UNICACHE: TASK- AND TYPE-AWARE KV CACHE COMPRESSION FOR UNIFIED MULTIMODAL MODELS

Anonymous authors
Paper under double-blind review

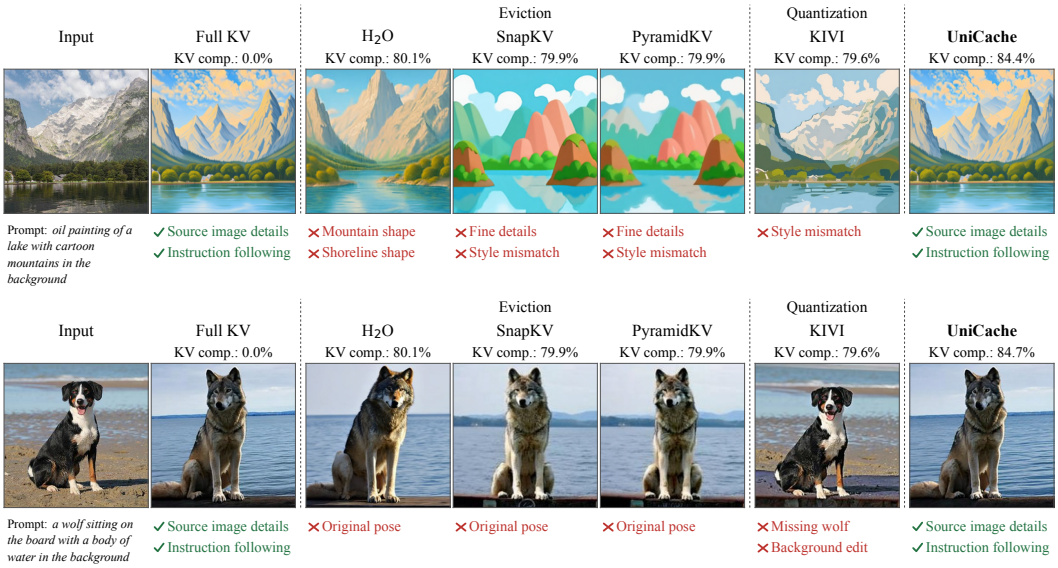

Figure 1: **UniCache is a training-free framework for task- and type-aware KV cache compression in unified multimodal models.** On BAGEL, UniCache achieves $5\times$ KV cache compression for understanding and editing and $2.5\times$ for generation with negligible quality loss, while improving long-context throughput by up to $1.78\times$ over the full-KV baseline. The editing examples illustrate its advantages over uniformly applied compression policies at a target compression rate of 80%.

ABSTRACT

Unified multimodal models combine understanding, generation, and editing within a single network, offering a promising foundation for versatile multimodal applications. However, growing multimodal contexts make KV cache storage and access increasingly costly. Existing KV cache compression methods are typically tailored to specific tasks and single-modality caches, while overlooking changes in cache importance across tasks and timesteps. However, in unified multimodal models, each task involves multiple KV cache types, and both their composition and dynamics differ across tasks. As a result, a single compression policy overlooks task- and type-specific requirements, leading to the loss of critical information and degraded quality across tasks. Based on these findings, we propose **UniCache**, a training-free framework for task- and type-aware KV cache compression. UNICACHE identifies the cache segments activated by each task and assigns suitable compression policies through offline calibration. It coordinates their parallel execution under a shared storage budget through attention-guided allocation and task-aware temporal scheduling. Experiments show that UNICACHE achieves **5**$\times$ KV cache compression for understanding and editing and **2.5**$\times$ for generation with negligible quality loss, while increasing throughput by up to **1.78**$\times$ in long-context settings, significantly improving the practicality of scaling unified multimodal models to longer context.

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.

1 INTRODUCTION

Unified multimodal models integrate understanding and generation within a single network, enabling visual understanding to support image creation and editing from multimodal instructions (Team, 2024; Zhou et al., 2025; Xie et al., 2025; Wu et al., 2025). Recent Mixture-of-Transformers (MoT) models, exemplified by BAGEL, combine modality-specific Transformer experts through shared attention to support these capabilities (Liang et al., 2024; Deng et al., 2025; Diao et al., 2026). However, processing increasingly rich multimodal contexts requires storing and repeatedly accessing large KV caches during inference (Shazeer, 2019; Kwon et al., 2023). As context length grows, the resulting memory footprint and access overhead constrain inference efficiency. KV cache compression therefore offers a practical way to make these models more efficient and support longer multimodal contexts.

Existing KV cache compression methods reduce memory costs through token eviction and quantization (Xiao et al., 2024; Zhang et al., 2023; Cai et al., 2025; He et al., 2024; Liu et al., 2024b; Tu et al., 2025; Yang et al., 2026). Most are designed for a fixed inference paradigm, whereas unified models must manage semantically distinct cache types whose roles change across tasks and inference stages. This challenge is particularly evident in image editing, where the model must follow an editing instruction while preserving source-image content unrelated to the requested change. As shown in Figure 1, eviction-based methods tend to lose source-image details, while quantization preserves those details relatively well but fails to apply the requested edit. These contrasting failures show why a single global compression policy struggles to preserve quality across tasks.

Our analysis of KV cache behavior across tasks in unified multimodal models reveals four challenges underlying these limitations:

- **Task-dependent cache composition.** Unified multimodal models contain multiple KV cache types within each task, and the types involved differ across tasks. Compression therefore cannot assume a fixed set of cache types across all task modes.
- **Type-dependent compression preferences.** KV cache types within the same task differ in their sensitivity to token eviction and quantization. A uniform policy may preserve the information in one KV cache type effectively while disproportionately degrading another.
- **Unequal importance across cache types.** Different KV cache types contribute unequally to attention, so that applying a uniform compression budget cannot account for these differences.
- **Task- and stage-dependent cache importance.** The relative importance of cache types and the overall importance of KV cache evolve differently across tasks and inference stages.

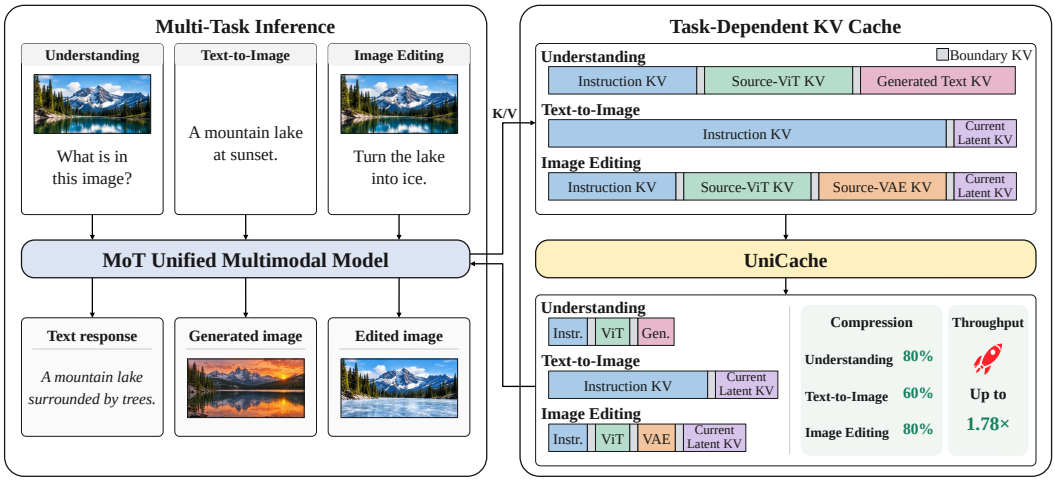

Figure 2: **Left**: Examples of deploying a Mixture-of-Transformers (MoT) unified multimodal model for multi-task inference: image understanding, text-to-image generation and image editing. **Right**: UniCache adopts task- and type-aware KV cache compression and significantly increases throughput and reduces memory overhead for KV cache.

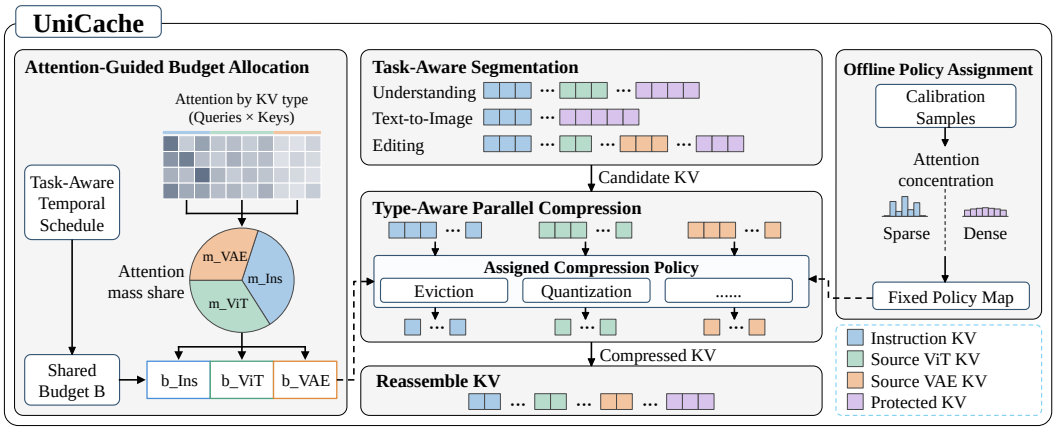

Figure 3: **Overview of UniCache.** It identifies the cache segments activated by each task, assigns suitable compression policies via offline calibration and allocates type-specific compression budgets from the corresponding attention statistics, compressing different cache types online and in parallel.

To address these challenges, we propose **UniCache**, a training-free framework for task- and type-aware KV cache compression. UniCache identifies the KV cache types present in each task and separates compression candidates from protected segments. Through offline calibration of attention distributions, UniCache assigns suitable compression policies to different candidate types. During inference, UniCache determines each segment's storage budget through attention-guided allocation and task-aware temporal scheduling, then applies the assigned compression policies independently and in parallel to their respective segments.

We demonstrate that, across different unified multimodal models and benchmarks, UniCache better preserves quality than uniform compression policies under comparable storage budgets. Figure 1 illustrates how UniCache preserves both source-image details and instruction following. With our efficient inference implementation, UniCache further reduces KV cache memory usage and improves long-context throughput. Notably, on BAGEL, UniCache achieves $5\times$ KV cache compression for understanding and editing and $2.5\times$ for generation with negligible quality loss, while increasing throughput by up to $1.78\times$ in long-context settings.

2 Related Work

Unified multimodal models and inference acceleration. Unified multimodal models integrate visual understanding and visual generation within a single architecture, and recent designs pursue this integration along markedly different paradigms (Team, 2024; Zhou et al., 2025; Xie et al., 2025; Wu et al., 2025). BAGEL (Deng et al., 2025) adopts a Mixture-of-Transformers (MoT) architecture, supporting understanding, image generation, and image editing within one model. As the capability coverage of such models grows, their inference efficiency has drawn increasing attention. Hyper-Bagel (Lu et al., 2025) accelerates both task modes within a single framework, applying speculative decoding to the autoregressive prefill–decode path used for understanding and diffusion distillation to the iterative denoising used for generation. Flash-Unified (Ke et al., 2026) goes further by exploiting the distinct computational profiles of the two task modes, combining per-task network pruning and dynamic layer skipping with visual token pruning for understanding and diffusion-head caching for generation. Together, these methods indicate that unified models differ significantly in their execution characteristics across different tasks, and that inference optimization must therefore be mode-aware. However, the heterogeneity of the KV cache that arises across task modes and across semantically distinct KV cache types remains underexplored.

KV cache compression. Existing KV cache compression methods are designed primarily for autoregressive models, where they reduce cache overhead through token eviction or sparse access. StreamingLLM (Xiao et al., 2024) exploits the attention sink phenomenon by retaining a small set of critical initial tokens, while H_2O (Zhang et al., 2023) and Scissorhands (Liu et al., 2023) decide

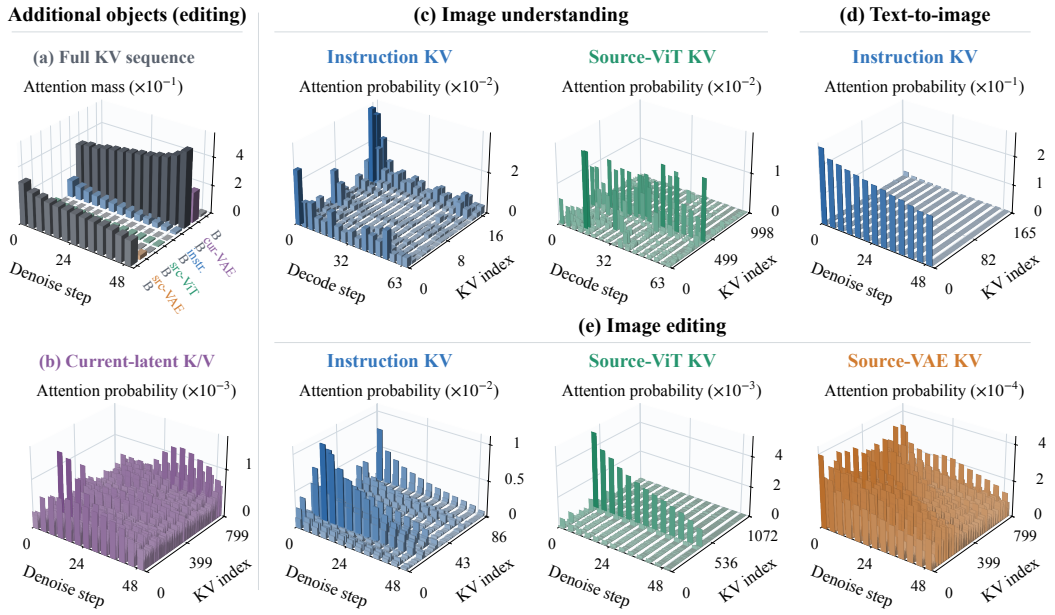

Figure 4: **Task- and type-dependent KV cache attention patterns in BAGEL at layer 8.** (a) Attention mass over the complete editing KV sequence, preserving the original segment order, where *B* denotes boundary KV. (b) Current-latent K/V recomputed at each denoising step. (c–e) Attention distributions of different cache types across image understanding, text-to-image generation, and image editing. We report attention probability averaged over heads and queries.

which KV entries to keep based on the accumulated importance of each token. SnapKV (Li et al., 2024) and PyramidKV (Cai et al., 2025) go further and use observed attention patterns to guide both token selection and the allocation of the cache budget across layers, and DiffKV (Zhang et al., 2025) introduces differentiated memory management for keys and values. However, these designs do not jointly account for the two dimensions of heterogeneity that characterize MoT-based unified multimodal models: task mode and cache type. UniCache fills this gap by coupling task-aware and type-aware cache management: it assigns a compression strategy per cache type and dynamically allocates a shared cache budget according to the task and the current inference stage.

3 UniCache

In this section, we introduce **UniCache**. We first analyze the challenges that unified multimodal models introduce to KV cache compression, then derive four design principles and explain how UniCache addresses these challenges through coordinated policy assignment and budget management. Figure 3 presents the resulting framework.

3.1 Task-Aware Cache Segmentation

Task-dependent cache composition. In MoT-based unified multimodal models, the information required by different tasks is represented by distinct KV cache types, as illustrated in Figure 2. In BAGEL, image understanding uses instruction and source-ViT KVs to represent the instruction and the source image's semantic information, respectively. Text-to-image generation conditions on instruction KV, while image editing uses instruction, source-ViT, and source-VAE KVs together, with source-VAE KV providing source-image details. These conditioning KVs are generated during prefill and reused throughout inference.

Candidate and protected segments. Beyond these conditioning KVs, the attention context contains objects with different management requirements. Boundary KVs often form attention sinks, while

current-latent KV is recomputed at each denoising step, as illustrated in Figure 4(a–b). Their distinct roles and update patterns motivate treating them separately as protected objects.

Design Principle 1: *Identify the KV cache types present in each task and distinguish compression candidates from protected objects according to their roles and update patterns.*

Accordingly, UniCache partitions the attention context into segments based on the current task and semantic role. The conditioning types identified above form the candidate segments passed to subsequent policy assignment or budget allocation, whereas boundary and current-latent KV remain uncompressed and are excluded from the shared compression budget.

3.2 Offline Policy Assignment

Type-dependent compression-policy preferences. The cache types present in a task can also favor different compression policies. As illustrated in Figure 4(c–d), instruction and source-ViT KVs exhibit concentrated attention, which favors eviction policies that retain critical tokens while removing low-contribution tokens. Quantization, however, may perturb the semantic representations of critical instruction KVs. In contrast, source-VAE KVs exhibit broadly distributed attention, with lower attention probabilities for individual tokens, where eviction can discard source-image spatial details, whereas quantization preserves complete token coverage. These different sensitivities help explain the failure modes in Figure 1. Figure 5 also provides quantitative support for these policy preferences under matched compression budgets.

Design Principle 2: *Compression policies should be assigned to cache types according to their attention distributions and compression sensitivity.*

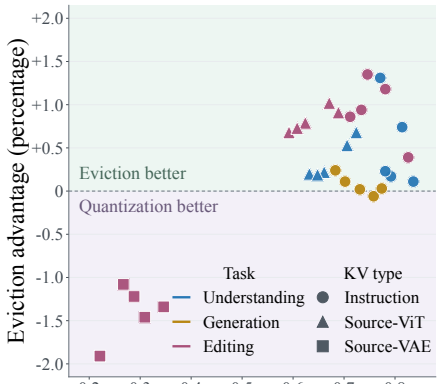

Figure 5: Each point represents a sampled subset for a task–KV-type pair with eviction or quantization at the same compression ratio. Details are provided in Appendix A.

Calibration-based policy assignment. UniCache assigns a compression policy to each candidate task–type pair using attention sparsity measured through offline calibration. These assignments remain fixed during inference, while their compression strengths are adjusted through online budget allocation.

3.3 Attention-Guided Dynamic Budget Allocation

Unequal importance across cache types. After assigning suitable compression policies to different cache types, we must also determine how much budget each policy should retain. As illustrated in Figure 4 and Figure 6, different cache types receive different amounts of attention. A uniform retention ratio would allocate storage in proportion to the original cache sizes, without accounting for these differences.

Design Principle 3: *A global compression budget should be distributed across cache types according to their relative importance.*

Chunk-level attention statistics. UniCache estimates cache importance by aggregating attention probabilities over chunks of layers and inference steps, reducing scheduling frequency. For each candidate cache type, we sum the attention probabilities over its retained keys and average across heads, queries, and layer–step pairs:

$$m_a^{(c)} = \frac{1}{|\Omega_c|} \sum_{(\ell,t)\in\Omega_c} \mathrm{Mean}_{h,q} \left[\sum_{k\in\mathcal{R}_a^{\ell,t}} P_{h,q,k}^{\ell,t} \right]. \tag{1}$$

Here, Ω_c contains the layer–step pairs in chunk c, and $\mathcal{R}_a^{\ell,t}$ contains the retained key indices of candidate cache type a at layer ℓ and step t. $P_{h,q,k}^{\ell,t}$ denotes the attention probability assigned by query q in head h to key k. Statistics collected in chunk c guide budget allocation for chunk $c+1$ within the same layer chunk. We initialize $m_a^{(0)}$ from the final prefill queries for understanding and from the first full-KV denoising step for generation and editing.

Type-level budget allocation. Using these chunk-level attention statistics, UniCache distributes a shared storage budget across candidate cache types in proportion to their attention mass, capped by each type's previous allocation:

$$b_a^{(c+1)} = \min\left\{ b_a^{(c)}, \frac{m_a^{(c)}}{\sum_j m_j^{(c)}} B^{(c+1)} \right\}, \qquad c \geq 0. \tag{2}$$

where $B^{(c+1)}$ denotes the shared storage budget, and $b_a^{(c)}$ denotes the allocated budget of type a within the layer chunk. We initialize $b_a^{(0)} = S_a$, where S_a is the original storage cost of type a. When a type reaches its previous allocation cap, remaining budget is redistributed proportionally among uncapped types with positive attention mass, with a small portion potentially remaining unused due to irreversible eviction.

Task- and stage-dependent cache importance. The allocation above determines how to divide a given budget, but the total budget should also vary across tasks or inference stages, as the model's reliance on conditioning KVs differs across these settings. As illustrated in Figure 6, during understanding, attention contributions of conditional KVs fluctuate with the newly generated tokens. In contrast, during generation and editing, the total attention mass assigned to conditioning KVs decreases as denoising progresses, and different cache types exhibit different rates of decline. Thus, the total budgets should also be temporally scheduled and adapt to task and inference stage.

Design Principle 4: *The total budget of compression should be adjusted over time according to the task and inference stage.*

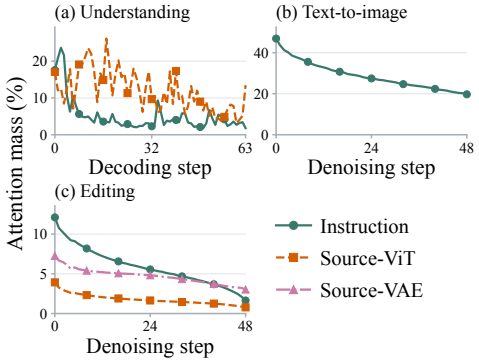

Figure 6: Attention mass across inference steps. Understanding exhibits step-to-step fluctuations, while generation and editing show overall declines at type-dependent rates.

Task-aware total budget. To implement this task-aware temporal scheduling of total budget, UniCache controls the total budget through a retention ratio $\rho^{(c+1)}$, giving $B^{(c+1)} = \rho^{(c+1)} \sum_a S_a$. Understanding uses a fixed task-level retention ratio because the conditioning context remains active throughout autoregressive decoding. For generation and editing, we instead adjust this ratio using the total candidate conditioning mass $M^{(c)} = \sum_a m_a^{(c)}$, relative to its initial full-KV reference $M^{(0)} > 0$. The first compressed chunk uses $\rho^{(1)} = \rho_{\text{init}}$, subsequent updates follow

$$\rho^{(c+1)} = \min\left\{ \rho^{(c)}, \max\left[\rho_{\min}, \rho_{\text{init}} \frac{M^{(c)}}{M^{(0)}} \right] \right\}, \qquad c \geq 1. \tag{3}$$

Here, $0 < \rho_{\min} \leq \rho_{\text{init}} \leq 1$ specify the minimum and initial retention ratios. Through this dynamic retention ratio, the global budget can be temporally scheduled with the relative importance of conditional KVs across tasks and inference stages.

3.4 Type-Aware Parallel Compression

Given these type-specific budgets, UniCache translates each allocation into the configuration of its assigned compression policy: token-eviction policies determine how many KV tokens to retain, while quantization policies determine the numerical precision and grouping granularity. Once configured,

Table 1: **Cross-task results on BAGEL and SenseNova-U1.** MMBench reports circular accuracy (%), MMMU reports validation accuracy (%), and GEdit reports Q_O. Best and second-best compressed results within each model are **bold** and underlined.

Model	Method	Understanding			Generation	Editing			
		MME ↑	MMBench ↑	MMMU ↑	GenEval ↑	GEdit Q_O ↑	PIE Structure ↓	PIE PSNR ↑	PIE CLIP Target ↑
BAGEL	-	2373.21	85.14	53.33	0.781	6.947	0.101	18.805	28.174
	H_2O	2372.60	**85.31**	52.78	0.777	6.860	0.125	14.663	28.567
	KIVI	2362.31	83.68	51.78	0.777	6.929	**0.098**	18.576	27.812
	StreamingLLM	2372.60	**85.31**	52.78	0.313	6.347	0.151	13.021	**29.118**
	SnapKV	2371.01	85.22	51.91	0.558	6.551	0.149	12.932	28.942
	PyramidKV	2367.02	84.88	52.12	0.558	6.325	0.153	12.549	28.912
	UniCache	**2377.36**	**85.31**	**52.89**	**0.780**	**6.965**	0.100	**18.969**	28.211
SenseNova-U1	-	2396.50	75.77	47.67	0.887	6.691	0.099	17.740	28.322
	H_2O	2290.27	75.43	47.67	0.875	4.977	0.142	14.773	27.984
	KIVI	2377.65	74.66	47.00	0.888	3.950	0.131	14.750	28.312
	UniCache	**2383.72**	**75.69**	**48.00**	**0.892**	**6.130**	**0.100**	**17.456**	**29.031**

each policy maintains its own runtime state and compresses only its assigned segments, allowing the segment compression to execute independently and in parallel. After that, the compressed candidates and protected segments are then reassembled in their original token order for attention computation.

4 Empirical Results

4.1 Experimental Setting

Model and Benchmarks We evaluate UniCache on two representative MoT unified multimodal models BAGEL (Deng et al., 2025) and SenseNova-U1 (Diao et al., 2026). We use MME (Fu et al., 2025), MMBench (Liu et al., 2024a), and MMMU (Yue et al., 2024) for understanding, GenEval (Ghosh et al., 2023) for text-to-image generation, and PIE-Bench (Ju et al., 2023) and GEdit-Bench (Liu et al., 2025) with English instructions and Qwen2.5-VL-72B-Instruct (Bai et al., 2025) as the judge for image editing. All experiments use the official evaluation protocols.

Baselines We compare UniCache with the full-KV model and five representative KV cache compression methods: H_2O (Zhang et al., 2023), KIVI (Liu et al., 2024b), StreamingLLM (Xiao et al., 2024), SnapKV (Li et al., 2024), and PyramidKV (Cai et al., 2025). For controlled comparison, all methods are configured to achieve approximately matched KV-memory reductions: around 80% on understanding and editing tasks and around 60% on text-to-image generation.

We also report the detailed settings of compression ratios and calibration in Appendix B and our analysis of calibration statistics for SensoNova-U1 in Appendix C.

4.2 Main Results

As illustrated in Table 1, UniCache effectively mitigates the failure modes encountered by prior KV cache compression methods and achieves balanced quality in all evaluated tasks through task-aware and type-aware KV cache compression.

For understanding tasks, methods that allocate a uniform compression budget introduce a bias between perceptual and cognitive capabilities. For instance, KIVI achieves the highest score in the Perception part in MME but the lowest in the Cognition part, as reported in Table 2. By contrast, UniCache performs comparably to the original model on both dimensions.

For generation and editing, UniCache also retains nearly all of the original model's performance. For text-to-image generation, as illustrated in Table 2, its adaptive budget allocation allows it to outperform existing methods in nearly every subcategory. For image editing, existing methods exhibit distinct limitations: token-eviction methods tend to discard structural details from the input image, whereas quantization methods compromise semantic understanding and instruction-following ability.

Table 2: **Detailed MME and GenEval results on BAGEL.** Comp. denotes KV-cache compression. Higher is better for all quality metrics. Best and second-best results among compressed methods are **bold** and underlined, respectively.

Method	MME				GenEval									
	Comp.	Perception	Cognition	Total	Comp.	Single	Two	Count	Color	Position	Binding	Image Acc.	Prompt Acc.	Overall
BF16	0%	1680.35	692.86	2373.21	0%	0.997	0.922	0.781	0.846	0.520	0.623	0.773	0.897	0.781
H_2O	80.07%	1679.74	692.86	2372.60	59.47%	0.997	0.909	**0.788**	**0.864**	0.505	0.600	0.768	0.902	0.777
KIVI	80.69%	**1685.53**	676.79	2362.31	59.12%	**1.000**	0.912	0.778	0.840	0.505	**0.628**	0.768	0.897	0.777
StreamingLLM	79.94%	1679.74	692.86	2372.60	58.01%	0.863	0.051	0.419	0.540	0.003	0.003	0.287	0.405	0.313
SnapKV	79.94%	1679.58	691.43	2371.01	61.25%	0.997	0.503	0.659	0.832	0.095	0.263	0.536	0.673	0.558
PyramidKV	79.94%	1676.30	690.71	2367.02	61.25%	0.997	0.505	0.659	0.832	0.095	0.260	0.536	0.675	0.558
UniCache	80.00%	1681.65	**695.71**	**2377.36**	59.76%	**1.000**	**0.929**	0.763	**0.864**	**0.515**	0.608	**0.771**	**0.906**	**0.780**

Table 3: **Method component ablations across tasks.** Targets are approximately 80% compression on MME and 60% on GenEval and PIE-Bench. Dashes indicate inapplicable settings.

	MME			GenEval	PIE-Bench		
Setting	Perception ↑	Cognition ↑	Total ↑	Overall ↑	Structure ↓	PSNR ↑	CLIP Target ↑
BAGEL	1680.35	692.86	2373.21	0.781	0.101	18.805	28.174
Full UniCache	1681.65	695.71	2377.36	0.780	0.101	19.100	28.171
w/o Task/Type Awareness	1679.74	692.86	2372.60	0.767	0.103	17.875	28.221
w/o Protection	1680.47	692.86	2373.32	0.567	0.102	18.959	27.235
w/o Attention-Guided Allocation	1680.32	688.01	2368.33	—	0.121	17.277	27.036
w/o Task-Aware Temporal Schedule	—	—	—	0.766	0.110	18.477	28.010

Figure 1 provides qualitative visual results. UniCache preserves different kinds of information and avoids these failure modes even under high compression ratios.

4.3 Ablation Studies

Component Analysis To assess the contribution of each component, we conduct ablation studies under a consistent experimental setup. Specifically, we remove each component from UniCache individually and evaluate the resulting variants across several benchmarks. As shown in Table 3, removing the protection of boundary and current-latent KV leads to degraded text-to-image generation performance. Moreover, attention-guided allocation is particularly critical for image understanding and image editing which involve multiple KV-cache types. Given the same total cache budget, the task-aware temporal schedule allocates more KV cache capacity to the early stages of generation and less to the later stages, so removing it causes slightly degradation in generation tasks as the uniform budget allocation across steps does not track the decline in attention mass of conditional KV across denoising steps. Collectively, these components enable UniCache to deliver balanced performance across tasks.

Temporal schedule. We further compare attention-guided scheduling against two alternatives: a constant budget and a linear decay schedule. The former maintains a fixed retention ratio throughout denoising, whereas the latter linearly interpolates from ρ_{init} to $\rho_{\min}$ as denoising progresses. All other components and compression policies are held fixed as described in Section 4.1. As shown in Table 4, attention-guided decay matches linear decay on GenEval and achieves the best results across all PIE-Bench metrics, showing that conditioning attention provides a more effective temporal signal for budget allocation across steps.

Table 4: **Ablation of temporal budget schedules.** We change the schedule methods for global budget scheduling and report results in GenEval and PIE-Bench.

Schedule	**GenEval ↑**	**Structure ↓**	**LPIPS ↓**	**SSIM ↑**	**CLIP Target ↑**	**CLIP Edit ↑**
Constant	0.776	0.102	0.184	0.757	28.010	27.070
Linear decay	**0.780**	0.102	0.184	0.757	28.187	27.285
Attention-guided decay	**0.780**	**0.101**	**0.179**	**0.763**	**28.211**	**27.293**

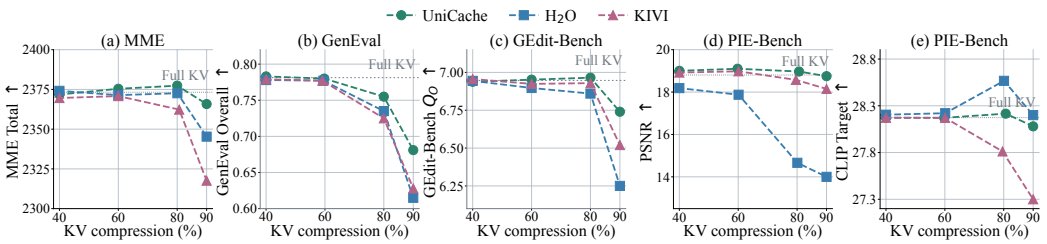

Figure 7: **Ablation of compression ratio across different methods and benchmarks.** Across different benchmarks, UniCache is robust to high compression ratio and maintains balanced performance at very high compression ratios.

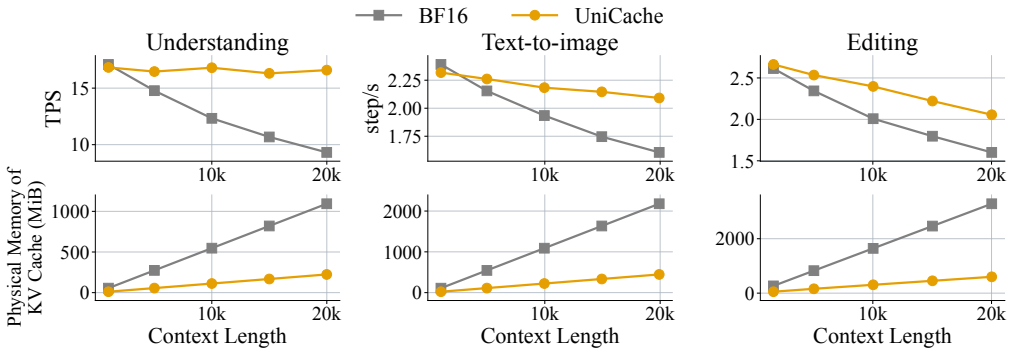

Figure 8: **Runtime efficiency of UniCache across context lengths.** Throughput is measured in tokens per second for understanding and denoising steps per second for generation and editing. Memory reports the persistent managed KV cache memory summed over all layers.

Compression ratio. We also compare UniCache with KIVI and H_2O in different compression ratios and report the results in Figure 7. Across different benchmarks, UniCache is more robust to high compression ratio than other methods. Notably, in PIE-Bench, H_2O and KIVI cannot avoid degradation in at least one of the two metrics, while UniCache maintains balanced performance across both metrics.

4.4 End-to-End Efficiency

We implement an efficient framework with physical KV compaction and low-bit memory, as introduced in Appendix E. All measurements are conducted on a single NVIDIA A100 GPU (40 GB). We report detailed results of BAGEL-7B-MoT in Figure 8. The results show that the benefit of physical compression increases with context length. At approximately 20K KV tokens per attention branch, UniCache improves throughput by up to $1.78\times$ while reducing persistent managed KV cache memory by about 80%.

5 Conclusion

In this work, we show that heterogeneous cache types and task-dependent dynamics limit existing KV-cache compression methods in MoT-based unified multimodal models. To address this challenge, we introduce UniCache, a training-free framework that combines offline type-aware policy assignment with attention-guided budget allocation and task-aware temporal scheduling. By coordinating different compression policies across cache segments, UniCache preserves quality across understanding, generation, and editing, achieving up to $5\times$ KV-cache compression with negligible performance loss and up to $1.78\times$ long-context throughput, improving the practicality of scaling unified multimodal model to longer context.

AI USE STATEMENT

Generative AI tools were used to improve the readability of the manuscript and to assist with minor code editing. All AI-assisted text and code were reviewed and verified by the authors. Generative AI tools were not used to formulate the research hypotheses, design the methodology, interpret the experimental results, or derive the main scientific claims. The authors take full responsibility for the content of this work.

REPRODUCIBILITY STATEMENT

We provide implementation details, compression configurations, and evaluation protocols in the main text and appendix. Our code is included in the supplementary material. All experiments use publicly available models and benchmarks.

ETHICS STATEMENT

This work studies inference efficiency in unified multimodal models using publicly available models and benchmarks. UniCache does not address biases or harmful-generation risks inherited from the underlying models. More efficient inference may facilitate both beneficial applications and misuse, deployments should therefore retain appropriate safety safeguards.

REFERENCES

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report, 2025. URL https://arxiv.org/abs/2502.13923.

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, 2025. URL https://arxiv.org/abs/2406.02069.

Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

Haiwen Diao, Penghao Wu, Hanming Deng, Jiahao Wang, Shihao Bai, Silei Wu, Weichen Fan, Wenjie Ye, Wenwen Tong, Xiangyu Fan, Yan Li, Yubo Wang, Zhijie Cao, Zhiqian Lin, Zhitao Yang, Zhongang Cai, Yuwei Niu, Yue Zhu, Bo Liu, Chengguang Lv, Haojia Yu, Haozhe Xie, Hongli Wang, Jianan Fan, Jiaqi Li, Jiefan Lu, Jingcheng Ni, Junxiang Xu, Kaihuan Liang, Lianqiang Shi, Linjun Dai, Linyan Wang, Oscar Qian, Peng Gao, Pengfei Liu, Qingping Sun, Rui Shen, Ruisi Wang, Shengnan Ma, Shuang Yang, Siyi Xie, Siying Li, Tianbo Zhong, Xiangli Kong, Xuanke Shi, Yang Gao, Yongqiang Yao, Yves Wang, Zhengqi Bai, Zhengyu Lin, Zixin Yin, Wenxiu Sun, Ruihao Gong, Quan Wang, Lewei Lu, Lei Yang, Ziwei Liu, and Dahua Lin. Sensenova-u1: Unifying multimodal understanding and generation with neo-unify architecture, 2026. URL https://arxiv.org/abs/2605.12500.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, Rongrong Ji, Caifeng Shan, and Ran He. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2025. URL https://arxiv.org/abs/2306.13394.

Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36: 52132–52152, 2023.

Yefei He, Feng Chen, Jing Liu, Wenqi Shao, Hong Zhou, Kaipeng Zhang, and Bohan Zhuang. Zipvl: Efficient large vision-language models with dynamic token sparsification. *arXiv preprint arXiv:2410.08584*, 2024.

Xuan Ju, Ailing Zeng, Yuxuan Bian, Shaoteng Liu, and Qiang Xu. Direct inversion: Boosting diffusion-based editing with 3 lines of code. *arXiv preprint arXiv:2310.01506*, 2023.

Junlong Ke, Zichen Wen, Boxue Yang, Yantai Yang, Xuyang Liu, Chenfei Liao, Zhaorun Chen, Shaobo Wang, and Linfeng Zhang. Flash-unified: A training-free and task-aware acceleration framework for native unified models, 2026. URL https://arxiv.org/abs/2603.15271.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

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, 2024. URL https://arxiv.org/abs/2404.14469.

Weixin Liang, Lili Yu, Liang Luo, Srinivasan Iyer, Ning Dong, Chunting Zhou, Gargi Ghosh, Mike Lewis, Wen-tau Yih, Luke Zettlemoyer, et al. Mixture-of-transformers: A sparse and scalable architecture for multi-modal foundation models. *arXiv preprint arXiv:2411.04996*, 2024.

Shiyu Liu, Yucheng Han, Peng Xing, Fukun Yin, Rui Wang, Wei Cheng, Jiaqi Liao, Yingming Wang, Honghao Fu, Chunrui Han, et al. Step1x-edit: A practical framework for general image editing. *arXiv preprint arXiv:2504.17761*, 2025.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. Mmbench: Is your multi-modal model an all-around player? In *European conference on computer vision*, pp. 216–233. Springer, 2024a.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time, 2023. URL https://arxiv.org/abs/2305.17118.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024b.

Yanzuo Lu, Xin Xia, Manlin Zhang, Huafeng Kuang, Jianbin Zheng, Yuxi Ren, and Xuefeng Xiao. Hyper-bagel: A unified acceleration framework for multimodal understanding and generation, 2025. URL https://arxiv.org/abs/2509.18824.

Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

Chameleon Team. Chameleon: Mixed-modal early-fusion foundation models. *arXiv preprint arXiv:2405.09818*, 2024.

Dezhan Tu, Danylo Vashchilenko, Yuzhe Lu, and Panpan Xu. Vl-cache: Sparsity and modality-aware kv cache compression for vision-language model inference acceleration. In *International Conference on Learning Representations*, volume 2025, pp. 219–239, 2025.

Chengyue Wu, Xiaokang Chen, Zhiyu Wu, Yiyang Ma, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, Chong Ruan, et al. Janus: Decoupling visual encoding for unified multimodal understanding and generation. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 12966–12977. IEEE, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks, 2024. URL https://arxiv.org/abs/2309.17453.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *International Conference on Learning Representations*, volume 2025, pp. 28240–28264, 2025.

Wanqi Yang, Yuexiao Ma, Alexander Conzelmann, Xiawu Zheng, Michael W Mahoney, T Konstantin Rusch, and Shiwei Liu. Alphaq: Calibration-free bit allocation for mixture-of-experts quantization. *arXiv preprint arXiv:2606.04980*, 2026.

Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, et al. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9556–9567, 2024.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John CS Lui, and Haibo Chen. Diffkv: Differentiated memory management for large language models with parallel kv compaction. In *Proceedings of the ACM SIGOPS 31st Symposium on Operating Systems Principles*, pp. 431–445, 2025.

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.

Chunting Zhou, Lili Yu, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *International Conference on Learning Representations*, volume 2025, pp. 6446–6469, 2025.

A Attention Concentration and Policy Preference Experiments

To measuring the relationship between attention concentration and policy preference, for each task–KV-type pair, we evaluate five sampled subsets, each containing 100 examples from the corresponding benchmark MME, GenEval and PIE-Bench. We compare H_2O and KIVI as representative methods of eviction and quantization under matched KV-memory budgets, applying compression only to the selected KV type and leaving the remaining types unchanged. Each compressed run is evaluated against its corresponding full-KV baseline using the task's primary metric.

To compare different policies, the metric we use to evaluate relative quality degradation is $D_\pi = 100(s_{\text{full}} - s_\pi)/s_{\text{full}}$. The numerator is reversed for a lower-is-better metric. Then we define eviction advantage as $D_{\text{KIVI}} - D_{\text{H}_2\text{O}}$, measured in percentage points. As a result, positive values favor eviction and negative values favor quantization. We report results in Figure 5.

B Detailed Experimental Settings

Compression Ratio For each experiment, we set a target compression ratio and record the realized compression ratio, that the method actually achieves. We report the target and realized KV cache compression ratio in Table 5.

Table 5: Target and realized KV cache compression ratios (%) on BAGEL.

Method	Understanding	Text-to-Image	Image Editing
Target compression	**80.00**	**60.00**	**80.00**
H_2O	80.07	59.47	80.09
KIVI	80.69	59.12	79.59
StreamingLLM	79.94	58.01	79.94
SnapKV	79.94	61.25	79.93
PyramidKV	79.94	61.25	79.93
UniCache	80.00	59.76	80.67

Calibration Setting We use calibration to determine compression policy. For each task, we randomly sample 10 calibration examples and measure attention concentration as the fraction of within-type attention mass captured by the top 10% of tokens, averaged across examples. We assign eviction to task–type pairs with concentration at least 0.5 and quantization otherwise, while calibrating $\rho_{\min}$ to match the target average KV compression ratio.

C Attention Analysis of SenseNova-U1

Unlike BAGEL, SenseNova-U1 represents the source image with a single native visual KV type. As shown in Figure 9, our offline calibration finds that attention over this KV type is also concentrated on a small subset of tokens. UniCache therefore automatically assigns an eviction policy to source-native-visual KV, adapting its policy assignment to the model's cache structure and attention distribution.

D Sampled Results Generated by UniCache

Figures 10 and 11 show generation and editing samples produced by UniCache at 60% and 80% KV cache compression, respectively.

E Efficient GPU Implementation

We provide both a PyTorch reference implementation and an optimized GPU inference engine for UniCache. The reference implementation uses attention masks and fake quantization to evaluate

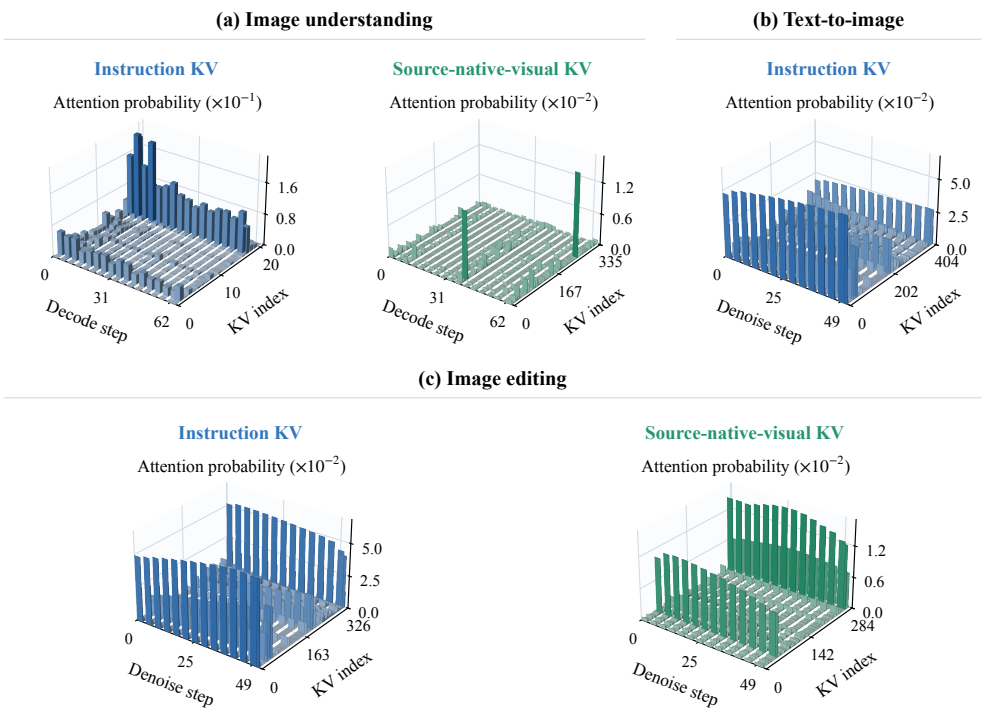

Figure 9: **Task- and type-dependent KV-cache attention patterns in SenseNova-U1 at layer 8.** Each panel shows token-level attention probability averaged over heads and queries for one example, with tokens kept in their original order. Source-native-visual KV denotes SenseNova-U1's native source-image representation.

compression policies, whereas the GPU engine physically compacts or packs KV entries to reduce persistent cache storage and attention overhead.

- **Physical, type-aware KV storage.** For eviction-managed segments, we store only retained KV entries and their logical token indices. Quantized segments use packed low-bit representations with scales, minima, and a short BFloat16 residual tail. Protected segments retain their original representations.
- **Batched GQA token selection.** Attention scores are aggregated across query heads sharing the same KV head. We then perform heavy-hitter and recent-token selection independently for each KV head using batched operations, reducing Python dispatch overhead.
- **Attention over heterogeneous cache segments.** The engine combines cache segments under a shared softmax normalization. For multi-query denoising, we use Triton kernels to dequantize packed KVs into reusable temporary workspaces for FlashAttention. When attention is evaluated in separate chunks, their outputs are merged using log-sum-exp statistics to preserve global normalization.
- **Efficient attention-guided allocation.** Since FlashAttention/SDPA kernels do not explicitly materialize or store the full attention probability matrix, we implement an accelerated kernel over packed KVs that computes attention statistics at the chunk level and efficiently accumulates the attention mass required for budget allocation. Before processing each chunk, a lightweight custom kernel determines the budget allocation for the next chunk.
- **Cached metadata and execution plans.** At runtime, we reuse segment boundaries and type identifiers. Once a physical layout is frozen, attention bypasses repeated segment matching and generic operator preparation.
- **Batched attention and CFG execution.** Compatible attention calls are combined through variable-length FlashAttention, while classifier-free guidance branches are processed in configurable batches to reduce repeated kernel launches.

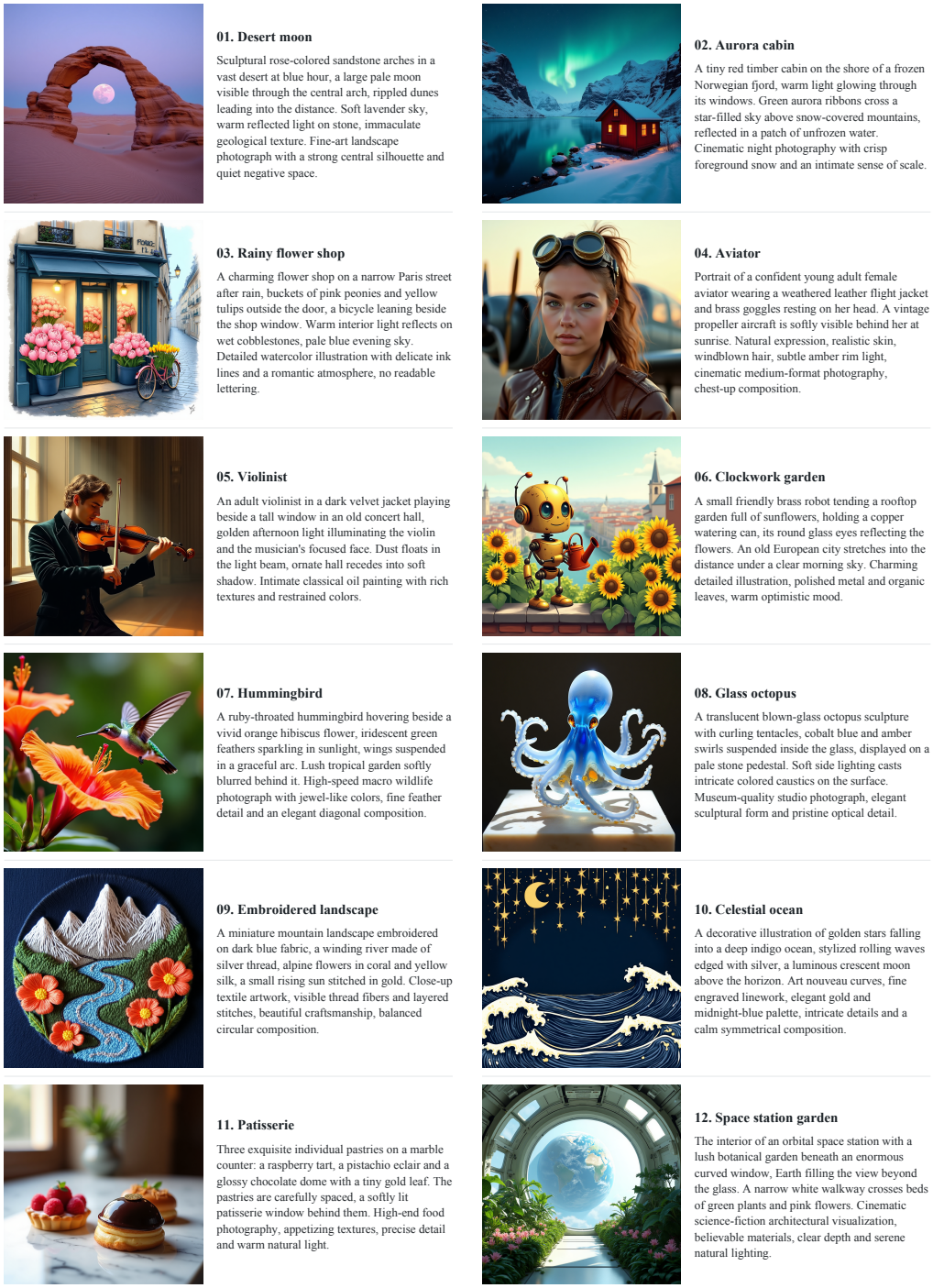

Figure 10: **Image generation samples produced by our UniCache optimized BAGEL model.** The compression rate is 60%. The model successfully handles complex descriptions, rendering high-quality textures and correct object placements, demonstrating that our KV cache compression framework preserves the generative performance of the model.

The engine targets BF16 inference on NVIDIA GPUs. In our efficiency evaluation, attention-guided allocation introduces only 0.45%–1.7% additional overhead, depending on the chunk size, which is small relative to the throughput and memory savings enabled by the overall system.



Figure 11: **Image editing samples produced by our UniCache-optimized BAGEL model.** Each group shows the source image, editing prompt, and edited output from left to right. UniCache preserves the original model's ability to follow instructions while preserving visual content unrelated to the requested edits.