RESCENEKV: RECONSTRUCTION-ERROR SELECTIVE KV CACHE EVICTION ACROSS NESTED SCALES FOR VISUAL AUTOREGRESSIVE MODELING

Anonymous authors
Paper under double-blind review



Figure 1: Qualitative results using ResceneKV on Infinity-8B (Han et al., 2025) with only 4% of the KV cache retained. KV cache memory is measured at a batch size of 4.

ABSTRACT

Visual autoregressive (VAR) modeling generates images by progressively predicting token maps from coarse to fine, through a next-scale paradigm that enables efficient sampling and strong zero-shot generalization. However, predicting each scale from all preceding scales requires retaining the corresponding keys and values in the KV cache. As generation proceeds to finer scales, the KV cache therefore grows continuously and becomes a major memory bottleneck for high-resolution image synthesis. Existing KV cache eviction methods typically rely on attention statistics to guide eviction decisions, although eviction candidates with similar attention mass can exhibit substantially different reconstruction errors. To address this mismatch, we propose ResceneKV, which directly measures the output reconstruction error induced by KV cache eviction and uses it for joint KV cache budget allocation. We further exploit the redundancy between the conditional and unconditional branches of classifier-free guidance by reusing the conditional KV cache for the unconditional branch at later scales. Extensive experiments across image generation benchmarks demonstrate that ResceneKV consistently improves pixel-level fidelity over existing KV cache eviction methods while maintaining generation quality across different memory budgets.

1 INTRODUCTION

Visual autoregressive (VAR) (Tian et al., 2024) modeling reformulates autoregressive image generation (Sun et al., 2024; Yao et al., 2024; Li et al., 2024b) into a next-scale paradigm, representing an image as successively finer token maps. Conventional next-token autoregressive generation (Esser et al., 2021; Lee et al., 2022; Yu et al., 2022) decodes individual tokens sequentially. In contrast,

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.

VAR predicts an entire token map at each scale in parallel, substantially reducing decoding steps while enabling zero-shot generalization to tasks such as inpainting and editing. Infinity (Han et al., 2025) extends this paradigm to high-resolution text-to-image generation by adopting a bitwise autoregressive formulation with a self-correction mechanism.

The next-scale paradigm, however, leads to a rapidly growing KV cache during generation. To predict a new scale, the model attends to all previously generated token maps, requiring their keys and values to remain in the KV cache for subsequent scales. This accumulation becomes particularly pronounced at finer scales, which account for the majority of visual tokens and thereby increase the computational cost of self-attention. For a final token map of spatial size $n \times n$, the number of visual tokens scales as $\mathcal{O}(n^2)$, resulting in $\mathcal{O}(n^4)$ complexity for self-attention. For Infinity-8B at batch size 4, the KV cache requires 42 GB of memory, compared with 16.8 GB for the model weights. Moreover, unlike the model weights, the KV cache scales with batch size, directly limiting parallel image generation.

To reduce KV cache memory in VAR, recent studies (Li et al., 2025; Qin et al., 2026; Xu et al., 2026; Cederlund et al., 2026) have explored KV cache eviction based on attention statistics. However, attention mass does not directly reflect the reconstruction error induced by eviction. Fig. 2(a) shows that similar attention mass can lead to substantially different reconstruction errors, revealing a mismatch between attention mass and the actual impact of eviction. We further observe substantial redundancy between the conditional and unconditional branches of Classifier-Free Guidance (CFG) (Ho & Salimans, 2022; Nichol et al., 2021). Their keys and values become increasingly similar at later scales, as illustrated in Fig. 2(c), making separate KV caches progressively redundant.

To address this mismatch, we propose ResceneKV, which directly measures the output reconstruction error induced by KV cache eviction rather than relying on attention-based importance. Based on these reconstruction errors, ResceneKV jointly allocates the limited KV cache budget across layers, attention heads, source scales, and generation scales. We further exploit the high cross-branch similarity at later scales by sharing the conditional KV cache with the unconditional branch. The reconstruction error induced by cache sharing is also incorporated into the same budget allocation framework, allowing both eviction and sharing to be considered under a unified memory constraint. Our contributions are summarized as follows:

- We show that similar attention mass can correspond to markedly different reconstruction errors. Based on this observation, ResceneKV evaluates candidate eviction sets using reconstruction error and jointly allocates the KV cache budget across layers, attention heads, source scales, and generation scales.
- We exploit the increasing cross-branch similarity at later scales by sharing the conditional KV cache with the unconditional branch, reducing redundant KV cache storage across the two CFG branches.
- Extensive experiments show that ResceneKV improves pixel-level fidelity over existing methods while maintaining generation quality under constrained budgets.

2 Related Work

Autoregressive Visual Generation. Autoregressive image generation has progressed from sequential pixel prediction in raster order (Van Den Oord et al., 2016; Chen et al., 2020) to autoregressive modeling over discrete visual tokens, enabled by tokenizers such as VQ-VAE (Van Den Oord et al., 2017) and VQ-GAN (Esser et al., 2021). Despite this compact representation, next-token prediction still requires a large number of sequential decoding steps. VAR (Tian et al., 2024) departs from this paradigm by reformulating image generation as coarse-to-fine next-scale prediction, where each step predicts an entire token map in parallel conditioned on all preceding scales. This shift has enabled further advances in large-scale and high-resolution generation. HART (Tang et al., 2024) combines discrete autoregressive generation with a lightweight diffusion module that models continuous residual information. Infinity (Han et al., 2025) further extends visual autoregressive modeling through bitwise token prediction with a scalable tokenizer for high-resolution text-to-image synthesis. Although next-scale prediction substantially reduces the number of decoding steps, each finer scale attends to all preceding scales, requiring their keys and values to remain in the KV cache and causing the cache to grow as generation proceeds.

KV Cache Eviction. KV cache eviction has been widely studied for LLM inference (Zhang et al., 2023; Wan et al., 2024; Qin et al., 2025), with representative methods preserving attention sinks and recent tokens (Xiao et al., 2024), selecting tokens based on attention scores (Li et al., 2024a), or allocating non-uniform budgets across layers (Cai et al., 2024). However, these approaches do not fully capture the multi-scale structure of VAR, in which cached representations are organized by source scale and reused across subsequent generation scales. Recent studies on KV cache eviction for VAR therefore explicitly account for this multi-scale structure. ScaleKV (Li et al., 2025) classifies layers as drafters or refiners based on scale-dependent attention patterns, whereas AMS-KV (Xu et al., 2026) adjusts layer-wise cache budgets based on inter-scale KV similarity and prioritizes condensed and local scales for retention. HACK (Qin et al., 2026) exploits distinct roles of attention heads by classifying them as contextual or structural and allocating asymmetric budgets according to head type. HeatKV (Cederlund et al., 2026) further ranks and evicts individual head-scale pairs based on scale-dependent attention statistics. While these methods guide eviction using attention patterns, statistics, or KV similarity, they do not directly quantify the output error induced by eviction. In contrast, our method directly evaluates eviction-induced reconstruction error.

3 PRELIMINARIES

Next-scale Prediction. VAR (Tian et al., 2024) adopts a next-scale paradigm, predicting token maps progressively from coarse to fine. A multi-scale tokenizer encodes an image feature map $f \in \mathbb{R}^{h\times w\times C}$ into K token maps $r_1, \ldots, r_K$ of increasing resolution, where $n_k = h_k \times w_k$ denotes the number of tokens in r_k, and the finest map r_K has spatial resolution $h_K \times w_K$. The autoregressive likelihood factorizes over these maps as

$$p(r_1, \ldots, r_K) \;=\; \prod_{k=1}^{K} p(r_k \,|\, r_1, \ldots, r_{k-1},\, \Psi)\,, \tag{1}$$

where Ψ denotes the text condition. At generation scale k, the model predicts all n_k tokens in the token map r_k in parallel, conditioned on the preceding maps $r_1, \ldots, r_{k-1}$.

Scale-wise KV Caching. The keys and values computed for each generated scale are retained for subsequent scales, allowing later scales to attend to earlier ones without recomputation. Let $X_k \in \mathbb{R}^{n_k\times d}$ denote the hidden representations at generation scale k. The corresponding keys and values are computed as $\mathbf{K}_k = X_k W_K$ and $\mathbf{V}_k = X_k W_V$, where $W_K, W_V \in \mathbb{R}^{d\times d}$ are the key and value projection matrices. For layer l and head h, let $\mathbf{K}_i^{l,h}$ and $\mathbf{V}_i^{l,h}$ denote the keys and values from source scale i. The KV cache at generation scale k is then given by

$$\mathbf{K}_{1:k-1}^{l,h} = \left[\mathbf{K}_1^{l,h}; \mathbf{K}_2^{l,h}; \ldots; \mathbf{K}_{k-1}^{l,h}\right], \qquad \mathbf{V}_{1:k-1}^{l,h} = \left[\mathbf{V}_1^{l,h}; \mathbf{V}_2^{l,h}; \ldots; \mathbf{V}_{k-1}^{l,h}\right]. \tag{2}$$

Thus, for each layer–head pair, the key cache contains $C_{k-1} = \sum_{i=1}^{k-1} n_i$ key vectors at generation scale k, and the value cache contains the same number of value vectors. With CFG, the conditional and unconditional branches maintain separate KV caches for the preceding scales. Since the final scale K is not reused by any subsequent generation scale, the cache budget covers only source scales $1, \ldots, K-1$. Let $m_{l,h,i}$ denote the cache allocation for source scale i at head h of layer l. Given a cache budget $b \in (0, 1]$, the total cache allocation is constrained by

$$\sum_{l=1}^{L}\sum_{h=1}^{H}\sum_{i=1}^{K-1} m_{l,h,i} \le b \cdot 2LHC_{K-1}. \tag{3}$$

4 RESCENEKV

Notation. For notational simplicity, we consider a single layer l and attention head h and omit these indices. At generation scale k, let $\mathbf{Q}_k, \mathbf{K}_k, \mathbf{V}_k \in \mathbb{R}^{n_k\times d_h}$ denote the query, key, and value representations of the current scale. Following Sec. 3, the representations from the preceding source scales constitute the KV cache $\mathbf{K}_{1:k-1}, \mathbf{V}_{1:k-1} \in \mathbb{R}^{C_{k-1}\times d_h}$. The attention output at scale k is

$$\mathbf{O}_k = \mathbf{A}_k\left[\mathbf{V}_{1:k-1}; \mathbf{V}_k\right], \qquad \mathbf{A}_k = \mathrm{softmax}\left(\frac{\mathbf{Q}_k\left[\mathbf{K}_{1:k-1}; \mathbf{K}_k\right]^\top}{\sqrt{d_h}}\right). \tag{4}$$

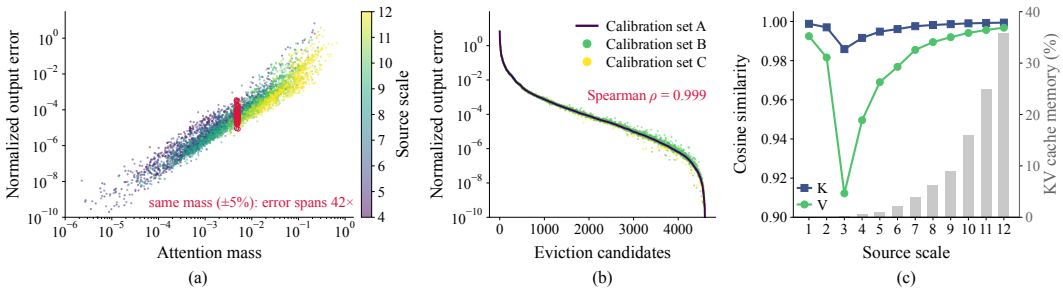

Figure 2: (a) Similar attention mass can induce substantially different reconstruction errors. (b) Eviction candidates sorted by reconstruction error on calibration set A show highly consistent errors on sets B and C. (c) Later scales exhibit high cross-branch similarity in keys and values while accounting for a large fraction of KV cache memory.

To characterize how the queries at generation scale k attend to each preceding source scale, we partition $\mathbf{A}_k$ according to the corresponding source scales:

$$\mathbf{A}_k = [\mathbf{A}_{k,1}, \mathbf{A}_{k,2}, \ldots, \mathbf{A}_{k,k-1}, \mathbf{A}_{k,k}], \tag{5}$$

where $\mathbf{A}_{k,i} \in \mathbb{R}^{n_k \times n_i}$ denotes the attention weights from generation scale k to source scale i.

4.1 Motivations

Limitations of Attention-based Eviction. Existing KV cache eviction methods (Li et al., 2025; Qin et al., 2026; Cederlund et al., 2026) commonly rank eviction candidates using attention statistics. However, attention mass does not directly reflect the output distortion induced by eviction. As shown in Fig. 2(a), eviction candidates with similar attention mass exhibit reconstruction errors that differ by up to a factor of 42. In VAR, we further observe that eviction-induced reconstruction errors are highly consistent across calibration sets. Fig. 2(b) orders eviction candidates according to the reconstruction errors measured on calibration set A and shows that the errors measured on sets B and C remain strongly correlated with those on set A, with a Spearman correlation of $\rho = 0.999$. This consistency supports reusing eviction selections from offline calibration during inference.

KV Cache Redundancy under Classifier-Free Guidance. CFG (Ho & Salimans, 2022) maintains separate KV caches for the conditional and unconditional branches, requiring separate KV cache storage for the two branches. We measure the cross-branch similarity of the corresponding keys and values using cosine similarity. As shown in Fig. 2(c), cross-branch similarity is particularly high at finer scales, while KV cache memory is heavily concentrated at these scales, with the three finest scales comprising 77% of the cached tokens. Since finer scales exhibit both high cross-branch similarity and large memory consumption, maintaining separate caches for both branches becomes increasingly redundant. Based on this observation, we share the conditional KV cache with the unconditional branch at later scales.

4.2 Reconstruction-error guided KV cache eviction

Motivated by the mismatch observed in Sec. 4.1, we directly evaluate the reconstruction error induced by KV cache eviction. Since multiple source scales may be evicted jointly, we represent the removed scales by an eviction set $\mathcal{D}$ and evaluate the resulting output distortion after renormalizing the remaining attention weights. At generation scale k, we consider a single query and let $\mathbf{a}_i \in \mathbb{R}^{1 \times n_i}$ denote the corresponding row of $\mathbf{A}_{k,i}$ for source scale $i < k$. The attention mass assigned to source scale i is $\beta_i = \sum_{j=1}^{n_i} [\mathbf{a}_i]_j$, and $\mathbf{V}_i \in \mathbb{R}^{n_i \times d_h}$ denotes its values.

Proposition 4.1 (Eviction-induced Reconstruction Error). *For an eviction set $\mathcal{D} \subseteq \{1, \ldots, k-1\}$, let $\mathbf{o}$ and $\widetilde{\mathbf{o}}$ denote the attention outputs before and after eviction. The reconstruction error is*

$$\mathcal{E}_{\mathcal{D}}^{\text{evict}} = \|\mathbf{o} - \widetilde{\mathbf{o}}\|_2^2 = \underbrace{\left(\frac{\beta_{\mathcal{D}}}{1-\beta_{\mathcal{D}}}\right)^2}_{\substack{\text{mass-dependent} \\ \text{factor}}} \underbrace{\|\bar{\mathbf{v}}_{\mathcal{D}} - \mathbf{o}\|_2^2}_{\substack{\text{value-dependent} \\ \text{discrepancy}}}, \quad \beta_{\mathcal{D}} = \sum_{i \in \mathcal{D}} \beta_i, \quad \bar{\mathbf{v}}_{\mathcal{D}} = \frac{\sum_{i \in \mathcal{D}} \mathbf{a}_i \mathbf{V}_i}{\beta_{\mathcal{D}}}. \tag{6}$$

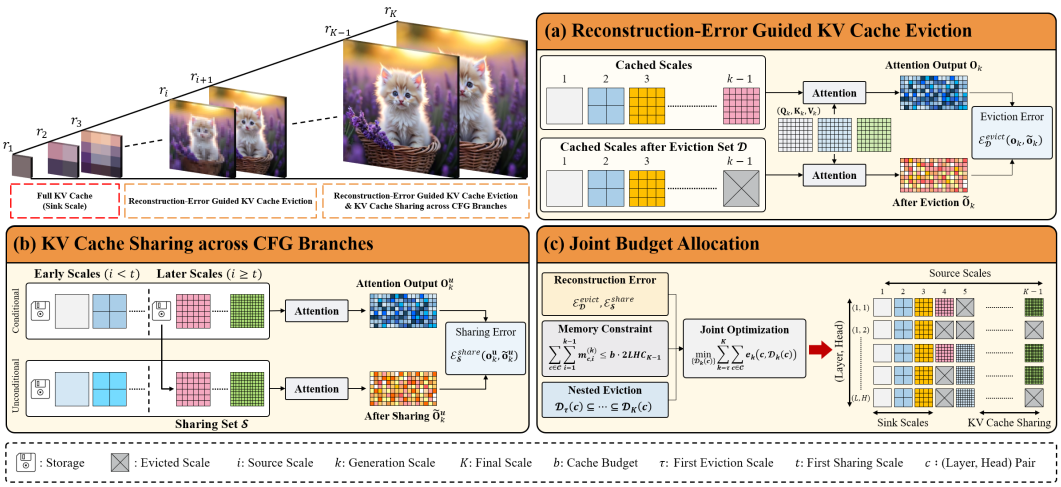

Figure 3: Overview of ResceneKV. (a) Reconstruction-error guided eviction measures the output distortion induced by each candidate eviction set. (b) KV cache sharing reuses the conditional KV cache for the unconditional branch at source scales $i \geq t$ while accounting for sharing-induced error. (c) Joint budget allocation determines the cache configuration under the KV cache budget while protecting sink scales from eviction and enforcing nested eviction across generation scales.

The derivation of Proposition 4.1 is provided in Appendix A.1. Eq. 6 shows that the reconstruction error is governed by two factors. The first reflects the magnitude of the output perturbation caused by removing attention mass and renormalizing the remaining attention weights. The second captures how different the attention-weighted average value of the evicted scales is from the original attention output. Existing KV cache eviction methods that rely on attention mass do not account for this value-dependent discrepancy. Consequently, eviction candidates with similar attention mass can still produce substantially different reconstruction errors. This result explains the mismatch between attention mass and reconstruction error observed in Fig. 2(a). Accordingly, we formulate the joint budget allocation in Sec. 4.4 based on reconstruction error rather than attention mass.

4.3 KV Cache Sharing across CFG branches

We exploit the cross-branch redundancy observed in Sec. 4.1 by sharing the conditional KV cache with the unconditional branch for source scales $i \geq t$, where t is the first source scale for sharing. Accordingly, we define $\mathcal{S} = \{i \mid t \leq i < K\}$ as the set of source scales subject to sharing, while separate KV caches are maintained for $i < t$. At generation scale k, only source scales in $\mathcal{S}$ with $i < k$ are available for sharing. Since differences in the keys and values of the two branches can still induce reconstruction error, we formulate the resulting error for subsequent budget allocation.

For a single unconditional query, let $\mathbf{a}_i^u \in \mathbb{R}^{1\times n_i}$ denote the attention weights over source scale i, and let $\mathbf{V}_i^u \in \mathbb{R}^{n_i\times d_h}$ denote its values before sharing. For $i \in \mathcal{S}$, we set $(\widetilde{\mathbf{K}}_i^u, \widetilde{\mathbf{V}}_i^u) = (\mathbf{K}_i^c, \mathbf{V}_i^c)$ and let $\widetilde{\mathbf{a}}_i^u$ denote the weights computed from $\widetilde{\mathbf{K}}_i^u$ under the pre-sharing softmax normalization. The corresponding attention masses for source scale i are $\beta_i^u = \sum_{j=1}^{n_i}[\mathbf{a}_i^u]_j$ and $\widetilde{\beta}_i^u = \sum_{j=1}^{n_i}[\widetilde{\mathbf{a}}_i^u]_j$.

Proposition 4.2 (Sharing-induced Reconstruction Error)**.** *For a sharing set $\mathcal{S}$, let $\mathbf{o}^u$ and $\widetilde{\mathbf{o}}^u$ denote the unconditional attention outputs before and after sharing. The pre-sharing attention mass and attention-weighted average value over $\mathcal{S}$ are $\beta_{\mathcal{S}}^u = \sum_{i\in\mathcal{S}}\beta_i^u$ and $\bar{\mathbf{v}}_{\mathcal{S}}^u = \left(\sum_{i\in\mathcal{S}}\mathbf{a}_i^u\mathbf{V}_i^u\right)/\beta_{\mathcal{S}}^u$, while the corresponding quantities under the pre-sharing normalization are $\widetilde{\beta}_{\mathcal{S}}^u = \sum_{i\in\mathcal{S}}\widetilde{\beta}_i^u$ and $\widetilde{\bar{\mathbf{v}}}_{\mathcal{S}}^u = \left(\sum_{i\in\mathcal{S}}\widetilde{\mathbf{a}}_i^u\widetilde{\mathbf{V}}_i^u\right)/\widetilde{\beta}_{\mathcal{S}}^u$. The reconstruction error induced by sharing over $\mathcal{S}$ is*

$$\mathcal{E}_{\mathcal{S}}^{\text{share}} = \|\mathbf{o}^u - \widetilde{\mathbf{o}}^u\|_2^2 = \frac{\left\|\beta_{\mathcal{S}}^u\left(\bar{\mathbf{v}}_{\mathcal{S}}^u - \mathbf{o}^u\right) - \widetilde{\beta}_{\mathcal{S}}^u\left(\widetilde{\bar{\mathbf{v}}}_{\mathcal{S}}^u - \mathbf{o}^u\right)\right\|_2^2}{\left(1 - \beta_{\mathcal{S}}^u + \widetilde{\beta}_{\mathcal{S}}^u\right)^2}. \tag{7}$$

The derivation of Proposition 4.2 is provided in Appendix A.2. Unlike eviction, which reduces memory by removing selected source scales from the KV cache, KV cache sharing reuses the conditional KV cache for the unconditional branch without removing those source scales. The effects of both mechanisms are therefore incorporated into the joint reconstruction error used for budget allocation.

4.4 Joint Budget Allocation

Under a fixed KV cache budget, the cache configuration cannot be determined independently at each generation scale. Once a source scale is evicted, it remains unavailable at all subsequent scales, coupling the eviction decisions throughout the generation process. KV cache sharing also changes the memory required for source scales with $i \geq t$. We therefore jointly optimize the cache allocation across layers, attention heads, source scales, and generation scales to minimize the accumulated reconstruction error under the memory constraint.

Let $\mathcal{C} = \{c = (l, h) \mid 1 \leq l \leq L,\ 1 \leq h \leq H\}$ denote the set of all (l, h) pairs. For each $c \in \mathcal{C}$ and generation scale k, let $\mathcal{D}_k(c) \subseteq \{s+1, \ldots, k-1\}$ denote the source scales excluded from the KV cache, where the first s source scales are protected from eviction. For source scales not included in $\mathcal{D}_k(c)$, the conditional and unconditional KV caches are maintained separately for $i < t$, while the conditional KV cache is shared with the unconditional branch for $i \geq t$. Accordingly, let $m_{c,i}^{(k)}$ denote the number of cached tokens from source scale $i < k$ across the two CFG branches:

$$m_{c,i}^{(k)} = \begin{cases} 0, & i \in \mathcal{D}_k(c), \\ 2n_i, & i \notin \mathcal{D}_k(c),\ i < t, \\ n_i, & i \notin \mathcal{D}_k(c),\ i \geq t. \end{cases} \tag{8}$$

For each candidate eviction set $\mathcal{D}_k(c)$, we evaluate the reconstruction error for each query using the eviction and sharing formulations in Eqs. 6 and 7. The errors are then averaged over the n_k queries:

$$e_k(c, \mathcal{D}_k(c)) = \frac{1}{n_k} \sum_{q=1}^{n_k} \mathcal{E}_{k,q}(c, \mathcal{D}_k(c)), \tag{9}$$

where $\mathcal{E}_{k,q}(c, \mathcal{D}_k(c))$ denotes the reconstruction error for query q under eviction and sharing. We denote by τ the first generation scale at which the KV cache exceeds the memory budget without eviction. Since eviction is irreversible across generation scales, the eviction sets are nested:

$$\mathcal{D}_\tau(c) \subseteq \mathcal{D}_{\tau+1}(c) \subseteq \cdots \subseteq \mathcal{D}_K(c), \qquad \forall c \in \mathcal{C}. \tag{10}$$

We jointly optimize the nested eviction sets for $k \geq \tau$ to minimize the accumulated reconstruction error while satisfying the cache budget at each generation scale:

$$\begin{aligned} &\min_{\{\mathcal{D}_k(c)\}} \sum_{k=\tau}^{K} \sum_{c \in \mathcal{C}} e_k(c, \mathcal{D}_k(c)) \\ &\text{s.t.} \quad \sum_{c \in \mathcal{C}} \sum_{i=1}^{k-1} m_{c,i}^{(k)} \leq b \cdot 2LHC_{K-1}, \qquad k = \tau, \ldots, K. \end{aligned} \tag{11}$$

The detailed reconstruction-error computation and optimization are provided in Appendix B.

5 Experiments

5.1 Experimental Setup

Models and Evaluation. We evaluate ResceneKV on the text-to-image models Infinity-2B and Infinity-8B (Han et al., 2025) at 20%, 10%, and 4% of the full KV cache budget (b in Eq. 3). We compare against the full-cache model and the KV cache eviction methods SnapKV (Li et al., 2024a), HACK (Qin et al., 2026), ScaleKV (Li et al., 2025), and HeatKV (Cederlund et al., 2026). For evaluation, we generate images from $5{,}000$ prompts in the MS-COCO 2017 validation set (Lin et al., 2014) and evaluate the generated outputs relative to the full-cache model using Fréchet Inception Distance (FID) (Heusel et al., 2017), Peak Signal-to-Noise Ratio (PSNR), Structural Similarity Index Measure (SSIM) (Wang et al., 2004), and Learned Perceptual Image Patch Similarity (LPIPS)

Table 1: Quantitative comparison on MS-COCO 2017 (Lin et al., 2014) under different KV cache budgets.

Method	Budget	Infinity-2B					Infinity-8B				
		KV Cache	FID↓	PSNR↑	SSIM↑	LPIPS↓	KV Cache	FID↓	PSNR↑	SSIM↑	LPIPS↓
Full Cache	100%	38550 MB	–	–	–	–	84328 MB	–	–	–	–
SnapKV	20%	7800 MB	3.02	23.03	0.800	0.113	17062 MB	2.92	22.71	0.761	0.104
HACK	20%	7800 MB	3.18	23.06	0.797	0.117	17062 MB	2.87	23.20	0.778	0.095
ScaleKV	20%	7800 MB	1.95	24.88	0.845	0.080	17062 MB	1.45	25.60	0.844	0.059
HeatKV	20%	7800 MB	1.01	28.43	0.908	0.044	17062 MB	1.05	27.43	0.884	0.043
ResceneKV	20%	7800 MB	**0.91**	**29.12**	**0.918**	**0.039**	17062 MB	**0.72**	**29.46**	**0.921**	**0.030**
SnapKV	10%	3900 MB	4.07	21.62	0.764	0.146	8531 MB	4.17	20.85	0.706	0.141
HACK	10%	3900 MB	3.84	21.42	0.757	0.152	8531 MB	4.24	21.11	0.710	0.138
ScaleKV	10%	3900 MB	2.59	22.86	0.800	0.113	8531 MB	2.13	23.25	0.786	0.085
HeatKV	10%	3900 MB	1.64	25.41	0.857	0.073	8531 MB	1.81	24.40	0.812	0.072
ResceneKV	10%	3900 MB	**1.43**	**26.37**	**0.876**	**0.062**	8531 MB	**1.31**	**25.95**	**0.859**	**0.054**
SnapKV	4%	1590 MB	4.75	19.88	0.720	0.194	3478 MB	6.19	19.09	0.649	0.193
HACK	4%	1590 MB	5.72	18.53	0.687	0.235	3478 MB	6.80	18.94	0.640	0.201
ScaleKV	4%	1590 MB	3.85	20.78	0.747	0.164	3478 MB	4.23	20.41	0.696	0.146
HeatKV	4%	1590 MB	2.69	22.14	0.786	0.127	3478 MB	2.73	21.78	0.737	0.111
ResceneKV	4%	1590 MB	**2.40**	**22.80**	**0.803**	**0.113**	3478 MB	**2.16**	**22.91**	**0.781**	**0.089**

(Zhang et al., 2018). We further evaluate generation quality on four benchmarks: GenEval (Ghosh et al., 2023) and DPG-Bench (Hu et al., 2024) for prompt following, and HPSv2.1 (Wu et al., 2023) and ImageReward (Xu et al., 2023) for human preference. For GenEval, we rewrite each prompt with a language model before generation, following the protocol used in the Infinity evaluation.

Implementation Details. We implement ResceneKV based on the HeatKV (Cederlund et al., 2026) codebase. We use $N = 10$ calibration prompts generated by Qwen3-VL-2B-Instruct (Bai et al., 2025) to estimate the reconstruction errors used for budget allocation. Following HeatKV, we set the number of sink scales to $s = 3$ and exclude the three coarsest source scales from eviction. With $K = 13$ generation scales, we set $t = 11$, such that sharing is applied to source scales $i \geq 11$. Following the configuration used in prior baselines (Li et al., 2025; Cederlund et al., 2026), we set the CFG scale to 3.0 for all experiments. All experiments use a batch size of 1, with Infinity-2B evaluated on a single RTX 3090 and Infinity-8B on a single NVIDIA H100.

5.2 Main Results

Comparison with KV Cache Eviction Baselines. We compare ResceneKV with existing KV cache eviction methods (Li et al., 2024a; Qin et al., 2026; Li et al., 2025; Cederlund et al., 2026) on MS-COCO 2017 (Lin et al., 2014) under KV cache budgets of 20%, 10%, and 4%. As shown in Fig. 1, ResceneKV maintains visual quality close to the full-cache output even with only a 4% KV cache budget. Table 1 further provides a quantitative comparison with the full-cache outputs using FID, PSNR, SSIM, and LPIPS. ResceneKV achieves the best results across all budgets on both Infinity-2B and Infinity-8B, consistently outperforming HeatKV, the strongest baseline. On Infinity-2B, ResceneKV reduces FID by 0.10, 0.21, and 0.29, while improving PSNR by 0.69, 0.96, and 0.66 dB at 20%, 10%, and 4% budgets, respectively. The improvement is even larger on Infinity-8B, where ResceneKV reduces FID by up to 0.57 and improves PSNR by up to 2.03 dB relative to HeatKV. Notably, ResceneKV at a 10% budget outperforms ScaleKV at 20% across all metrics for both models despite using only half the cache budget. These results demonstrate that ResceneKV better preserves the full-cache outputs under increasingly constrained KV cache budgets.

Evaluation of Generation Quality. Table 2 evaluates image generation quality at a 10% KV cache budget. GenEval (Ghosh et al., 2023) and DPG (Hu et al., 2024) measure prompt alignment, whereas HPSv2.1 (Wu et al., 2023) and ImageReward (Xu et al., 2023) assess human preference. ResceneKV remains competitive with the strongest baselines across the generation benchmarks while closely preserving the full-cache performance. Differences among methods are relatively small on the prompt-alignment benchmarks, where performance remains largely stable under KV cache eviction. Despite retaining only 10% of the KV cache, ResceneKV maintains generation quality close to that of the full-cache model. For example, on Infinity-8B, ResceneKV achieves a GenEval score

Table 2: Quantitative comparison on GenEval (Ghosh et al., 2023), HPSv2.1 (Wu et al., 2023), DPG (Hu et al., 2024) and ImageReward (Xu et al., 2023) under a 10% KV cache budget.

Method	GenEval				HPSv2.1			DPG	ImageReward
	Two Obj.	Position	Color Attri.	Overall	Photo	Concept-art	Overall	Overall	Overall
Infinity-2B	0.876	0.415	0.540	0.720	29.43	30.44	30.53	82.25	0.925
SnapKV	0.851	0.405	0.528	0.711	29.11	29.86	30.03	82.20	0.904
HACK	0.876	0.400	0.518	0.710	29.33	30.14	30.36	81.88	0.911
ScaleKV	0.876	0.410	0.540	0.712	29.36	30.24	30.36	82.24	0.919
HeatKV	0.881	**0.413**	**0.565**	0.720	**29.43**	**30.44**	**30.52**	82.19	**0.929**
ResceneKV	**0.886**	0.408	**0.565**	**0.722**	29.41	30.43	30.51	**82.30**	0.926
Infinity-8B	0.904	0.637	0.665	0.783	29.41	31.25	30.96	86.07	1.051
SnapKV	0.902	0.618	0.632	0.772	29.03	30.63	30.37	86.05	1.011
HACK	**0.909**	0.632	0.660	0.784	29.00	30.50	30.28	86.14	0.998
ScaleKV	**0.909**	0.635	0.660	0.784	29.34	31.04	30.78	**86.20**	1.029
HeatKV	0.904	**0.642**	0.685	**0.791**	29.28	31.07	30.80	86.07	1.038
ResceneKV	0.904	0.640	**0.688**	0.787	**29.38**	**31.20**	**30.90**	**86.20**	**1.047**

Table 3: Component analysis of ResceneKV on Infinity-2B using MS-COCO 2017.

Budget	Components		FID ↓	PSNR ↑	SSIM ↑	LPIPS ↓
	Recon. Error	Sharing				
20%	–	–	1.01	28.43	0.908	0.044
	✓	–	0.93	29.01	0.917	0.039
	–	✓	0.99	28.64	0.911	0.042
	✓	✓	**0.91**	**29.12**	**0.918**	**0.039**
10%	–	–	1.64	25.41	0.857	0.073
	✓	–	1.46	26.20	0.873	0.064
	–	✓	1.61	25.56	0.861	0.071
	✓	✓	**1.43**	**26.37**	**0.876**	**0.062**
4%	–	–	2.69	22.14	0.786	0.127
	✓	–	2.43	22.68	0.799	0.116
	–	✓	2.59	22.38	0.791	0.122
	✓	✓	**2.40**	**22.80**	**0.803**	**0.113**

Table 4: KV cache sharing scale ablation on Infinity-2B using MS-COCO 2017.

Budget	t	FID ↓	PSNR ↑	SSIM ↑	LPIPS ↓
20%	–	0.93	29.01	0.917	0.039
	10	0.94	28.98	0.917	0.039
	11	**0.91**	**29.12**	**0.918**	**0.039**
	12	0.92	29.03	0.917	0.039
10%	–	1.46	26.20	0.873	0.064
	10	1.46	26.35	**0.876**	0.062
	11	**1.43**	**26.37**	0.876	**0.062**
	12	1.46	26.25	0.874	0.063
4%	–	2.43	22.68	0.799	0.116
	10	**2.37**	**22.82**	**0.803**	**0.113**
	11	2.40	22.80	0.803	0.113
	12	2.41	22.72	0.801	0.115

of 0.787 and an ImageReward score of 1.047, closely matching the full-cache scores of 0.783 and 1.051, respectively. These results show that ResceneKV preserves strong prompt alignment and human preference while substantially reducing KV cache usage.

5.3 Analytical Experiments

Ablation Studies. Table 3 analyzes the contribution of reconstruction-error guided KV cache eviction and KV cache sharing. Using reconstruction error alone substantially improves performance over the baseline across all budgets, showing the benefit of directly accounting for eviction-induced output distortion. KV cache sharing alone also consistently improves all evaluation metrics, and combining it with reconstruction-error guided KV cache eviction further improves performance, yielding the best overall results. Table 4 examines the effect of the KV cache sharing scale t, where "–" indicates that sharing is disabled. The results remain relatively stable across $t \in \{10, 11, 12\}$, with $t = 11$ providing the strongest overall performance at the 20% and 10% budgets, while $t = 10$ performs slightly better under the more constrained 4% budget. Based on these results, we use $t = 11$ as the default setting.

Layer-wise Output Error Analysis. We compare the normalized output error across transformer layers at generation scale $k = 13$. As shown in Fig. 4(a), ResceneKV consistently yields lower errors than HeatKV across layers. ResceneKV without KV cache sharing already reduces the error relative to HeatKV, showing the benefit of eviction guided by reconstruction error. The full ResceneKV further decreases the error across most layers under the same cache budget. This trend demonstrates the effectiveness of the proposed components in reducing layer-wise output error.

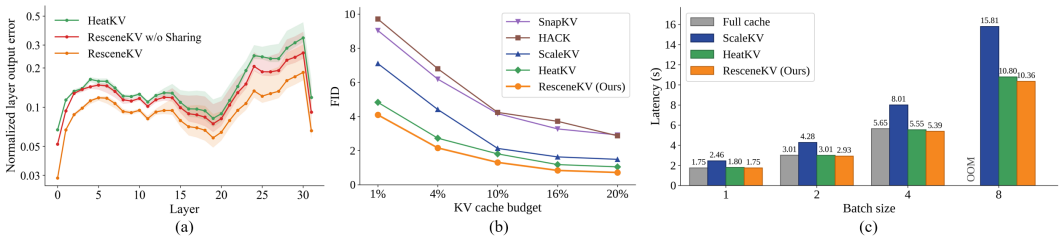

Figure 4: (a) Layer-wise normalized output error on Infinity-2B. (b) FID under different KV cache budgets on Infinity-8B. (c) Latency under different batch sizes on Infinity-8B.

Trade-off between Quality and Efficiency. Fig. 4(b) compares generation quality across different KV cache budgets on Infinity-8B. FID increases for all methods as the cache budget decreases, while ResceneKV consistently achieves the lowest FID across the evaluated budgets. At a 4% budget, ResceneKV achieves an FID of 2.16, compared with 2.73 for HeatKV. The performance gap over prior methods becomes more pronounced as the cache budget decreases, indicating that ResceneKV remains effective under more constrained memory settings. These results show that ResceneKV achieves a better balance between generation quality and cache memory.

Efficiency Analysis. We evaluate inference latency across different batch sizes on Infinity-8B using a single NVIDIA H100, comparing ResceneKV with the full-cache model and existing KV cache eviction methods. As shown in Fig. 4(c), ResceneKV consistently achieves the lowest latency among the eviction methods across all batch sizes. At a batch size of 8, the full-cache model runs out of memory, while ResceneKV remains feasible with a latency of 10.36 s, compared with 15.81 s for ScaleKV and 10.80 s for HeatKV. These results show that ResceneKV substantially reduces KV cache memory without introducing additional inference overhead, while allowing larger batch sizes.

Sensitivity to the Number of Calibration Prompts. To assess sensitivity to the calibration set size, we evaluate ResceneKV with different numbers of calibration prompts on Infinity-2B under a 10% KV cache budget. As shown in Table 5, performance remains stable as the number of calibration prompts varies from 1 to 20. FID ranges from 1.44 to 1.46, while PSNR remains between 26.27 and 26.34 dB. Increasing the number of prompts beyond 10 provides only marginal changes in both metrics, indicating that the reconstruction-error estimates are not highly sensitive to the calibration set size. These results show that a small calibration set is sufficient to achieve stable performance without extensive calibration. Based on this observation, we use $N = 10$ calibration prompts in all experiments.

Table 5: Sensitivity to the number of calibration prompts.

# Prompts	FID ↓	PSNR ↑
1	1.46	26.27
5	1.45	26.30
10	**1.44**	**26.34**
15	**1.44**	26.33
20	1.45	26.33

6 Conclusion

In this work, we introduced ResceneKV to address the growing KV cache memory overhead in visual autoregressive generation. Motivated by the mismatch between attention statistics and the actual reconstruction error caused by eviction, ResceneKV directly evaluates eviction-induced reconstruction error and uses it to guide KV cache allocation across layers, attention heads, source scales, and generation scales. We additionally reduce the redundancy introduced by CFG through KV cache sharing at later scales. ResceneKV closely matches the full-cache results even under highly constrained cache budgets, while maintaining generation quality and enabling more memory-efficient inference. Experimental results demonstrate the effectiveness of ResceneKV in preserving generation quality while substantially reducing KV cache memory.

Limitations. ResceneKV determines its cache allocation from a fixed calibration set and applies the resulting eviction configuration to all inputs. Although this provides stable performance with a small calibration set, it does not adapt to input-specific variations in the importance of source scales during generation. Developing an input-adaptive allocation strategy is therefore an interesting direction for future work.

AI USE STATEMENT

In this work, we used AI tools to assist with language editing, sentence-level rewriting, improving the clarity of the manuscript, and identifying relevant literature. AI tools were also used to support parts of the early-stage research ideation process and to assist with code design. All technical content, including the proposed method, mathematical formulations, experimental design, results, and AI-assisted code, was carefully reviewed and verified. We take full responsibility for the final content of this work.

REFERENCES

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

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024. v4, May 2025.

Jonathan Cederlund, Axel Berg, William Isaksson, Durmus Alp Emre Acar, Chuteng Zhou, and Pontus Giselsson. HeatKV: Head-tuned KV-cache compression for visual autoregressive modeling. *arXiv preprint arXiv:2605.14877*, 2026.

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.

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.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. GenEval: An object-focused framework for evaluating text-to-image alignment. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. arXiv:2310.11513.

Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. *arXiv preprint arXiv:2412.04431*, 2025.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 6626–6637, 2017.

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

Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu. ELLA: Equip diffusion models with LLM for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 2024. introduces DPG-Bench.

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

Kunjun Li, Zigeng Chen, Cheng-Yen Yang, and Jenq-Neng Hwang. Memory-efficient visual autoregressive modeling with scale-aware KV cache compression. *Advances in Neural Information Processing Systems*, 2025. arXiv:2505.19602.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a. arXiv:2404.14469.

Zongming Li, Tianheng Cheng, Shoufa Chen, Peize Sun, Haocheng Shen, Longjin Ran, Xiaoxin Chen, Wenyu Liu, and Xinggang Wang. Controlar: Controllable image generation with autoregressive models. *arXiv preprint arXiv:2410.02705*, 2024b.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision (ECCV)*, pp. 740–755, 2014.

Alex Nichol, Prafulla Dhariwal, Aditya Ramesh, Pranav Shyam, Pamela Mishkin, Bob McGrew, Ilya Sutskever, and Mark Chen. Glide: Towards photorealistic image generation and editing with text-guided diffusion models. *arXiv preprint arXiv:2112.10741*, 2021.

Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li. Cake: Cascading and adaptive kv cache eviction with layer preferences. In *International Conference on Learning Representations*, volume 2025, pp. 89945–89977, 2025.

Ziran Qin, Youru Lv, Mingbao Lin, Hang Guo, Zeren Zhang, Danping Zou, and Weiyao Lin. Head-aware KV cache compression for efficient visual autoregressive modeling. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2026. arXiv:2504.09261.

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.

Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. Hart: Efficient visual generation with hybrid autoregressive transformer. *arXiv preprint*, 2024.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

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, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

Zhongwei Wan, Ziang Wu, Che Liu, Jinfa Huang, Zhihong Zhu, Peng Jin, Longyue Wang, and Li Yuan. Look-m: Look-once optimization in kv cache for efficient multimodal long-context inference. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 4065–4078, 2024.

Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4): 600–612, 2004.

Xiaoshi Wu, Yiming Hao, Keqiang Sun, Yixiong Chen, Feng Zhu, Rui Zhao, and Hongsheng Li. Human preference score v2: A solid benchmark for evaluating human preferences of text-to-image synthesis. *arXiv preprint arXiv:2306.09341*, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Boxun Xu, Yu Wang, Zihu Wang, and Peng Li. Ams-kv: Adaptive kv caching in multi-scale visual autoregressive transformers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 27206–27214, 2026.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. ImageReward: Learning and evaluating human preferences for text-to-image generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, 2023. arXiv:2304.05977.

Ziyu Yao, Jialin Li, Yifeng Zhou, Yong Liu, Xi Jiang, Chengjie Wang, Feng Zheng, Yuexian Zou, and Lei Li. Car: Controllable autoregressive modeling for visual generation. *arXiv preprint arXiv:2410.04671*, 2024.

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*, 2(3):5, 2022.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in neural information processing systems*, 36:34661–34710, 2023.

A PROOFS

A.1 PROOF OF PROPOSITION 4.1

Proof. For a single query $\mathbf{q}$ at generation scale k, let $\mathbf{k}_{i,j} \in \mathbb{R}^{d_h}$ denote the j-th key vector of scale i, and define the corresponding attention logit as $z_{i,j} = \mathbf{q}\mathbf{k}_{i,j}^\top/\sqrt{d_h}$. Let $\mathbf{a}_i \in \mathbb{R}^{1\times n_i}$ denote the attention weights over scale i, for $i = 1, \ldots, k$. The pre-eviction softmax normalization term Z and attention weight $[\mathbf{a}_i]_j$ are

$$Z = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\exp(z_{i,j}), \qquad [\mathbf{a}_i]_j = \frac{\exp(z_{i,j})}{Z}. \tag{12}$$

Let $\mathbf{v}_{i,j} \in \mathbb{R}^{d_h}$ denote the j-th value vector of scale i. The attention output before eviction is

$$\mathbf{o} = \sum_{i=1}^{k}\sum_{j=1}^{n_i}[\mathbf{a}_i]_j\mathbf{v}_{i,j} = \sum_{i=1}^{k}\mathbf{a}_i\mathbf{V}_i. \tag{13}$$

For an eviction set $\mathcal{D} \subseteq \{1, \ldots, k-1\}$, the total attention mass assigned to the removed source scales is

$$\beta_{\mathcal{D}} = \sum_{i\in\mathcal{D}}\beta_i = \sum_{i\in\mathcal{D}}\sum_{j=1}^{n_i}[\mathbf{a}_i]_j = \frac{\sum_{i\in\mathcal{D}}\sum_{j=1}^{n_i}\exp(z_{i,j})}{Z}. \tag{14}$$

The attention-weighted average value of the evicted scales is

$$\bar{\mathbf{v}}_{\mathcal{D}} = \frac{\sum_{i\in\mathcal{D}}\mathbf{a}_i\mathbf{V}_i}{\beta_{\mathcal{D}}}, \qquad \sum_{i\in\mathcal{D}}\mathbf{a}_i\mathbf{V}_i = \beta_{\mathcal{D}}\bar{\mathbf{v}}_{\mathcal{D}}. \tag{15}$$

After removing the source scales in $\mathcal{D}$, the softmax normalization term becomes

$$\widetilde{Z} = Z - \sum_{i\in\mathcal{D}}\sum_{j=1}^{n_i}\exp(z_{i,j}) = Z(1-\beta_{\mathcal{D}}). \tag{16}$$

Since the logits of the remaining tokens are unchanged, their attention weights after eviction are

$$[\widetilde{\mathbf{a}}_i]_j = \frac{\exp(z_{i,j})}{\widetilde{Z}} = \frac{[\mathbf{a}_i]_j}{1-\beta_{\mathcal{D}}}, \qquad i \notin \mathcal{D}. \tag{17}$$

Using the renormalized attention weights and Eq. 13, the attention output after eviction is

$$\widetilde{\mathbf{o}} = \sum_{i\notin\mathcal{D}}\sum_{j=1}^{n_i}[\widetilde{\mathbf{a}}_i]_j\mathbf{v}_{i,j} = \frac{\sum_{i\notin\mathcal{D}}\mathbf{a}_i\mathbf{V}_i}{1-\beta_{\mathcal{D}}} = \frac{\mathbf{o} - \sum_{i\in\mathcal{D}}\mathbf{a}_i\mathbf{V}_i}{1-\beta_{\mathcal{D}}} = \frac{\mathbf{o} - \beta_{\mathcal{D}}\bar{\mathbf{v}}_{\mathcal{D}}}{1-\beta_{\mathcal{D}}}. \tag{18}$$

Subtracting the post-eviction output from the original output gives

$$\mathbf{o} - \widetilde{\mathbf{o}} = \mathbf{o} - \frac{\mathbf{o} - \beta_{\mathcal{D}}\bar{\mathbf{v}}_{\mathcal{D}}}{1-\beta_{\mathcal{D}}} = \frac{\beta_{\mathcal{D}}}{1-\beta_{\mathcal{D}}}\left(\bar{\mathbf{v}}_{\mathcal{D}} - \mathbf{o}\right). \tag{19}$$

The reconstruction error induced by eviction $\mathcal{D}$ is therefore

$$\mathcal{E}_{\mathcal{D}}^{\text{evict}} = \|\mathbf{o} - \widetilde{\mathbf{o}}\|_2^2 = \left(\frac{\beta_{\mathcal{D}}}{1-\beta_{\mathcal{D}}}\right)^2\|\bar{\mathbf{v}}_{\mathcal{D}} - \mathbf{o}\|_2^2. \tag{20}$$

□

A.2 Proof of Proposition 4.2

Proof. At generation scale k, we consider a single unconditional query $\mathbf{q}^u$. Let $\mathcal{S} = \{i \mid t \le i < K\}$ denote the source scale indices subject to KV cache sharing over the generation process. Only indices in $\mathcal{S}$ with $i < k$ are available when generating scale k. For each preceding source scale i, let $\mathbf{k}^u_{i,j}$ denote the j-th key vector before sharing, and for $i \in \mathcal{S}$, let $\widetilde{\mathbf{k}}^u_{i,j}$ denote the corresponding key vector after sharing. The corresponding attention logits are $z^u_{i,j} = \mathbf{q}^u(\mathbf{k}^u_{i,j})^\top/\sqrt{d_h}$ and $\widetilde{z}^u_{i,j} = \mathbf{q}^u(\widetilde{\mathbf{k}}^u_{i,j})^\top/\sqrt{d_h}$. Using the pre-sharing normalization Z^u, we define

$$Z^u = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\exp(z^u_{i,j}), \qquad [\mathbf{a}^u_i]_j = \frac{\exp(z^u_{i,j})}{Z^u}, \qquad [\widetilde{\mathbf{a}}^u_i]_j = \frac{\exp(\widetilde{z}^u_{i,j})}{Z^u}. \tag{21}$$

Here, $\mathbf{a}^u_i$ denotes the pre-sharing attention weights, whereas $\widetilde{\mathbf{a}}^u_i$ denotes the weights computed from the shared keys under the same pre-sharing normalization Z^u. The total attention mass over the source scales in $\mathcal{S}$ before sharing is $\beta^u_{\mathcal{S}}$, while the corresponding mass computed from the shared keys under the pre-sharing normalization is $\widetilde{\beta}^u_{\mathcal{S}}$, given by

$$\beta^u_{\mathcal{S}} = \sum_{i\in\mathcal{S}}\beta^u_i = \frac{\sum_{i\in\mathcal{S}}\sum_{j=1}^{n_i}\exp(z^u_{i,j})}{Z^u}, \qquad \widetilde{\beta}^u_{\mathcal{S}} = \sum_{i\in\mathcal{S}}\widetilde{\beta}^u_i = \frac{\sum_{i\in\mathcal{S}}\sum_{j=1}^{n_i}\exp(\widetilde{z}^u_{i,j})}{Z^u}. \tag{22}$$

Let $\mathbf{v}^u_{i,j} \in \mathbb{R}^{d_h}$ denote the j-th value vector of source scale i before sharing. We define the attention-weighted average values over $\mathcal{S}$ as

$$\bar{\mathbf{v}}^u_{\mathcal{S}} = \frac{\sum_{i\in\mathcal{S}}\mathbf{a}^u_i\mathbf{V}^u_i}{\beta^u_{\mathcal{S}}}, \qquad \widetilde{\bar{\mathbf{v}}}^u_{\mathcal{S}} = \frac{\sum_{i\in\mathcal{S}}\widetilde{\mathbf{a}}^u_i\widetilde{\mathbf{V}}^u_i}{\widetilde{\beta}^u_{\mathcal{S}}}. \tag{23}$$

Sharing replaces the original logits of the selected source scales with those computed from the shared keys. Thus, the post-sharing normalization becomes

$$\widetilde{Z}^u = Z^u - \sum_{i\in\mathcal{S}}\sum_{j=1}^{n_i}\exp(z^u_{i,j}) + \sum_{i\in\mathcal{S}}\sum_{j=1}^{n_i}\exp(\widetilde{z}^u_{i,j}) = Z^u\left(1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}\right). \tag{24}$$

The unconditional attention output before sharing is

$$\mathbf{o}^u = \sum_{i=1}^{k}\sum_{j=1}^{n_i}[\mathbf{a}^u_i]_j\mathbf{v}^u_{i,j} = \sum_{i=1}^{k}\mathbf{a}^u_i\mathbf{V}^u_i. \tag{25}$$

Using the post-sharing normalization in Eq. 24, the unconditional attention output after sharing is

$$\widetilde{\mathbf{o}}^u = \frac{\sum_{i\notin\mathcal{S}}\mathbf{a}^u_i\mathbf{V}^u_i + \sum_{i\in\mathcal{S}}\widetilde{\mathbf{a}}^u_i\widetilde{\mathbf{V}}^u_i}{1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}} = \frac{\mathbf{o}^u - \sum_{i\in\mathcal{S}}\mathbf{a}^u_i\mathbf{V}^u_i + \sum_{i\in\mathcal{S}}\widetilde{\mathbf{a}}^u_i\widetilde{\mathbf{V}}^u_i}{1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}} = \frac{\mathbf{o}^u - \beta^u_{\mathcal{S}}\bar{\mathbf{v}}^u_{\mathcal{S}} + \widetilde{\beta}^u_{\mathcal{S}}\widetilde{\bar{\mathbf{v}}}^u_{\mathcal{S}}}{1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}}. \tag{26}$$

Subtracting the post-sharing output from the original output gives

$$\mathbf{o}^u - \widetilde{\mathbf{o}}^u = \frac{\left(1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}\right)\mathbf{o}^u - \left(\mathbf{o}^u - \beta^u_{\mathcal{S}}\bar{\mathbf{v}}^u_{\mathcal{S}} + \widetilde{\beta}^u_{\mathcal{S}}\widetilde{\bar{\mathbf{v}}}^u_{\mathcal{S}}\right)}{1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}} = \frac{\beta^u_{\mathcal{S}}\left(\bar{\mathbf{v}}^u_{\mathcal{S}} - \mathbf{o}^u\right) - \widetilde{\beta}^u_{\mathcal{S}}\left(\widetilde{\bar{\mathbf{v}}}^u_{\mathcal{S}} - \mathbf{o}^u\right)}{1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}}. \tag{27}$$

The reconstruction error induced by sharing over $\mathcal{S}$ is therefore

$$\mathcal{E}^{\text{share}}_{\mathcal{S}} = \|\mathbf{o}^u - \widetilde{\mathbf{o}}^u\|_2^2 = \frac{\left\|\beta^u_{\mathcal{S}}\left(\bar{\mathbf{v}}^u_{\mathcal{S}} - \mathbf{o}^u\right) - \widetilde{\beta}^u_{\mathcal{S}}\left(\widetilde{\bar{\mathbf{v}}}^u_{\mathcal{S}} - \mathbf{o}^u\right)\right\|_2^2}{\left(1-\beta^u_{\mathcal{S}}+\widetilde{\beta}^u_{\mathcal{S}}\right)^2}. \tag{28}$$

□

B Additional Details on Joint Budget Allocation

The joint budget allocation in Sec. 4.4 couples the eviction sets across generation scales because an earlier eviction persists at all subsequent scales while the KV cache budget must be satisfied at each scale. Here, we provide the detailed computation of the reconstruction error and the optimization procedure used to solve this problem. We first use a Gram matrix formulation to account for interactions among the reconstruction-error contributions of multiple source scales. We then introduce Lagrange multipliers to decouple the global budget constraints and solve the nested eviction decisions for each layer–head pair by dynamic programming.

Reconstruction Error under Eviction and Sharing. Let $\mathcal{C} = \{c = (l, h) \mid 1 \le l \le L,\ 1 \le h \le H\}$ denote the set of layer–head pairs. For $c \in \mathcal{C}$, we consider the reconstruction error for query q at generation scale k. To evaluate different combinations of eviction and sharing, we decompose the error terms in Eqs. 6 and 7 into source-scale contributions. The error contribution associated with evicting source scale i is

$$\boldsymbol{\delta}_i^{\text{evict}}(q) = \mathbf{a}_i(q)\mathbf{V}_i - \beta_i(q)\mathbf{o}(q). \tag{29}$$

For a shared source scale, the corresponding contribution on the unconditional branch is

$$\boldsymbol{\delta}_i^{\text{share}}(q) = \mathbf{a}_i^u(q)\mathbf{V}_i^u - \widetilde{\mathbf{a}}_i^u(q)\widetilde{\mathbf{V}}_i^u - \left(\beta_i^u(q) - \widetilde{\beta}_i^u(q)\right)\mathbf{o}^u(q). \tag{30}$$

For a given eviction set $\mathcal{D}_k(c)$, the error contribution and the corresponding change in normalization for source scale i are

$$\left(\boldsymbol{\delta}_i(q, \mathcal{D}_k(c)), \Delta\beta_i(q, \mathcal{D}_k(c))\right) = \begin{cases} \left(\boldsymbol{\delta}_i^{\text{evict}}(q), \beta_i(q)\right), & i \in \mathcal{D}_k(c), \\ \left(\boldsymbol{\delta}_i^{\text{share}}(q), \beta_i^u(q) - \widetilde{\beta}_i^u(q)\right), & i \notin \mathcal{D}_k(c),\ i \ge t, \\ (\mathbf{0}, 0), & i \notin \mathcal{D}_k(c),\ i < t. \end{cases} \tag{31}$$

The reconstruction error $\mathcal{E}_{k,q}$ used in Eq. 9 can be evaluated using the Gram matrix $G_{i,j}(q, \mathcal{D}_k(c)) = \langle \boldsymbol{\delta}_i(q, \mathcal{D}_k(c)), \boldsymbol{\delta}_j(q, \mathcal{D}_k(c)) \rangle$ as

$$\mathcal{E}_{k,q}(c, \mathcal{D}_k(c)) = \frac{\left\| \sum_{i=1}^{k-1} \boldsymbol{\delta}_i(q, \mathcal{D}_k(c)) \right\|_2^2}{\left(1 - \sum_{i=1}^{k-1} \Delta\beta_i(q, \mathcal{D}_k(c))\right)^2} = \frac{\sum_{i=1}^{k-1}\sum_{j=1}^{k-1} G_{i,j}(q, \mathcal{D}_k(c))}{\left(1 - \sum_{i=1}^{k-1} \Delta\beta_i(q, \mathcal{D}_k(c))\right)^2}. \tag{32}$$

Lagrangian Decomposition. The KV cache constraint at each generation scale is shared across all $c \in \mathcal{C}$, so the eviction sets cannot be optimized separately for each c. We introduce a non-negative Lagrange multiplier λ_k for the budget constraint at generation scale k. The Lagrangian of Eq. 11 is

$$\mathcal{L} = \sum_{k=\tau}^{K}\sum_{c\in\mathcal{C}} e_k(c, \mathcal{D}_k(c)) + \sum_{k=\tau}^{K} \lambda_k \left(\sum_{c\in\mathcal{C}}\sum_{i=1}^{k-1} m_{c,i}^{(k)} - b \cdot 2LHC_{K-1} \right), \quad \lambda_k \ge 0. \tag{33}$$

For a given multiplier vector $\boldsymbol{\lambda} = (\lambda_\tau, \ldots, \lambda_K)$, Eq. 33 can be rewritten as

$$\mathcal{L} = \sum_{c\in\mathcal{C}}\sum_{k=\tau}^{K} \left[e_k(c, \mathcal{D}_k(c)) + \lambda_k \sum_{i=1}^{k-1} m_{c,i}^{(k)} \right] - b \cdot 2LHC_{K-1} \sum_{k=\tau}^{K} \lambda_k. \tag{34}$$

For fixed $\boldsymbol{\lambda}$, the Lagrangian can be written as a sum of terms, each depending only on the eviction sets of a single $c \in \mathcal{C}$. The last term is independent of the eviction sets. Therefore, the optimization can be solved independently for each c. The remaining problem is

$$\min_{\mathcal{D}_\tau(c) \subseteq \cdots \subseteq \mathcal{D}_K(c)} \sum_{k=\tau}^{K} \left[e_k(c, \mathcal{D}_k(c)) + \lambda_k \sum_{i=1}^{k-1} m_{c,i}^{(k)} \right]. \tag{35}$$

Dynamic Programming over Nested Eviction Sets. For a given multiplier vector $\boldsymbol{\lambda}$, the optimization in Eq. 35 can be solved independently for each $c \in \mathcal{C}$. At generation scale k, each DP state

is represented by an eviction set $\mathcal{D} \subseteq \{s+1, \ldots, k-1\}$, where the first s source scales remain protected from eviction. We write $m_{c,i}^{(k)}(\mathcal{D})$ for the cache allocation in Eq. 8 under the state $\mathcal{D}$.

Let $F_k^c(\mathcal{D})$ denote the minimum accumulated objective from generation scale k to K under fixed $\boldsymbol{\lambda}$, given the eviction set $\mathcal{D}$ at scale k. At the final generation scale,

$$F_K^c(\mathcal{D}) = e_K(c, \mathcal{D}) + \lambda_K \sum_{i=1}^{K-1} m_{c,i}^{(K)}(\mathcal{D}). \tag{36}$$

For $k < K$, the recurrence is

$$F_k^c(\mathcal{D}) = e_k(c, \mathcal{D}) + \lambda_k \sum_{i=1}^{k-1} m_{c,i}^{(k)}(\mathcal{D}) + \min_{\mathcal{D}' \supseteq \mathcal{D}} F_{k+1}^c(\mathcal{D}'), \tag{37}$$

where $\mathcal{D}' \subseteq \{s+1, \ldots, k\}$. The minimization is restricted to $\mathcal{D}' \supseteq \mathcal{D}$, since any source scale evicted at generation scale k must remain evicted at subsequent scales. For all $k < \tau$, the KV cache satisfies the budget without eviction. We therefore optimize the nested eviction sequence over $k = \tau, \ldots, K$. The initial eviction set is

$$\mathcal{D}_\tau^*(c) = \arg\min_{\mathcal{D}} F_\tau^c(\mathcal{D}). \tag{38}$$

For each subsequent generation scale, the optimal eviction set is selected as

$$\mathcal{D}_{k+1}^*(c) \in \arg\min_{\mathcal{D}' \supseteq \mathcal{D}_k^*(c)} F_{k+1}^c(\mathcal{D}'), \qquad k = \tau, \ldots, K-1. \tag{39}$$

This formulation determines which source scales are evicted and when each eviction occurs over the generation process.

C Additional Quantitative Comparison

Table 6: Quantitative comparison on GenEval (Ghosh et al., 2023), HPSv2.1 (Wu et al., 2023), DPG (Hu et al., 2024) and ImageReward (Xu et al., 2023) under different KV cache budgets.

Method	Budget	GenEval				HPSv2.1			DPG	ImageReward
		Two Obj.	Position	Color Attri.	Overall	Photo	Concept-art	Overall	Overall	Overall
Infinity-2B	100%	0.876	0.415	0.540	0.720	29.43	30.44	30.53	82.25	0.925
SnapKV	20%	0.874	0.405	0.540	0.713	29.28	30.13	30.29	82.12	0.914
HACK	20%	0.871	**0.415**	0.522	0.709	29.29	30.17	30.32	82.21	0.910
ScaleKV	20%	0.874	0.412	0.540	0.713	29.41	30.35	30.45	**82.35**	0.926
HeatKV	20%	0.874	0.405	0.542	**0.718**	**29.44**	**30.43**	**30.53**	82.21	**0.931**
ResceneKV	20%	**0.876**	0.407	**0.552**	0.717	29.40	30.42	30.51	82.27	0.929
SnapKV	10%	0.851	0.405	0.528	0.711	29.11	29.86	30.03	82.20	0.904
HACK	10%	0.876	0.400	0.518	0.710	29.33	30.14	30.36	81.88	0.911
ScaleKV	10%	0.876	0.410	0.540	0.712	29.36	30.24	30.36	82.24	0.919
HeatKV	10%	0.881	**0.413**	**0.565**	0.720	**29.43**	**30.44**	**30.52**	82.19	**0.929**
ResceneKV	10%	**0.886**	0.408	**0.565**	**0.722**	29.41	30.43	30.51	**82.30**	0.926
SnapKV	4%	0.851	0.400	0.532	0.706	29.10	29.88	30.03	81.84	0.892
HACK	4%	0.854	0.405	0.507	0.706	29.11	29.79	29.96	81.49	0.881
ScaleKV	4%	0.859	**0.420**	**0.552**	0.713	29.27	30.15	30.27	82.16	0.911
HeatKV	4%	**0.879**	0.415	0.535	0.714	**29.41**	**30.37**	**30.47**	**82.41**	0.921
ResceneKV	4%	**0.879**	0.407	0.547	**0.717**	29.39	**30.37**	**30.47**	82.24	**0.923**
Infinity-8B	100%	0.904	0.637	0.665	0.783	29.41	31.25	30.96	86.07	1.051
SnapKV	20%	0.902	0.620	**0.668**	**0.786**	29.28	30.13	30.29	86.07	1.032
HACK	20%	0.891	0.618	0.635	0.774	29.29	30.17	30.32	**86.19**	1.010
ScaleKV	20%	0.904	0.610	0.635	0.773	29.41	30.35	30.45	86.13	1.035
HeatKV	20%	0.891	0.620	0.650	0.778	**29.44**	**30.43**	**30.53**	86.15	1.041
ResceneKV	20%	**0.912**	**0.623**	0.660	0.782	29.40	30.42	30.51	86.08	**1.047**
SnapKV	10%	0.902	0.618	0.632	0.772	29.03	30.63	30.37	86.05	1.011
HACK	10%	**0.909**	0.632	0.660	0.784	29.00	30.50	30.28	86.14	0.998
ScaleKV	10%	**0.909**	0.635	0.660	0.784	29.34	31.04	30.78	**86.20**	1.029
HeatKV	10%	0.904	**0.642**	0.685	**0.791**	29.28	31.07	30.80	86.07	1.038
ResceneKV	10%	0.904	0.640	**0.688**	0.787	**29.38**	**31.20**	**30.90**	**86.20**	**1.047**
SnapKV	4%	0.891	0.598	0.590	0.762	29.10	29.88	30.03	86.13	0.989
HACK	4%	0.884	0.610	0.613	0.771	29.11	29.79	29.96	**86.27**	0.984
ScaleKV	4%	0.889	0.608	0.635	0.778	29.27	30.15	30.27	86.10	1.011
HeatKV	4%	0.899	**0.618**	**0.660**	**0.785**	**29.41**	**30.37**	**30.47**	86.00	1.025
ResceneKV	4%	**0.907**	0.613	0.658	0.782	29.39	**30.37**	**30.47**	86.03	**1.030**

Table 6 evaluates generation quality across different KV cache budgets using GenEval (Ghosh et al., 2023), HPSv2.1 (Wu et al., 2023), DPG (Hu et al., 2024), and ImageReward (Xu et al., 2023). Across both Infinity-2B and Infinity-8B, ResceneKV maintains generation quality close to that of the full-cache model while remaining competitive with prior KV cache eviction methods. These results demonstrate that ResceneKV effectively preserves generation quality under highly constrained KV cache budgets.

Table 7: Quantitative comparison on MS-COCO 2017 (Lin et al., 2014) and GenEval (Ghosh et al., 2023) for HART (Tang et al., 2024) under different KV cache budgets. HART includes 300 text-conditioning tokens in the KV cache, which are kept intact for all methods. The budget applies only to image tokens.

Method	Budget	KV Cache	MS-COCO 2017				GenEval			
			FID↓	PSNR↑	SSIM↑	LPIPS↓	Two Obj.	Position	Color Attri.	Overall↑
Full Cache	100%	12722 MB	–	–	–	–	0.616	0.093	0.205	0.496
SnapKV	20%	3085 MB	17.42	18.19	0.590	0.300	0.523	**0.108**	0.188	0.458
ScaleKV	20%	3085 MB	28.16	17.14	0.537	0.364	0.447	0.085	0.170	0.420
HACK	20%	3085 MB	7.14	18.42	0.633	0.251	0.561	0.083	0.213	0.476
HeatKV	20%	3085 MB	3.80	22.96	0.804	0.117	**0.619**	0.095	0.205	**0.496**
ResceneKV	20%	3085 MB	**3.02**	**24.44**	**0.846**	**0.090**	0.609	0.090	**0.225**	0.491
SnapKV	10%	1879 MB	31.30	15.73	0.486	0.438	0.417	0.093	0.158	0.417
ScaleKV	10%	1879 MB	44.25	15.68	0.481	0.444	0.359	0.068	0.118	0.375
HACK	10%	1879 MB	10.59	16.79	0.563	0.333	0.543	0.075	0.180	0.459
HeatKV	10%	1879 MB	**5.36**	20.14	0.704	0.187	**0.616**	0.090	**0.220**	0.495
ResceneKV	10%	1879 MB	5.53	**20.20**	**0.705**	**0.186**	**0.616**	**0.098**	0.210	**0.496**
SnapKV	4%	1157 MB	30.88	14.84	0.470	0.463	0.427	0.073	0.135	0.392
ScaleKV	4%	1157 MB	33.79	14.50	0.449	0.494	0.394	0.078	0.120	0.395
HACK	4%	1157 MB	19.86	13.83	0.446	0.507	0.432	0.053	0.155	0.399
HeatKV	4%	1157 MB	11.66	17.48	0.585	0.303	**0.606**	0.093	**0.228**	0.490
ResceneKV	4%	1157 MB	**7.99**	**18.29**	**0.627**	**0.256**	0.593	**0.110**	**0.228**	**0.497**

We further evaluate ResceneKV on HART (Tang et al., 2024) to examine its applicability to a different visual autoregressive architecture. As shown in Table 7, ResceneKV consistently maintains high fidelity to the full-cache outputs on MS-COCO 2017 across different KV cache budgets. At the 20% and 4% budgets, ResceneKV achieves the best FID, PSNR, SSIM, and LPIPS among all eviction methods, while remaining competitive at 10%. The improvement is particularly pronounced at the 4% budget, where ResceneKV reduces FID from 11.66 to 7.99 and improves PSNR from 17.48 to 18.29 dB compared with HeatKV. On GenEval, ResceneKV also remains competitive across the individual metrics, with overall scores close to those of the full-cache model even under highly constrained budgets. These results demonstrate the applicability of ResceneKV beyond the Infinity model family.

D COMPARISON ON QUALITATIVE RESULTS

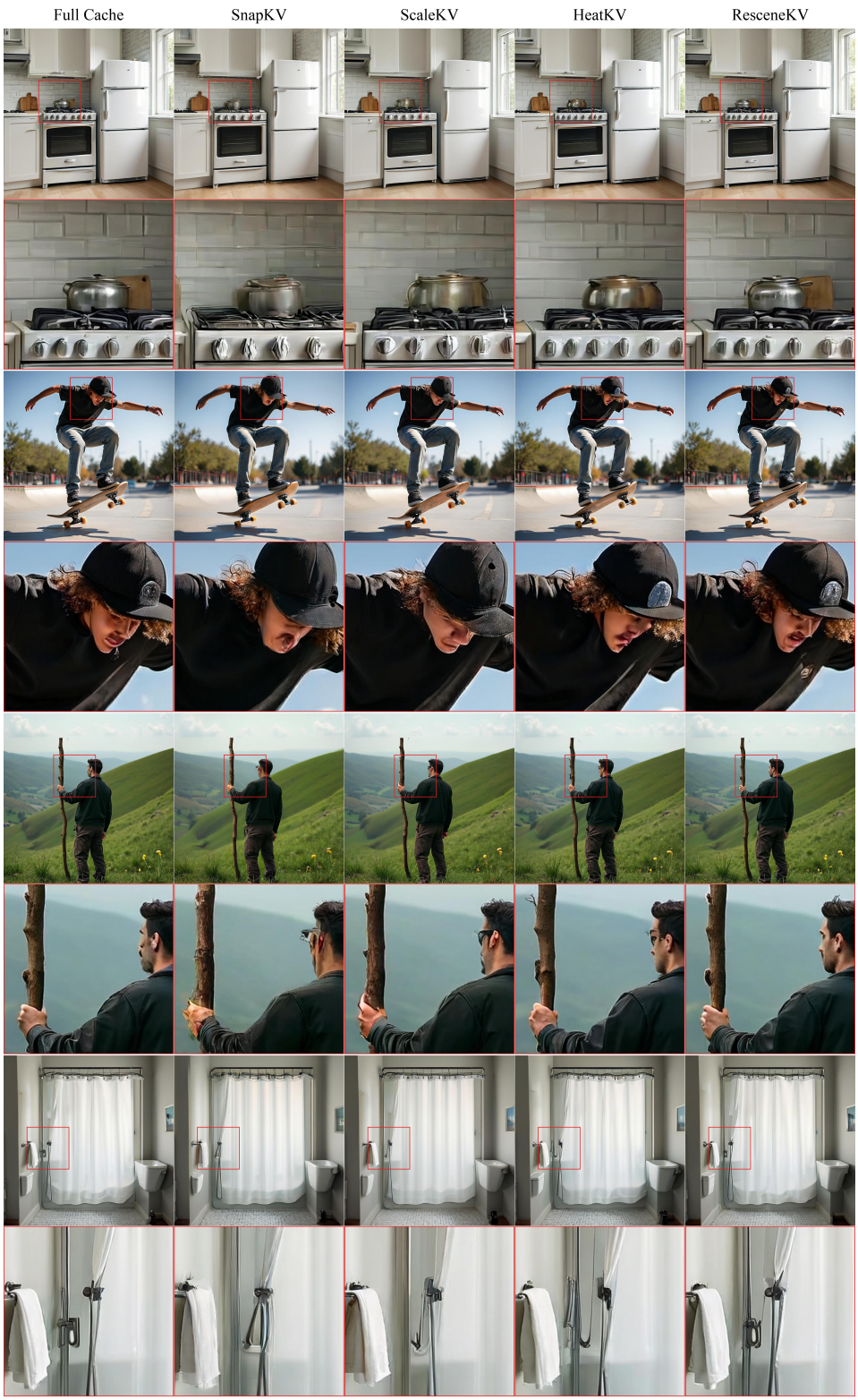

Figure 5: Qualitative comparison on Infinity-2B (Han et al., 2025) at the 4% cache budget.

E ADDITIONAL QUALITATIVE RESULTS ON GENERATION QUALITY

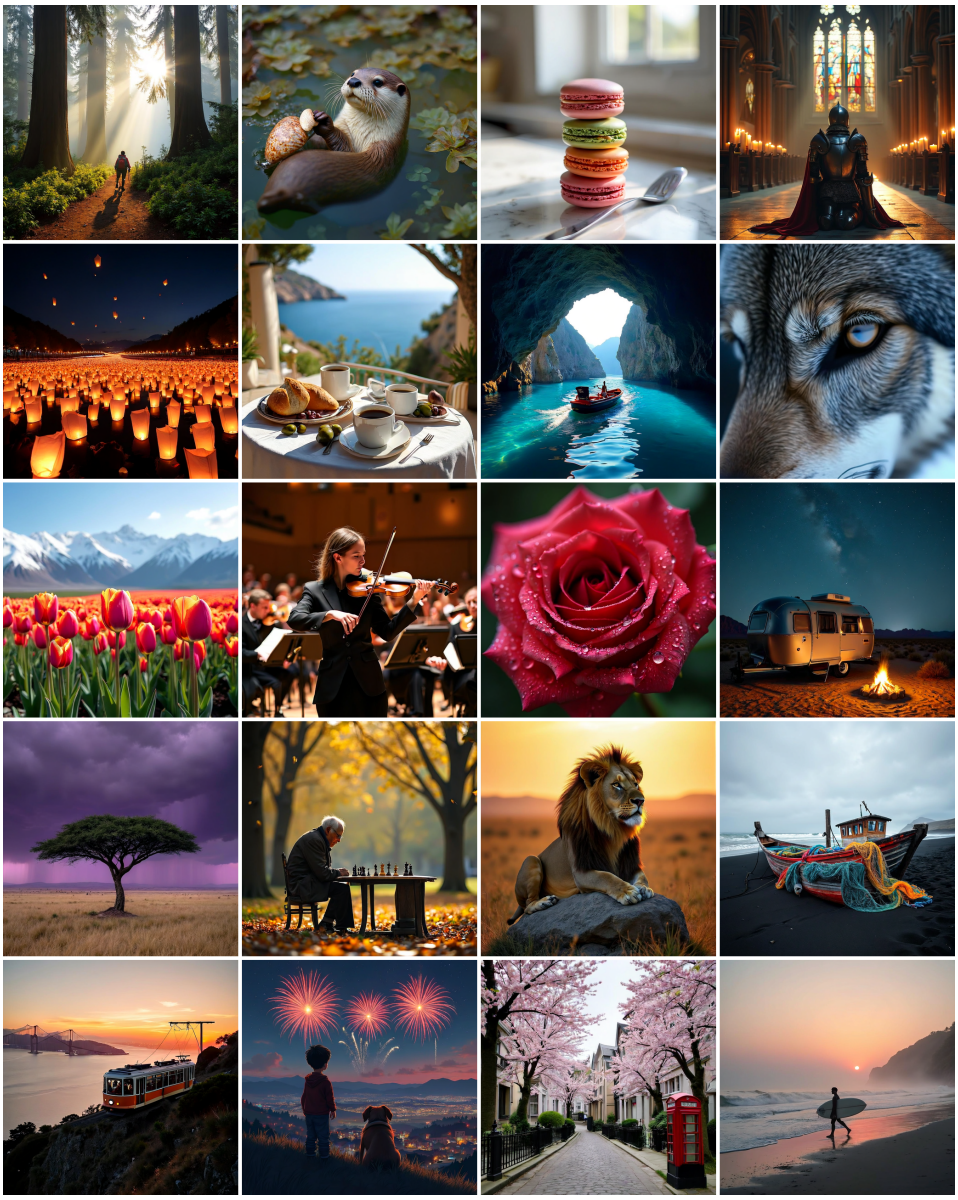

Figure 6: Qualitative results of Infinity-2B (Han et al., 2025) at the 4% budget.



Figure 7: Qualitative results of Infinity-8B (Han et al., 2025) at the 4% budget.



Figure 8: Qualitative results of HART (Tang et al., 2024) at the 10% budget.