating caching tokens offers additional benefits with minimal computational cost.

*Correlations of refreshing tokens and attention scores.* We evaluate the impact of different strategies for selecting *refreshing tokens*. As shown in Table 3, we compare three approaches.

The first strategy follows MARché's approach, selecting refreshing tokens with high attention scores to generating tokens. The second selects tokens with low attention scores to generating tokens, and the third randomly selects a subset of tokens for refreshing, ignoring contextual relevance.

As the results indicate, selecting tokens based on low attention scores leads to worse performance than random selection, with a clearly higher FID. This suggests that refreshing tokens should be chosen based on their strong correlation with generating tokens in order to maintain high image generation quality.

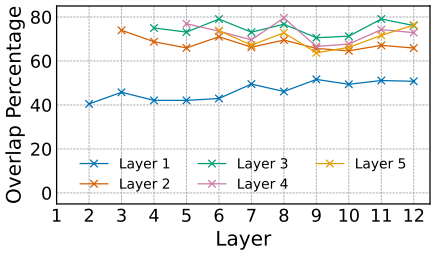


**Fig. 4: Top-$K$ attention overlap across decoder layers.** Deeper layers (3–5) show higher consistency overall, while Layer 1 exhibits much lower overlap (49.3%).

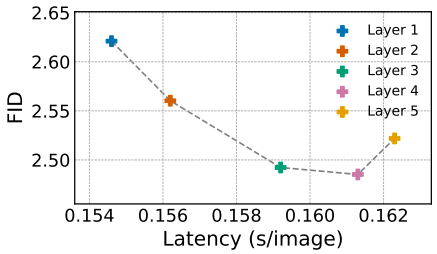


**Fig. 5: Latency–FID trade-off for different refreshing token selection layers.** Deeper layers improve FID with higher latency, while earlier layers decode faster but degrade quality.

*Choosing the decoder layer for refreshing token selection.* We analyze the impact of selecting different decoder layers for computing attention scores used in refreshing token selection. This decision involves a trade-off between computational efficiency and generation quality, as it affects both which tokens are refreshed and where full attention is computed.

Figure 4 shows the overlap ratio between layer 1-5 and all subsequent layers. As shown in the figure, deeper layers tend to produce refreshing token selections that align more closely with other layers. Specifically, Layer 3 achieves the highest

average overlap of 74.4% with the rest, while Layer 1 shows lower alignment at 49.3%. Layer 2 exhibits a moderate level of agreement at 66.8%, suggesting it captures context reasonably well while still remaining computationally efficient.

This trade-off is also reflected in generation performance. Figure 5 shows that using Layer 1 results in faster decoding (0.154 s/image) but yields a relatively high FID of 2.62. On the other hand, selecting a deeper layer like Layer 4 improves FID to 2.49 but with increased latency (0.1613 s/image). Layer 2 provides a balanced compromise, attaining an FID of 2.56 with moderate latency (0.1593 s/image).

In our experiments, we adopt Layer 2 as a default, as it provides a favorable trade-off between speed and quality while maintaining reasonable alignment with other layers' token selection.

### 5.4 Analysis of Cache-Aware Attention

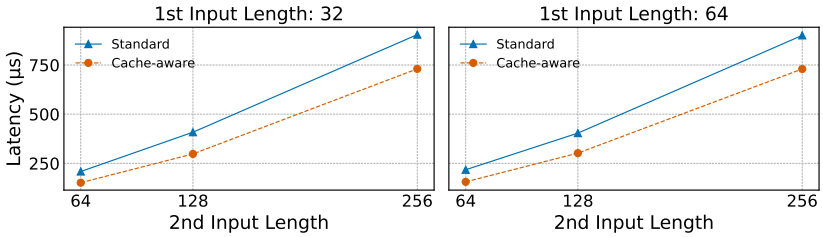


**Fig. 6: Runtime comparison of standard attention vs. cache-aware attention.** Both methods process the same total number of attention operations over two input segments of different lengths. *Standard attention* concatenates the inputs and processes them jointly, while *cache-aware attention* handles them separately.

*Runtime efficiency of cache-aware attention implementation.* Cache-aware attention takes $q_i$ with active $(K_A, V_A)$ and cached $(K_C, V_C)$. A vanilla approach concatenates $K_A$ with $K_C$ (and $V_A$ with $V_C$) and applies a standard attention kernel, which we find suboptimal. Assuming $q_i, K_A, V_A, K_C, V_C$ are all preloaded, we benchmark this baseline against our fused cache-aware kernel.

Figure 6 shows that our kernel consistently reduces latency despite the same total attention workload. For example, with input lengths 32 and 256, latency drops from 904.2$\mu$s to 730.3$\mu$s (19.2%), and overall reductions range from 16.2% to 27.3%. The gains mainly come from improved memory locality: we avoid explicit concatenation and process active and cached tokens on separate paths, gathering only active tokens contiguously and reducing non-contiguous memory accesses.

*Speedup contributions of cache-aware attention.* Cache-aware attention provides latency gains through efficient memory management and the reuse of cached keys and values. In addition, it enables skipping part of the feed-forward layer (FFN) computation, since cached tokens do not require regeneration. We refer to this as FFN skipping. Therefore, we identify two components contributing to the overall speedup: (1) cache-aware attention itself and (2) FFN skipping. To disentangle their contributions, we implement four different methods, each utilizing either cache-aware attention, FFN skipping, or both, and evaluate MAR with each implementation.

**Table 4: Latency and speedup comparison across different implementations**, isolating the effects of cache-aware attention and FFN skipping.

| Method | Implementation | Latency (s) | Speedup |
|---|---|---|---|
| MAR | Standard attention + full FFN | 0.104 | 1× |
| MAR + selected FFN | Standard attention + FFN skipping | 0.092 | 1.12× |
| MAR + cache-aware Attn | Cache-aware attention + full FFN | 0.067 | 1.55× |
| MARché | Cache-aware attention + FFN skipping | 0.064 | 1.63× |

Table 4 reports the latency and speedup of each implementation. The results indicate that cache-aware attention contributes primarily to the speedup of MARché, while FFN skipping also provides a smaller yet meaningful improvement. Based on these results, we confirm that cache-aware attention significantly reduces the computation path for the cached token set and enables more efficient memory management, making it the key factor behind the observed speedup.

## 6 Conclusion

We presented MARché, a cache-aware decoding framework for masked autoregressive image generation that significantly reduces inference latency while maintaining high image quality. By selectively refreshing only contextually relevant tokens based on attention scores, MARché avoids redundant computation and achieves up to 1.7× speedup across model scales without modifying the original architecture. Our approach highlights the potential of combining autoregressive modeling with efficient token reuse strategies in high-resolution image generation. While our method focuses on masked autoregressive models, the core ideas behind cache-aware attention and selective refresh may extend to other generative settings, including multi-modal generation.

## Acknowledgment

We sincerely thank all the reviewers for their time and constructive comments. This material is based upon work supported by NSF award number 2224319, REAL@USC-Meta center, and VMware gift.

# Bibliography


[1] Adnan, M., Arunkumar, A., Jain, G., Nair, P.J., Soloveychik, I., Kamath, P.: Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. Proceedings of Machine Learning and Systems **6**, 114–127 (2024)

[2] Agarwal, S., Mitra, S., Chakraborty, S., Karanam, S., Mukherjee, K., Saini, S.K.: Approximate caching for efficiently serving {Text-to-Image} diffusion models. In: 21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). pp. 1173–1189 (2024)

[3] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020)

[4] Chang, H., Zhang, H., Barber, J., Maschinot, A., Lezama, J., Jiang, L., Yang, M.H., Murphy, K., Freeman, W.T., Rubinstein, M., et al.: Muse: Text-to-image generation via masked generative transformers. arXiv preprint arXiv:2301.00704 (2023)

[5] Chang, H., Zhang, H., Jiang, L., Liu, C., Freeman, W.T.: Maskgit: Masked generative image transformer. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11315–11325 (2022)

[6] Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: International conference on machine learning. pp. 1691–1703. PMLR (2020)

[7] Chen, R., Wang, Z., Cao, B., Wu, T., Zheng, S., Li, X., Wei, X., Yan, S., Li, M., Liang, Y.: Arkvale: Efficient generative llm inference with recallable key-value eviction. Advances in Neural Information Processing Systems **37**, 113134–113155 (2024)

[8] Dao, T.: Flashattention-2: Faster attention with better parallelism and work partitioning. arXiv preprint arXiv:2307.08691 (2023)

[9] Dao, T., Fu, D., Ermon, S., Rudra, A., Ré, C.: Flashattention: Fast and memory-efficient exact attention with io-awareness. Advances in neural information processing systems **35**, 16344–16359 (2022)

[10] Ding, M., Yang, Z., Hong, W., Zheng, W., Zhou, C., Yin, D., Lin, J., Zou, X., Shao, Z., Yang, H., et al.: Cogview: Mastering text-to-image generation via transformers. Advances in neural information processing systems **34**, 19822–19835 (2021)

[11] Esser, P., Rombach, R., Ommer, B.: Taming transformers for high-resolution image synthesis. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 12873–12883 (2021)

[12] Ge, S., Zhang, Y., Liu, L., Zhang, M., Han, J., Gao, J.: Model tells you what to discard: Adaptive kv cache compression for llms. arXiv preprint arXiv:2310.01801 (2023)

[13] Goyal, S., Tula, D., Jain, G., Shenoy, P., Jain, P., Paul, S.: Masked generative nested transformers with decode time scaling. arXiv preprint arXiv:2502.00382 (2025)
[14] Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., Hochreiter, S.: Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems **30** (2017)
[15] Jang, D., Park, S., Yang, J.Y., Jung, Y., Yun, J., Kundu, S., Kim, S.Y., Yang, E.: Lantern: Accelerating visual autoregressive models with relaxed speculative decoding. arXiv preprint arXiv:2410.03355 (2024)
[16] Lee, D., Kim, C., Kim, S., Cho, M., Han, W.S.: Autoregressive image generation using residual quantization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11523–11532 (2022)
[17] Lezama, J., Chang, H., Jiang, L., Essa, I.: Improved masked image generation with token-critic. In: European Conference on Computer Vision. pp. 70–86. Springer (2022)
[18] Li, T., Chang, H., Mishra, S., Zhang, H., Katabi, D., Krishnan, D.: Mage: Masked generative encoder to unify representation learning and image synthesis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2142–2152 (2023)
[19] Li, T., Tian, Y., Li, H., Deng, M., He, K.: Autoregressive image generation without vector quantization. Advances in Neural Information Processing Systems **37**, 56424–56445 (2024)
[20] Liu, E., Ning, X., Lin, Z., Yang, H., Wang, Y.: Oms-dpm: Optimizing the model schedule for diffusion probabilistic models. In: International Conference on Machine Learning. pp. 21915–21936. PMLR (2023)
[21] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. Advances in Neural Information Processing Systems **35**, 5775–5787 (2022)
[22] Lu, C., Zhou, Y., Bao, F., Chen, J., Li, C., Zhu, J.: Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. arXiv preprint arXiv:2211.01095 (2022)
[23] Ma, X., Fang, G., Wang, X.: Deepcache: Accelerating diffusion models for free. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 15762–15772 (2024)
[24] Ni, Z., Wang, Y., Zhou, R., Han, Y., Guo, J., Liu, Z., Yao, Y., Huang, G.: Enat: Rethinking spatial-temporal interactions in token-based image synthesis. Advances in Neural Information Processing Systems **37**, 90431–90455 (2024)
[25] Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A., et al.: Conditional image generation with pixelcnn decoders. Advances in neural information processing systems **29** (2016)
[26] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. Advances in neural information processing systems **35**, 27730–27744 (2022)

[27] Parmar, N., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: International conference on machine learning. pp. 4055–4064. PMLR (2018)
[28] Peebles, W., Xie, S.: Scalable diffusion models with transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4195–4205 (2023)
[29] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PmLR (2021)
[30] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
[31] Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
[32] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. Journal of machine learning research **21**(140), 1–67 (2020)
[33] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-shot text-to-image generation. In: International conference on machine learning. pp. 8821–8831. Pmlr (2021)
[34] Razavi, A., Van den Oord, A., Vinyals, O.: Generating diverse high-fidelity images with vq-vae-2. Advances in neural information processing systems **32** (2019)
[35] Reed, S., Oord, A., Kalchbrenner, N., Colmenarejo, S.G., Wang, Z., Chen, Y., Belov, D., Freitas, N.: Parallel multiscale autoregressive density estimation. In: International conference on machine learning. pp. 2912–2921. PMLR (2017)
[36] Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., Chen, X.: Improved techniques for training gans. Advances in neural information processing systems **29** (2016)
[37] Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., Dao, T.: Flashattention-3: Fast and accurate attention with asynchrony and low-precision. Advances in Neural Information Processing Systems **37**, 68658–68685 (2024)
[38] Shen, Z., Zhang, M., Zhao, H., Yi, S., Li, H.: Efficient attention: Attention with linear complexities. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 3531–3539 (2021)
[39] Sun, P., Jiang, Y., Chen, S., Zhang, S., Peng, B., Luo, P., Yuan, Z.: Autoregressive model beats diffusion: Llama for scalable image generation. arXiv preprint arXiv:2406.06525 (2024)
[40] Tang, J., Zhao, Y., Zhu, K., Xiao, G., Kasikci, B., Han, S.: Quest: Query-aware sparsity for efficient long-context llm inference. arXiv preprint arXiv:2406.10774 (2024)

[41] Teng, Y., Shi, H., Liu, X., Ning, X., Dai, G., Wang, Y., Li, Z., Liu, X.: Accelerating auto-regressive text-to-image generation with training-free speculative jacobi decoding. arXiv preprint arXiv:2410.01699 (2024)
[42] Tian, K., Jiang, Y., Yuan, Z., Peng, B., Wang, L.: Visual autoregressive modeling: Scalable image generation via next-scale prediction. Advances in neural information processing systems **37**, 84839–84865 (2024)
[43] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)
[44] Van Den Oord, A., Vinyals, O., et al.: Neural discrete representation learning. Advances in neural information processing systems **30** (2017)
[45] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
[46] Weber, M., Yu, L., Yu, Q., Deng, X., Shen, X., Cremers, D., Chen, L.C.: Maskbit: Embedding-free image generation via bit tokens. arXiv preprint arXiv:2409.16211 (2024)
[47] Wimbauer, F., Wu, B., Schoenfeld, E., Dai, X., Hou, J., He, Z., Sanakoyeu, A., Zhang, P., Tsai, S., Kohler, J., et al.: Cache me if you can: Accelerating diffusion models through block caching. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6211–6220 (2024)
[48] Xiao, G., Tian, Y., Chen, B., Han, S., Lewis, M.: Efficient streaming language models with attention sinks. arXiv preprint arXiv:2309.17453 (2023)
[49] Yan, F., Wei, Q., Tang, J., Li, J., Wang, Y., Hu, X., Li, H., Zhang, L.: Lazymar: Accelerating masked autoregressive models via feature caching. arXiv preprint arXiv:2503.12450 (2025)
[50] Yu, J., Xu, Y., Koh, J.Y., Luong, T., Baid, G., Wang, Z., Vasudevan, V., Ku, A., Yang, Y., Ayan, B.K., et al.: Scaling autoregressive models for content-rich text-to-image generation. arXiv preprint arXiv:2206.10789 **2**(3), 5 (2022)
[51] Yuan, Z., Zhang, H., Pu, L., Ning, X., Zhang, L., Zhao, T., Yan, S., Dai, G., Wang, Y.: Ditfastattn: Attention compression for diffusion transformer models. Advances in Neural Information Processing Systems **37**, 1196–1219 (2024)
[52] Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al.: H2o: Heavy-hitter oracle for efficient generative inference of large language models. Advances in Neural Information Processing Systems **36**, 34661–34710 (2023)
[53] Zhao, T., Fang, T., Huang, H., Liu, E., Wan, R., Soedarmadji, W., Li, S., Lin, Z., Dai, G., Yan, S., et al.: Vidit-q: Efficient and accurate quantization of diffusion transformers for image and video generation. arXiv preprint arXiv:2406.02540 (2024)
[54] Zhao, T., Ning, X., Fang, T., Liu, E., Huang, G., Lin, Z., Yan, S., Dai, G., Wang, Y.: Mixdq: Memory-efficient few-step text-to-image diffusion models

with metric-decoupled mixed precision quantization. In: European Conference on Computer Vision. pp. 285–302. Springer (2024)