COMPRESSING KV-CACHE COMMUNICATION IN MULTI-AGENT SYSTEMS WITH COLLECTIVE COMPENSATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent LLM systems are moving beyond discrete-token messages toward richer relays that preserve internal state. Recent work such as LatentMAS transmits full key-value (KV) caches between agents but pays a high memory and communication cost. We adapt KV-cache eviction to inter-agent relay and introduce **Orthogonal BackFill (OBF)**. OBF collectively recovers complementary information from discarded values and incorporates it into the retained values. Eviction reduces per-agent relay volume by about $4.7\times$, with peak GPU memory savings of 5%–15% across model sizes. Across seven benchmarks at Qwen3-4B, compressed relay retains only 9.9%–20.2% of prompt KV while remaining close to full relay in accuracy and exceeding it on both coding benchmarks. At the same relay budget, adding OBF improves accuracy over eviction alone in 31 of 42 comparisons across Qwen3-4B, 8B, and 14B, with headwise OBF improving in 17 of 21 comparisons.

1 INTRODUCTION

Large language model (LLM) based multi-agent systems (MAS) rely on communication to coordinate reasoning, critique intermediate outputs, and refine final decisions. Early multi-agent frameworks use natural language as the communication medium, where agents exchange explicit messages under predefined roles or protocols (Li et al., 2023; Hong et al., 2024; Chen et al., 2025). Text-based communication is lossy. The model produces a probability distribution over the vocabulary, but only one sampled token is transmitted, discarding all other plausible alternatives. This has pushed work toward richer communication channels that preserve more of the model's internal computation (Pham et al., 2024; Tang et al., 2025). LatentMAS (LMAS) pairs latent reasoning with lossless inter-agent communication. Each agent reasons in continuous latent space and relays its full key-value (KV) cache directly to the next agent, so downstream reasoning can continue from internal states rather than from text alone (Zou et al., 2025). That fidelity is expensive. Relaying full KV caches consumes memory and bandwidth that grow with the number of agents, rounds, and prompt tokens.

A natural solution is to compress the communicated KV states, as recent work on efficient LLM inference suggests that not all cached states are equally important. Methods such as StreamingLLM, H_2O, and Scissorhands show that selective retention or eviction of KV states can reduce memory usage while preserving generation quality in single-agent settings (Xiao et al., 2024; Zhang et al., 2023; Liu et al., 2023b). However, these methods target online cache management during token-by-token generation within a single agent. In LatentMAS-style communication, compression instead occurs at the relay boundary between agents, where the relevant question is whether the transmitted states remain useful for *downstream continuation by another agent*.

In this work, we adapt KV-cache eviction methods to inter-agent relay and propose **Orthogonal BackFill (OBF)**. Under a fixed relay budget, we aim to make each retained state carry as much signal as possible for the downstream agent. OBF extracts from the discarded value states a low-rank component orthogonal to the retained ones and injects it into the retained values, so that information complementary to the retained set is preserved.

Our experiments show that compressed relay can match full KV relay at a fraction of the bandwidth, and that what governs relay quality is whether the preserved state stays useful to the downstream

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.

agent rather than how much of it survives. We further study how selector granularity and OBF design affect this trade-off across tasks.

Our contributions are as follows.

- We formulate KV-cache communication between agents as a compression problem under a fixed relay budget. A four-part cache decomposition distinguishes local prompt states from inherited history and current reasoning, enabling eviction rules to be adapted to inter-agent relay.
- We identify the relay boundary as an opportunity for collective compensation and propose **Orthogonal BackFill (OBF)** to exploit it. OBF jointly extracts complementary information from discarded values through orthogonal projection and low-rank summarization, then incorporates it into the retained values.
- We show that compressed relay remains competitive with full relay across seven benchmarks and Qwen3 models at 4B, 8B, and 14B, while retaining only 9.9%–20.2% of prompt KV.

2 Related Work

Why Multi-Agent Systems. MAS outperform single-agent baselines on complex tasks such as software engineering, mathematical reasoning, and deliberative evaluation (Hong et al., 2024; Du et al., 2023; Chan et al., 2023; Liang et al., 2024). They typically distribute reasoning across roles, prompts, or subtasks rather than forcing the entire problem into a single trajectory (Li et al., 2023; Wu et al., 2023; Zhou et al., 2023). All of these frameworks communicate through natural language, even though text is a lossy compression of each agent's internal state and is not strictly necessary for inter-agent exchange. This motivates relaying internal state without decoding it to text.

From Language to Latent Communication. Early MAS frameworks such as **CAMEL** (Li et al., 2023) and **MetaGPT** (Hong et al., 2024) organize collaboration through natural-language exchanges under predefined roles and workflows, while later systems such as **Optima** (Chen et al., 2025) train agent policies to shorten messages and reduce redundant exchanges. More recent work moves beyond text-only communication and explores richer, latent-state carriers for inter-agent information exchange. **CIPHER** (Pham et al., 2024) communicates through expected output embeddings, **StateDelta** (Tang et al., 2025) augments language with hidden-state changes, and **COCONUT** (Hao et al., 2025) studies latent-space reasoning without decoding intermediate steps into text.

LatentMAS and KV Relay. Our setting is most directly related to LMAS (Zou et al., 2025), a training-free framework for pure latent collaboration among LLM agents. Instead of passing decoded intermediate messages, LMAS transfers the full KV cache from one agent to the next, so downstream reasoning continues directly from the predecessor's internal state. This yields higher fidelity than text-based communication, but it also turns communicated KV into the dominant communication object. We address this bottleneck by compressing at the relay boundary rather than modifying the reasoning procedure itself. Cache-to-Cache also explores KV-cache communication (Fu et al., 2026).

KV Compression, Eviction, and Compensation. A separate line of work studies how to reduce KV-cache cost in long-context inference. Quantization methods such as **KVQuant** (Hooper et al., 2025) reduce the footprint of retained KV tensors, while eviction methods such as $\mathbf{H_2O}$ (Zhang et al., 2023), **StreamingLLM** (Xiao et al., 2024), and **Scissorhands** (Liu et al., 2023b) selectively retain only the most useful past states. More recent methods mitigate the information loss from hard eviction through compensation or merging. **RazorAttention** (Tang et al., 2024) summarizes removed context through compensation tokens in non-retrieval heads, while **CaM** (Zhang et al., 2024) and **WeightedKV** (Yuan et al., 2025) merge discarded information into retained states instead of removing it outright. All of these target single-agent online inference, where compression interleaves with token generation rather than acting at an inter-agent boundary.

Positioning. In contrast, we compress once at the relay boundary, evicting a whole block rather than token by token during generation, and ask only whether the result still serves the next agent. Appendix E evaluates relay-adapted cache merging and quantization baselines. OBF also injects only the orthogonal residual and **broadcasts one low-rank correction to all retained values** rather than merging deleted tokens individually.

3 PRELIMINARIES

3.1 KV CACHE NOTATION

For an autoregressive Transformer decoder processing a token sequence $Y = \{y_1, \ldots, y_T\}$, at each layer $l \in \{1, \ldots, L_{\text{layer}}\}$ with H attention heads the self-attention cache contains

$$\mathbf{K}^{(l)} \in \mathbb{R}^{T \times H \times d_k}, \qquad \mathbf{V}^{(l)} \in \mathbb{R}^{T \times H \times d_v}.$$

The per-head slices are

$$\mathbf{K}^{(l,h)} := \mathbf{K}^{(l)}_{:,h,:} \in \mathbb{R}^{T \times d_k}, \qquad \mathbf{V}^{(l,h)} := \mathbf{V}^{(l)}_{:,h,:} \in \mathbb{R}^{T \times d_v}, \qquad h \in \{1, \ldots, H\}.$$

Token-level and full-cache views. At a fixed layer, the cache forms a token-indexed sequence $\mathbf{S}^{(l)} := \{(\mathbf{K}^{(l)}_{t,:,:}, \mathbf{V}^{(l)}_{t,:,:})\}_{t=1}^{T}$. The full cache is $\mathrm{KV}(Y) = \{\mathbf{K}^{(l)}, \mathbf{V}^{(l)}\}_{l=1}^{L_{\text{layer}}}$, and we write $\mathbf{S} := \mathrm{KV}(Y)$ when treating it as a single token-indexed object, with $\mathbf{S}[t]$ the state at position t.

Index selection and concatenation. Any retained cache is a subsequence of $\mathbf{S}$ selected by an index list. For an ordered index list $I = (i_1, \ldots, i_m)$ with $1 \le i_1 < \cdots < i_m \le T$, $\mathbf{S}_I[r] = \mathbf{S}[i_r]$ and $\mathbf{S}_I \in \mathbb{R}^{m \times D_{KV}}$. Row-wise concatenation $[\mathbf{S}_{I_a}; \mathbf{S}_{I_b}]$ corresponds to index list $I_a \| I_b$.

3.2 AGENTIC COMMUNICATION IN LMAS

In LMAS, agents communicate through latent messages represented by Transformer KV-cache states produced during autoregressive latent inference. Let π_i denote the i-th agent in the communication chain. After completing its local computation, π_i transmits a message $\mathcal{M}_i$, where $\mathcal{M}_i = [\mathcal{M}_{i-1}; \mathrm{KV}(Y_i)]$. Here, $\mathrm{KV}(Y_i)$ denotes the token-level KV-cache states produced by agent π_i. We decompose these states as $\mathrm{KV}(Y_i) = [\mathbf{S}_{P_i}; \mathbf{S}_{G_i}]$, where $\mathbf{S}_{P_i}$ is induced by the local prompt P_i and $\mathbf{S}_{G_i}$ by autoregressive latent generation. The receiver π_{i+1} continues decoding by using $\mathcal{M}_i$ as `PastKV`. Because each $\mathrm{KV}(Y_k)$ is appended in full, message length grows linearly with the number of agents. We formalize the resulting compression problem in Section 4.1.

4 METHODOLOGY

We extend single-agent KV-cache eviction to multi-agent latent communication. Building on a four-part cache decomposition, we adapt single-agent eviction rules such as StreamingLLM and H_2O to the relay setting, and propose **OBF** to address the limitations of direct eviction.

4.1 PROBLEM FORMULATION

We formulate inter-agent relay as a KV compression problem. Since $|\mathcal{M}_i|$ grows linearly with the number of agents under full relay ($|\mathcal{M}_i| \propto \sum_{k=0}^{i} |\mathrm{KV}(Y_k)|$), each agent compresses its prompt states before transmission. Let Φ denote a compression operator acting on the prompt states $\mathbf{S}_{P_i}$, optionally conditioned on the reasoning states $\mathbf{S}_{G_i}$:

$$\mathcal{M}_i \;=\; [\mathcal{M}_{i-1};\; \Phi(\mathbf{S}_{P_i};\, \mathbf{S}_{G_i});\; \mathbf{S}_{G_i}].$$

Under a per-agent budget B that retains at most B eligible prompt positions, our goal is to design Φ that maximizes downstream task accuracy at a fixed relay budget.

4.2 FROM SINGLE-AGENT KV-CACHE EVICTION TO MULTI-AGENT CACHE DECOMPOSITION

Existing single-agent eviction methods implicitly decompose the KV cache into three functional parts: an **attention sink** of initial tokens that stabilize the softmax distribution (Xiao et al., 2024); a **candidate pool** of middle tokens targeted for importance-based eviction (Zhang et al., 2023; Liu et al., 2023b); and a **local window** of recent tokens (Xiao et al., 2024; Zhang et al., 2023).

In multi-agent collaboration, latent memory is no longer produced by a single homogeneous stream. Instead, it accumulates across agent boundaries as alternating prompt and generation segments,

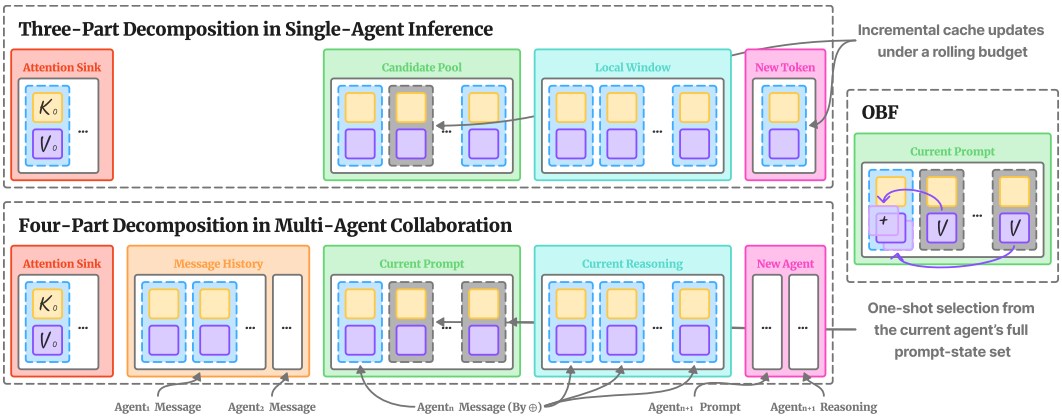


Figure 1: **Overview of role decomposition and the OBF compression operator.** Top: single-agent inference uses three roles (sink, candidate pool, local window) and updates the cache incrementally under a rolling budget. Bottom: our multi-agent setup adds inherited message history and selects retained prompt states in a single shot. Right inset: OBF previews residual injection into the retained values. See Figure 2 for eviction granularities and Figure 3 for OBF mechanics.

e.g., $[\mathbf{S}_{P_1}; \mathbf{S}_{G_1}; \mathbf{S}_{P_2}; \mathbf{S}_{G_2}; \dots] = [\mathcal{M}_{i-1}; \mathbf{S}_{P_i}; \mathbf{S}_{G_i}]$. This structure makes the candidate region heterogeneous. Inherited memory must be preserved as transmitted context, so we separate it from the current agent's local prompt context, and the role of preserving recent context is now filled by the current agent's active latent reasoning states. We therefore use a four-part decomposition (Figure 1):

1. **Attention Sink ($\mathcal{S} \subset \mathcal{M}_{i-1}$):** A persistent anchor, originated by π_0 and inherited across the chain, that maintains numerical stability.
2. **Inherited Message History ($\mathcal{H} \subset \mathcal{M}_{i-1}$):** The compressed historical context transmitted by upstream agents.
3. **Current Prompt Context ($\mathcal{P} = \mathbf{S}_{P_i}$):** The latent states induced by the task-specific instructions of the current agent.
4. **Current Latent Reasoning ($\mathcal{G} = \mathbf{S}_{G_i}$):** The latent states generated by the current agent's ongoing reasoning, which form the immediate causal basis for π_i's generation.

Unlike incremental eviction during decoding, compression at the relay boundary operates on the current agent's prompt cache as a block, allowing collective compensation for the discarded values.

4.3 Eviction Baselines in LatentMAS (LMAS)

We adapt standard KV-cache eviction rules to LMAS by compressing only the current agent's local prompt states $\mathcal{P}$, while the current reasoning states $\mathcal{G}$ and the inherited memory $\mathcal{M}_{i-1}$ are forwarded unchanged. The message transmitted by agent π_i is

$$\mathcal{M}_i = [\mathcal{M}_{i-1};\ \Phi(\mathcal{P}\,;\,\mathcal{G});\ \mathcal{G}],$$

where $\Phi(\mathcal{P}\,;\,\mathcal{G})$ returns a retained (and optionally augmented) subset of the prompt states, conditioned on the reasoning states $\mathcal{G}$ that determine which prompt positions are informative. We take $\mathcal{M}_0 = \mathcal{S}$, so the shared sink instantiated by π_0 is carried forward inside $\mathcal{M}_{i-1}$ thereafter.

4.3.1 LMAS-Sink: Generation-Only Retention

LMAS-Sink instantiates the retention rule with a simple policy that preserves only the shared **Attention Sink** $\mathcal{S}$ (initialized by the first agent) and treats each local prompt $\mathcal{P}_i$ as transient context. The retention rule is $\Phi_{\text{Gen}}(\mathcal{P}_i\,;\,\mathcal{G}_i) = \emptyset$, so only $\mathcal{G}_i$ is appended to $\mathcal{M}_{i-1}$ at each round.

4.3.2 LMAS-Attn: Attention-Based Selection

LMAS-Attn retains prompt tokens that receive high attention from the current latent reasoning states and evicts the rest under a fixed budget, following importance-based retention (Zhang et al., 2023).

Attention mass from the current reasoning states. For a given layer l and a given attention head h, let $a_{t,j}^{(l,h)}$ denote the attention weight from current reasoning position $t \in \mathcal{G}_i$ to prompt position $j \in \mathcal{P}_i$. We define the attention mass assigned to prompt position j as

$$A_j^{(l,h)} = \sum_{t \in \mathcal{G}_i} a_{t,j}^{(l,h)}. \tag{1}$$

Retention rule. Let $\Omega_i \subseteq \mathcal{P}_i$ denote the eligible indices (excluding padding and, when $i = 1$, the shared sink $\mathcal{S}$). Under headwise and layerwise granularity (Figure 2), the retained set is

$$\mathcal{P}_{\text{keep},i}^{\text{head}} = \text{TopK}_B\big(\{A_j^{(l,h)}\}_{j\in\Omega_i}\big), \qquad \mathcal{P}_{\text{keep},i}^{\text{layer}} = \text{TopK}_B\Big(\big\{\textstyle\sum_{h=1}^{H} A_j^{(l,h)}\big\}_{j\in\Omega_i}\Big),$$

where B corresponds to `kv_budget` and ties are broken deterministically. Writing $\mathcal{P}_{\text{keep},i}$ for either choice, the retention rule is $\Phi_{\text{Attn}}(\mathcal{P}_i\,;\,\mathcal{G}_i) = \mathcal{P}_{\text{keep},i}$.

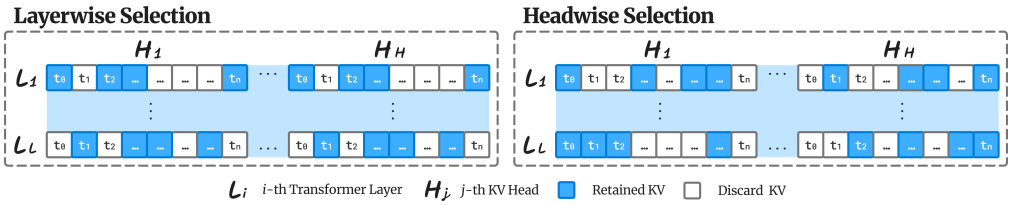


Figure 2: **Comparison of KV cache eviction granularities.** (1) **Layerwise**: shared across heads within each layer; and (2) **Headwise**: independent for each KV head and each layer. Qualitative examples of the retained prompt tokens under different granularities are shown in Appendix L.

4.4 OBF: Residual Principal-Subspace Injection for Hard Eviction

In LMAS, eviction is applied block-wise to each agent's local prompt states, so a single compression step can delete a large block of prompt KV at once. We propose **OBF**, which extracts a low-rank residual from the deleted values and injects it back into the retained ones (Figure 3). Given any base selector producing a retained prompt set $\mathcal{P}_{\text{keep},i} \subset \mathcal{P}_i$ (e.g., the LMAS-Attn rule above), let $\mathcal{P}_{\text{del},i} = \mathcal{P}_i \setminus \mathcal{P}_{\text{keep},i}$ denote the deleted positions.

Retained and deleted value states. For each layer l and KV head h, let $\mathbf{V}^{(l,h)} \in \mathbb{R}^{L_{\text{prompt}}\times d}$ denote the value states of the prompt tokens, whose t-th row is $\mathbf{V}_t^{(l,h)}$. All of the following operations are applied independently for each (l,h), so we omit these superscripts below. We treat $\mathcal{P}_{\text{keep}}$ and $\mathcal{P}_{\text{del}}$ as ordered index lists specifying the kept and deleted portions of the value matrix, and write $\mathbf{V}_{\text{keep}} = \mathbf{V}[\mathcal{P}_{\text{keep}}, :]$ and $\mathbf{V}_{\text{del}} = \mathbf{V}[\mathcal{P}_{\text{del}}, :]$.

Residual relative to the retained span. We construct an orthonormal basis for $\text{span}(\mathbf{V}_{\text{keep}})$ via a reduced QR decomposition on $\mathbf{V}_{\text{keep}}^\top$:

$$\mathbf{Q} = \text{orth}\big(\mathbf{V}_{\text{keep}}^\top\big) \in \mathbb{R}^{d\times r}, \qquad \mathbf{Q}^\top\mathbf{Q} = \mathbf{I}_r, \qquad r \le \min(B, d). \tag{2}$$

Subtracting from $\mathbf{V}_{\text{del}}$ its projection onto the span of retained values yields the residual

$$\mathbf{R} = \mathbf{V}_{\text{del}} - (\mathbf{V}_{\text{del}}\mathbf{Q})\mathbf{Q}^\top. \tag{3}$$

The residual $\mathbf{R}$ is therefore orthogonal to the span of retained values.

> **Why inject only the orthogonal component?** The retained values already span part of the deleted value content. Injecting that component would reinforce directions already represented in the cache. The projection ablation in Appendix D tests its effect on accuracy.

Injection direction. We first denoise $\mathbf{R}$ by applying Singular Value Decomposition (SVD) and retaining its top k right singular vectors as the residual principal basis $\mathbf{C} \in \mathbb{R}^{k\times d}$, where $k = \min(\texttt{pca_rank}, \text{rank}(\mathbf{R}))$. Appendix A.5 gives the ranks used in each experiment, and Appendix G examines rank sensitivity. We then summarize the deleted residual content by an

attention-weighted average: writing $\mathcal{P}_{\text{del}} = \{t_j\}_{j=1}^{N}$ for the deleted positions in the same order as the rows of $\mathbf{R}$,

$$\bar{\mathbf{r}} = \sum_{j=1}^{N} w_j \, \mathbf{R}[j,:] \in \mathbb{R}^d, \qquad w_j = \frac{A_{t_j}}{\sum_{t \in \mathcal{P}_{\text{del}}} A_t + \epsilon}, \tag{4}$$

where A_t denotes the attention mass $A_t^{(l,h)}$ from Eq. 1. Projecting $\bar{\mathbf{r}}$ onto the residual principal subspace yields the injection direction

$$\Delta = \big(\bar{\mathbf{r}}\mathbf{C}^\top\big)\mathbf{C} \in \mathbb{R}^d. \tag{5}$$

Why aggregate via PCA? Each discarded token is relatively low-ranked by the eviction selector, but the discarded set can still contain useful structure. PCA captures the leading directions of variation in the residual matrix. OBF summarizes these directions in the retained values.

Fast SVD alternative. Since OBF needs only the top-k right singular subspace of $\mathbf{R}$, a fast SVD approximation can estimate this subspace without computing the full decomposition. We use subspace iteration with products $\mathbf{R}^\top(\mathbf{RS})$, followed by a small projected eigendecomposition, without forming the full Gram matrix. Main-text accuracy results use exact SVD. Appendix H provides the fast algorithm and its analysis.

Dynamic scaling by deleted-versus-retained demand. We then scale the injection by the deleted-versus-retained demand ratio:

$$\delta = \frac{\tilde{A}_{\text{del}}}{\tilde{A}_{\text{keep}} + \epsilon} \Delta, \qquad \text{where } \tilde{A}_{\text{keep}} = \sum_{t \in \mathcal{P}_{\text{keep}}} A_t, \quad \tilde{A}_{\text{del}} = \sum_{t \in \mathcal{P}_{\text{del}}} A_t. \tag{6}$$

Why scale by demand? The PCA direction tells us *what* to inject; the demand ratio tells us *how much*. When deleted positions collectively receive little attention ($\tilde{A}_{\text{del}}$ small), OBF attenuates the injection according to this sending-agent attention signal. Strong backfill therefore requires both a coherent residual direction and high deleted demand.

Injection into retained prompt values. OBF modifies *only* the value states and leaves keys unchanged.[1] In this work, we use a uniform injection strategy: we add the same residual vector δ to all retained value states,

$$\mathbf{V}_t^* \leftarrow \mathbf{V}_t + \delta, \qquad \forall\, t \in \mathcal{P}_{\text{keep}}. \tag{7}$$

The retention rule is $\Phi_{\text{OBF}}(\mathcal{P}_i\,;\,\mathcal{G}_i) = \mathcal{P}_{\text{keep},i} \oplus \delta_i$, where $\oplus$ denotes adding δ_i to the values at retained positions while keeping keys unchanged. We ablate each of the three design choices (projection, filtering, and scaling) in Appendix D.

5 Experiments

5.1 Setup

Tasks and Datasets. Following the evaluation protocol of LatentMAS (Zou et al., 2025), we evaluate on a suite of benchmarks organized into four groups: (i) **Math**: GSM8K (Cobbe et al., 2021) and AIME 2024/2025 (Maxwell-Jia, 2024; MathArena, 2025); (ii) **Expert QA**: GPQA (Rein et al., 2023) and MedQA (Jin et al., 2021); (iii) **Commonsense QA**: ARC (Easy/Challenge) (Clark et al., 2018); (iv) **Coding**: MBPP-Plus and HumanEval-Plus (Liu et al., 2023a).

AIME24/25 were excluded from aggregate analyses after we observed substantial across-seed variability on their 30-problem test sets. Their per-task results are retained for reference.

Implementation Details. We build on the official LatentMAS implementation (Zou et al., 2025) with Qwen3-4B, Qwen3-8B and Qwen3-14B (Yang et al., 2025) as backbones, evaluating only the sequential multi-agent setting. All compared methods share the backbone, prompts, agent-chain

[1] If $\mathcal{P}_{\text{del}} = \emptyset$, $k = 0$, or $\|\mathbf{R}\|_F$ is negligible, OBF is skipped for that (i, l, h); we use $\epsilon = 10^{-12}$ throughout for numerical stability.

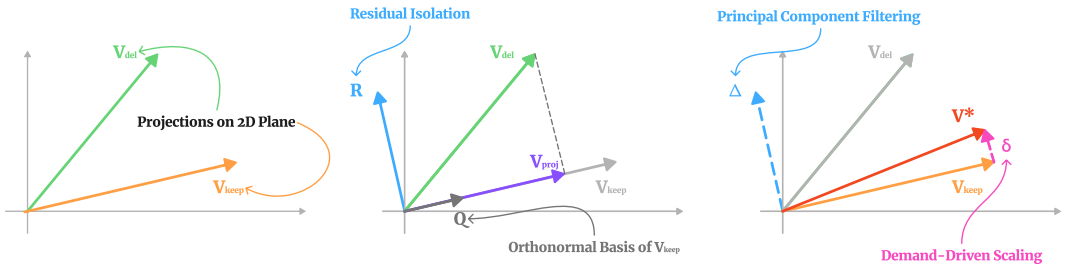


Figure 3: **Geometric illustration of OBF.** 2D sketch (not a literal high-dimensional mapping). From discarded values $\mathbf{V}_{\text{del}}$, OBF isolates the residual $\mathbf{R}$ orthogonal to the retained span $\mathbf{Q}$ (Eq. 3), keeps its principal subspace Δ (Eq. 5), scales it by the demand ratio (Eq. 6), and fuses the result δ uniformly into $\mathbf{V}_{\text{keep}}$ (Eq. 7) without changing which positions are kept.

configuration, and decoding setup, differing only in the compression operator. Appendix A gives everything needed to reproduce the runs, including the communication protocol, prompt templates, pseudocode, hyperparameters and hardware.

Compared Methods.

- **Full-KV LatentMAS (Full).** Each agent relays the complete KV cache of the full history.
- **LMAS-Attn (H).** Attention-based hard eviction inspired by importance-based retention (Zhang et al., 2023). For each (l, h), we rank prompt positions by that head's attention mass from $\mathcal{G}_i$ and keep the top-B independently, so retained positions vary across heads and layers.
- **LMAS-Attn (L).** A layerwise variant that aggregates attention across heads within each layer before selection; all heads in a layer share the retained positions.
- **LMAS-Attn + OBF (H-OBF, L-OBF).** The H and L selectors followed by OBF injection (Section 4.4) into the retained values. Rank settings are in Appendix A.5.
- **LMAS-Sink (Gen).** A sink and generation-only relay (Xiao et al., 2024) that preserves a global attention sink and forwards only each agent's newly generated KV states. LMAS-Sink has no H/L variant because it performs no position-wise selection.

Metrics and KV Budget. We report **accuracy** using the standard metric of each benchmark. For efficiency, we impose a fixed **KV token budget** B on each agent's relayed prompt states $\mathcal{P}_i$, retaining at most B positions per agent. In Table 1, L is the total uncompressed prompt length summed across the non-judger agents and averaged over samples. The reported prompt-side **compression ratio** is $\rho = (NB + s)/L$, where $N = 3$ is the number of non-judger agents and $s = 4$ is the shared sink size, retained once per rollout.

Method	Math			Expert QA		Commonsense QA		Coding	
	GSM8K	AIME24	AIME25	GPQA	MedQA	ARC-E	ARC-C	MBPP+	HumanEval+
L / ρ (%)	524 / 19.1	663 / 15.1	819 / 12.2	942 / 10.6	1012 / 9.9	496 / 20.2	524 / 19.1	778 / 12.8	828 / 12.1
Full	**88.98**	48.89	43.33	**39.26**	**61.67**	**91.98**	**87.03**	58.20	68.90
L	86.58	**53.33**	**52.22**	33.16	59.89	90.59	85.13	59.35	65.45
L-OBF	87.74	52.22	45.56	31.98	61.56	90.24	85.30	59.26	**69.11**
H	86.76	51.11	45.56	33.33	59.00	90.64	85.69	59.08	62.40
H-OBF	87.52	52.22	46.67	34.01	61.44	90.78	85.75	59.70	68.29
Gen	82.21	**53.33**	47.78	29.78	57.22	85.70	79.72	**61.73**	64.63

Table 1: **Downstream accuracy across benchmarks.** Each entry is the task's standard accuracy metric (%). Best per benchmark is in **bold**.

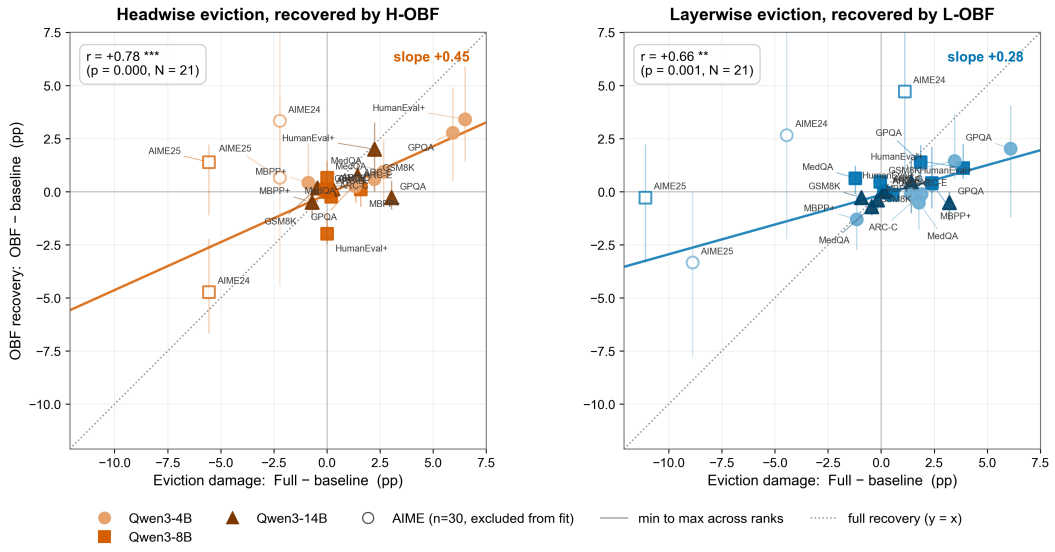


Figure 4: **Larger eviction losses are associated with larger OBF gains.** Each point is one (model, benchmark) pair, *baseline* is the eviction selector without OBF (H on the left, L on the right), and markers are averaged over the ranks swept at that scale.

5.2 Results

We report downstream accuracy (Table 1). Across benchmarks, L ranges from 496 to 1012 tokens and ρ from 9.9% to 20.2%.

Downstream accuracy. We compare each OBF variant against its eviction baseline at the same relay budget. **H-OBF improves over H on all seven non-AIME benchmarks at 4B** (Table 1), and L-OBF improves over L on four. On HumanEval+, H-OBF gains 5.89 percentage points over H, while L-OBF reaches 69.11% against Full's 68.90% with only 12.1% of the prompt KV. The gains extend beyond 4B: L-OBF improves on six of seven non-AIME tasks at 8B (Table 4); H-OBF improves on six of seven at 14B (Table 5). At 14B, H-OBF gains 3.25 points over H on HumanEval+, exceeding Full by 1.02 points (Table 5). H-OBF at 8B and L-OBF at 14B are close to their eviction baselines on average (Appendix B).

Inference efficiency. Compressed relay decodes 8–10% faster in tokens per second, because shorter relayed prompts make each step cheaper. With fast SVD, OBF takes about 1.05× Full's end-to-end time, including compression (Table 12); peak GPU memory falls by roughly 2 GB at every backbone (Table 14). All agents share one process and one GPU, so Full never pays to transmit its 290 MB. Appendix I gives the complete cost breakdown.

Recovery and eviction damage. Figure 4 plots, for each (model, benchmark) pair, what eviction removes against what OBF gives back. Each point averages OBF accuracy across all ranks swept for that model–benchmark pair, giving each pair equal weight. Regressing recovery on damage over the N=21 non-AIME cells gives a slope of +0.45 with $r = +0.78$ ($p < 10^{-4}$) for the headwise selector and +0.28 with $r = +0.66$ (p=0.001) for the layerwise one. The fitted intercepts are not significantly different from zero. Appendix C gives the full definition and a budget-sweep check.

5.3 Additional Analyses

Component ablations are reported in Appendix D, compression comparisons in Appendix E, budget sweeps in Appendix F, and rank sweeps in Appendix G. Additional analyses cover fast SVD (Appendix H), downstream generation cost (Appendix J), and internal diagnostics (Appendix K).

Method	Compression	MedQA	GSM8K	ARC-C	MBPP+	HEval+	Mean	Worst
Full (uncompressed)	1.0×	–	–	–	–	–	–	–
Layerwise	4.6×	−1.78	−2.40	−1.91	+1.15	−3.46	−1.68	−3.46
L-OBF	4.6×	−0.11	−1.24	−1.73	+1.06	**+0.20**	**−0.36**	**−1.73**
CaM-inspired merging	4.6×	−2.89	−1.72	−2.10	**+3.35**	−1.42	−0.96	−2.89
Full + int8	1.9×	**+0.22**	−0.99	**+0.80**	−1.85	−2.44	−0.85	−2.44
Full + int4	3.7×	−0.33	**−0.48**	+0.09	+0.18	−6.71	−1.45	−6.71
L-OBF + int8	**9.0**×	−2.89	−1.74	−1.54	+2.38	−3.05	−1.37	−3.05

Table 2: **KV compression comparisons on Qwen3-4B.** Accuracy changes (pp) relative to Full. Compression denotes relay-size reduction; Mean and Worst summarize the five tasks. Best per column is in **bold**. Setup and analysis are in Appendix E.

6 Discussion

Preserving the entire cache is not always necessary for effective downstream continuation. At 4B, compressed variants perform better on coding tasks, while Full remains stronger on reasoning and QA benchmarks (Table 1). These differences suggest that compression can remove useful context as well as content that does not help the receiving agent. OBF addresses the former without restoring the full cache. Across three model scales at a fixed relay budget, larger eviction losses are associated with larger rank-averaged OBF gains (Appendix C).

Recovering discarded content does not mean adding it all back. Compensation should account for what the retained cache already represents. In our internal diagnostics, roughly half of the discarded value energy lies outside the retained span (Appendix K.3). Removing the projection lowers average accuracy by 4.85 percentage points across five benchmarks (Appendix D). These results suggest that compensation should separate missing content from information already retained.

At the relay boundary, OBF processes the discarded block collectively to preserve shared background context. At approximately $4.6\times$ relay reduction, L-OBF achieves 0.60 percentage points higher average accuracy than CaM-inspired cache merging (Zhang et al., 2024) and 1.32 points higher than layerwise eviction across five benchmarks at 4B (Table 2). L-OBF with int8 quantization achieves $9.0\times$ relay reduction, with an average accuracy loss of 1.37 points relative to Full (Appendix E). These results suggest that OBF's collective compensation better suits inter-agent KV-cache communication.

To reduce compression overhead, we introduce a fast SVD approximation that accelerates compression by $7.6\times$ for L-OBF and $5.4\times$ for H-OBF (Appendix I). When exact and fast SVD receive the same input caches in an L-OBF numerical check, the median layerwise relative cache error is 7.06×10^{-5}, and the maximum whole-cache error is 1.91×10^{-4}. Appendix H bounds cache perturbations from subspace approximation and reports these numerical checks and full-rollout accuracy comparisons.

Discarding a cache position need not mean discarding all of its information. At the relay boundary, one-shot eviction can be paired with collective compensation to preserve complementary information within the retained values. KV-cache compression can therefore optimize both which positions to retain and what information they carry.

7 Limitations

Our results use the Qwen3 dense grouped-query attention family (4B, 8B, and 14B) within the sequential four-agent chain of Zou et al. (2025). Extension to other model families (e.g., Llama), architectures (MoE, non-GQA), and multi-agent topologies remains to be verified. Our evaluation covers reasoning, QA, and code benchmarks. Performance on real-world multi-agent applications such as long-form collaboration, tool use, or full agentic workflows remains untested.

AI USE STATEMENT

Generative AI tools assisted with implementing and checking code, checking mathematical arguments, interpreting experimental results, identifying literature, formatting references, and revising the manuscript. The authors reviewed the AI-assisted work and take responsibility for the final text, claims, references, and artifacts.

Concretely, generative AI wrote parts of the compression operators, the OBF implementation, and the batched inference and evaluation harness, as well as the plotting and table-generation code. It also assisted with manuscript revisions. Generative AI assisted with deriving the error bounds comparing fast and exact SVD in Appendix H. AI-assisted code was read and tested by the authors, including a degenerate-case check in which compression is disabled and the relay path must reproduce uncompressed full relay. All reported measurements come from the authors' experimental runs; tables and figures are based on those outputs. Reference entries assembled with AI assistance were checked against the original publications.

ETHICS STATEMENT

The work studies inference efficiency and involves no human participants and no personal data. All experiments use publicly released models (Yang et al., 2025) and public benchmarks within their intended evaluation purpose, and we release no new data set. The method operates on cached intermediate states and does not modify model weights. Its effects on safety behavior have not been evaluated. A relayed KV cache encodes the content of an agent's prompt and warrants the same confidentiality as the prompt itself.

REPRODUCIBILITY STATEMENT

Our code will be released upon acceptance, covering the method, all compared baselines, the evaluation harness, and the scripts that produce the reported tables. Raw per-run outputs are not distributed because of their size; they are regenerated by the provided scripts under the logged seeds and configurations. Appendices A.1 and A.2 give the four-agent sequential setting and the verbatim prompt templates. Appendix A.5 gives the full experimental configuration: backbones, decoding precision and sampling parameters, fixed hyperparameters, per-task generation budgets, hardware, and the per-backbone seed sets behind the three-seed means. Appendix A.6 states the compression and OBF procedures as pseudocode, and Appendix A.7 documents the four adjustments required to run compressed relay under batched inference, which determine whether a reimplementation reproduces our numbers.

REFERENCES

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. Chateval: Towards better LLM-based evaluators through multi-agent debate, 2023. URL https://arxiv.org/abs/2308.07201.

Weize Chen, Jiarui Yuan, Chen Qian, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Optima: Optimizing effectiveness and efficiency for LLM-based multi-agent system, 2025. URL https://arxiv.org/abs/2410.08115.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 Reasoning Challenge. *arXiv preprint arXiv:1803.05457*, 2018. doi: 10.48550/arXiv.1803.05457. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. doi: 10.48550/arXiv.2110.14168. URL https://arxiv.org/abs/2110.14168.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2023. URL https://arxiv.org/abs/2305.14325.

Tianyu Fu, Zihan Min, Hanling Zhang, Jichao Yan, Guohao Dai, Wanli Ouyang, and Yu Wang. Cache-to-cache: Direct semantic communication between large language models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 43130–43158, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/474ada926b331d78f06d95e8913111cc-Paper-Conference.pdf.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space, 2025. URL https://arxiv.org/abs/2412.06769.

Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for a multi-agent collaborative framework, 2024. URL https://arxiv.org/abs/2308.00352.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length LLM inference with KV cache quantization, 2025. URL https://arxiv.org/abs/2401.18079.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hyeonho Fang, and Peter Szolovits. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421, 2021. doi: 10.3390/app11146421. URL https://www.mdpi.com/2076-3417/11/14/6421.

Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society, 2023. URL https://arxiv.org/abs/2303.17760.

Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate, 2024. URL https://arxiv.org/abs/2305.19118.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation, 2023a. URL https://arxiv.org/abs/2305.01210.

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, 2023b. URL https://arxiv.org/abs/2305.17118.

MathArena. Aime 2025. Hugging Face Datasets, 2025. URL https://huggingface.co/datasets/MathArena/aime_2025. Accessed 2026-02-18.

Maxwell-Jia. Aime 2024. Hugging Face Datasets, 2024. URL https://huggingface.co/datasets/Maxwell-Jia/AIME_2024. Accessed 2026-02-18.

Chau Pham, Boyi Liu, Yingxiang Yang, Zhengyu Chen, Tianyi Liu, Jianbo Yuan, Bryan A. Plummer, Zhaoran Wang, and Hongxia Yang. Let models speak ciphers: Multiagent debate through embeddings, 2024. URL https://arxiv.org/abs/2310.06272.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023. doi: 10.48550/arXiv.2311.12022. URL https://arxiv.org/abs/2311.12022.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient KV cache compression through retrieval heads, 2024. URL https://arxiv.org/abs/2407.15891.

Yichen Tang, Weihang Su, Yujia Zhou, Yiqun Liu, Min Zhang, Shaoping Ma, and Qingyao Ai. Augmenting multi-agent communication with state delta trajectory, 2025. URL https://arxiv.org/abs/2506.19209.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation, 2023. URL https://arxiv.org/abs/2308.08155.

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.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Jian Yuan, Ziwei He, Haoli Bai, Jingwen Leng, and Bo Jiang. Weightedkv: Attention scores weighted key-value cache merging for large language models, 2025. URL https://arxiv.org/abs/2503.01330.

Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji. CaM: Cache merging for memory-efficient LLMs inference. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 58840–58850. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/zhang24n.html.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. H_2o: Heavy-hitter oracle for efficient generative inference of large language models, 2023. URL https://arxiv.org/abs/2306.14048.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. Least-to-most prompting enables complex reasoning in large language models, 2023. URL https://arxiv.org/abs/2205.10625.

Jiaru Zou, Xiyuan Yang, Ruizhong Qiu, Gaotang Li, Katherine Tieu, Pan Lu, Ke Shen, Hanghang Tong, Yejin Choi, Jingrui He, James Zou, Mengdi Wang, and Ling Yang. Latent collaboration in multi-agent systems, 2025. URL https://arxiv.org/abs/2511.20639.

A EXPERIMENT SETTINGS

A.1 SEQUENTIAL LATENTMAS SETTING

We follow the sequential LatentMAS (LMAS) setting of Zou et al. (2025) without modification. Four agents are arranged in a pipeline: Planner, Critic, Refiner, and Judger. Each non-Judger agent performs latent reasoning under its role-specific prompt and relays its full KV cache to the next agent as `past_key_values`; the downstream agent conditions on its text input (role prompt and question) and the transferred KV cache. Only the Judger decodes a textual response.

A.2 PROMPT TEMPLATES

We reuse the LMAS prompt templates verbatim. Let $\mathcal{Q}$ denote the input question shared across agents.

System Prompt. All agents share the same system prompt:

```
You are Qwen, created by Alibaba Cloud.  You are a
helpful assistant.
```

Planner.

```
You are a Planner Agent.  Given an input question,
design a clear, step-by-step plan for how to solve the
question.
Question:  Q
Your outlined plan should be concise with a few
bulletpoints for each step.  Do not produce the final
answer.
Now output your plan to solve the question below:
```

Critic.

```
Question:  Q
You are a Critic Agent to evaluate the correctness
of the input plan for the given question and provide
helpful feedback for improving the plan.
The plan information is provided in latent KV
representation format.  Review the plan and question
and output:
(1) original plan contents
(2) constructive feedback on the original plan.
Format your response as follows:
Original Plan:  [Copy the provided Planner Agent's
plan here]
Feedback:  [Your detailed feedback to improve the plan
here]
Now, output your response below:
```

Refiner.

```
Question:  Q
You are a Refiner Agent to provide a refined
step-by-step plan for solving the given question.
You are provided with:
(1) latent-format information:  a previous plan with
feedback
```

```
(2) text-format information: the input question you
need to solve.
Based on the input, write a refined and improved plan
to solve the question. Make sure your output plan is
correct and concise.
Now, output your refined plan below:
```

Judger. For all tasks, the Judger is told that latent information is available as reference and may contain irrelevant content. The task-specific output constraint differs by benchmark family.

(1) Math / numeric-answer tasks (`gsm8k`, `aime2024`, `aime2025`):

```
Target Question: Q
You are a helpful assistant. You are provided with
latent information for reference and a target question
to solve.
The latent information might contain irrelevant
contents. Ignore it if it is not helpful for solving
the target question.
You must reason step-by-step to solve the provided
Target Question without outputting other irrelevant
information.
Now, reason step by step and output the final answer
inside \boxed{YOUR_FINAL_ANSWER}.
```

(2) Multiple-choice tasks (`arc_easy`, `arc_challenge`, `gpqa`, `medqa`):

```
Target Question: Q
You are a helpful assistant. You are provided with
latent information for reference and a target question
to solve.
The latent information might contain irrelevant
contents. Ignore it if it is not helpful for solving
the target question.
You must reason step-by-step to solve the provided
Target Question without outputting other irrelevant
information.
Your final answer must be selected from A,B,C,D. For
example \boxed{A}. Do not add any other contents
inside the box.
Now, reason step by step and output the final answer
inside \boxed{YOUR_FINAL_ANSWER}.
```

(3) Code-generation tasks (`mbppplus`, `humanevalplus`):

```
Target Question: Q
You are a helpful assistant. You are provided with
latent information for reference and a target question
to solve.
The latent information might contain irrelevant
contents. Ignore it if it is not helpful for solving
the target question.
You must reason step-by-step to solve the provided
Target Question without outputting other irrelevant
information.
You must put all python code as self-contained Python
function in markdown code blocks. Do not add any
other contents inside the markdown code block.
```

```
Now, reason step by step and output the final answer
inside a Python markdown code block.
```

A.3 Hardware Environment

Unless otherwise specified, all experiments run on NVIDIA A100-PCIE-40GB nodes.

A.4 Maximum Generated Tokens and Batch Sizes

Table 3 lists the maximum generated tokens and batch sizes used throughout the paper, following the LMAS defaults.

Datasets	Max generated tokens	Batch size
GSM8K, ARC-E, ARC-C	2048	4
MBPP+, HumanEval+, MedQA	4096	2
GPQA	8192	2
AIME2024, AIME2025	20000	2

Table 3: **Task-specific token generation budgets and batch sizes.**

A.5 Shared Experimental Configuration

All experiments in the paper share the sequential LMAS setting (Appendix A.1), prompt templates (Appendix A.2), hardware (Appendix A.3), and per-task decoding budgets (Appendix A.4). We use the Qwen3 family (Yang et al., 2025) as the backbone throughout; the specific model is stated per-table. Decoding uses `bfloat16` precision with stochastic sampling (temperature 0.6, top-p 0.95).

Hyperparameters. Unless otherwise specified, experiments use `kv_budget` $= 32$ prompt tokens retained per non-judger agent per round, `sink_size` $= 4$ tokens retained once across the rollout, and `latent_steps` $= 40$ latent iterations per round. In the primary accuracy comparisons, L-OBF uses `pca_rank` $= 4$ at Qwen3-4B and Qwen3-8B and rank 2 at Qwen3-14B; H-OBF uses rank 2 at all three scales. Rank comparisons are reported in Appendix G. Explicit rank sweeps use the settings labelled in their tables and figures.

Evaluation samples. Each run evaluates 1,319 GSM8K, 30 AIME2024, 30 AIME2025, 197 GPQA, 300 MedQA, 2,376 ARC-E, 1,172 ARC-C, 378 MBPP+, and 164 HumanEval+ problems. GPQA uses `fingertap/GPQA-Diamond`; its loader skips questions longer than 4,000 characters. MedQA uses the 300-item AgentFlow-format `medqa.json`, with the same query-length filter. The supplementary README identifies the exact MedQA file by its SHA-256 checksum.

Seeding. Accuracy comparisons are three-seed means unless otherwise stated. Internal diagnostics and token-selection visualizations use individual runs, as specified in their sections. Seed sets are chosen per backbone:

- **Qwen3-4B**: seeds $\{4, 44, 444\}$.
- **Qwen3-8B**: seeds $\{8, 88, 888\}$.
- **Qwen3-14B**: seeds $\{14, 1414, 141414\}$.

AIME reporting convention. AIME24 and AIME25 each contain only 30 problems, so a single correct answer changes per-seed accuracy by 3.33 percentage points; our runs also show substantial across-seed variation. We retain their per-task results but exclude both datasets from aggregate accuracy comparisons, win counts, rank scores, and the cross-benchmark recovery fit. The same exclusion applies to every method and rank. Accuracy averages therefore use the seven non-AIME benchmarks, or the explicitly named subset in each ablation or comparison.

A.6 ALGORITHM

We organize the algorithm into one unified relay procedure (Algorithm 1) that treats the compression step as a black box, and two sub-algorithms (Algorithms 2 and 3) that instantiate the compression operator Φ. Setting $\Phi = \Phi_{\text{Gen}}$ is trivial ($\Phi_{\text{Gen}}(\mathcal{P}_i) = \emptyset$) and omitted.

Algorithm 1 Unified relay at agent π_i.

Require: Prompt P_i; previous message $\mathcal{M}_{i-1}$; latent-step count p; compression operator Φ.
Ensure: Message $\mathcal{M}_i$.

$(\mathbf{K}, \mathbf{V}, \mathcal{G}_i, a) \leftarrow \textsc{LatentAutoregression}\big(\text{past_kv}{=}\mathcal{M}_{i-1},\ \text{prompt}{=}P_i,\ \text{steps}{=}p\big)$
$\mathcal{P}_i^\star \leftarrow \Phi(\mathbf{K}, \mathbf{V}, a)$ ▷ see Alg. 2 / Alg. 3
$\mathcal{M}_i \leftarrow [\mathcal{M}_{i-1}\,;\ \mathcal{P}_i^\star\,;\ \mathcal{G}_i]$
return $\mathcal{M}_i$

The following two sub-algorithms use the notation of Section 4: $\mathcal{P}_{\text{keep}}/\mathcal{P}_{\text{del}}$ are retained / deleted prompt index sets, and $\mathbf{V}_{\text{keep}}/\mathbf{V}_{\text{del}}$ the corresponding value slices.

Algorithm 2 Φ_{Attn}: attention-based selector (Section 4.3.2).

Require: $\mathbf{K}, \mathbf{V}$; attention tensor a; budget B; granularity $\in \{\text{H}, \text{L}\}$.

$A_j^{(l,h)} \leftarrow \sum_{t=1}^{p} a_{t,j}^{(l,h)}$ for $j \in \mathcal{P}_i$
if granularity $=$ H **then**
  $\mathcal{P}_{\text{keep}}^{(l,h)} \leftarrow \text{TopK}_B\big(\{A_j^{(l,h)}\}_{j\in\mathcal{P}_i}\big)$
else if granularity $=$ L **then**
  $\mathcal{P}_{\text{keep}}^{(l,h)} \leftarrow \text{TopK}_B\big(\{\sum_h A_j^{(l,h)}\}_{j\in\mathcal{P}_i}\big)$
end if
$\mathbf{K}_{\text{keep}}, \mathbf{V}_{\text{keep}} \leftarrow \mathbf{K}[\mathcal{P}_{\text{keep}}^{(l,h)}, :],\ \mathbf{V}[\mathcal{P}_{\text{keep}}^{(l,h)}, :]$
return $(\mathbf{K}_{\text{keep}}, \mathbf{V}_{\text{keep}}, \mathcal{P}_{\text{keep}}^{(l,h)})$

Algorithm 3 Φ_{OBF}: Orthogonal BackFill (Section 4.4).

Require: $\mathbf{K}, \mathbf{V}$; attention tensor a; budget B; granularity $\in \{\text{H}, \text{L}\}$; PCA rank k.

$(\mathbf{K}_{\text{keep}}, \mathbf{V}_{\text{keep}}, \mathcal{P}_{\text{keep}}^{(l,h)}) \leftarrow \Phi_{\text{Attn}}(\mathbf{K}, \mathbf{V}, a, B, \text{granularity})$ ▷ Alg. 2
for each (l, h) **do**
  $\mathcal{P}_{\text{del}}^{(l,h)} \leftarrow \mathcal{P}_i \setminus \mathcal{P}_{\text{keep}}^{(l,h)}$; $\mathbf{V}_{\text{del}} \leftarrow \mathbf{V}[\mathcal{P}_{\text{del}}^{(l,h)}, :]$
  if $\mathcal{P}_{\text{del}}^{(l,h)} = \emptyset$ **then continue**
  end if
  $\tilde{A}_{\text{keep}} \leftarrow \sum_{j\in\mathcal{P}_{\text{keep}}^{(l,h)}} A_j^{(l,h)}$; $\tilde{A}_{\text{del}} \leftarrow \sum_{j\in\mathcal{P}_{\text{del}}^{(l,h)}} A_j^{(l,h)}$
  $\mathbf{Q} \leftarrow \text{QR}(\mathbf{V}_{\text{keep}}^\top)$; $\mathbf{R} \leftarrow \mathbf{V}_{\text{del}} - \mathbf{V}_{\text{del}}\mathbf{Q}\mathbf{Q}^\top$
  $\mathbf{C} \leftarrow \text{SVD}_k(\mathbf{R})$
  $\tilde{w}_t \leftarrow A_t^{(l,h)}/(\tilde{A}_{\text{del}} + \epsilon)$ for $t \in \mathcal{P}_{\text{del}}^{(l,h)}$; $\bar{\mathbf{r}} \leftarrow \sum_t \tilde{w}_t\, \mathbf{R}_t$
  $\delta \leftarrow (\bar{\mathbf{r}}\mathbf{C}^\top)\mathbf{C}$; $\alpha \leftarrow \tilde{A}_{\text{del}}/(\tilde{A}_{\text{keep}} + \epsilon)$
  $\mathbf{V}_{\text{keep}} \leftarrow \mathbf{V}_{\text{keep}} + \alpha\, \mathbf{1}\delta$ ▷ $\mathbf{1} \in \mathbb{R}^{B\times 1}$, broadcast $\alpha\delta$ to every retained row
end for
return $(\mathbf{K}_{\text{keep}}, \mathbf{V}_{\text{keep}})$

A.7 IMPLEMENTATION DETAILS

Our implementation builds on the official LatentMAS codebase (Zou et al., 2025). Supporting compressed relay under batched HuggingFace inference requires four adjustments to the inference path.

Right padding. We use right padding. Our compressors protect sink tokens as the first s indices of the KV cache (following StreamingLLM (Xiao et al., 2024)), so position 0 must be a real token for every sample; left padding would place a pad KV in that slot for short samples. Because the last position of a padded row is now a pad token, we gather the hand-off hidden state at each sample's last valid index using the attention mask, not `[:, -1, :]`.

Cross-agent attention mask. At each hand-off, the inherited KV cache $\mathcal{M}_{i-1}$ is concatenated with the freshly encoded prompt KV $\mathcal{P}_i$, and an attention mask covering both ranges is fed to the model. We propagate the real attention mask while it aligns with the cache, preventing attention to pad-origin KV slots. When compression changes the cache length, the implementation uses an all-ones mask for the compressed past at the next hand-off. This is valid for the evaluated compression methods because they physically remove padded prompt positions: selection is restricted to valid positions in each sample, the shared sink is taken from positions valid for every sample, and the latent states are real. Subsequent relays preserve only this compressed history plus newly selected valid prompt states and latent states, so padding does not enter the compressed past.

Per-sample position tracking. Under right padding and compression, samples in a batch accumulate different numbers of real KV entries across agents, latent steps, and (per layer and head) evictions. HuggingFace's single `cache_position` shared across the batch is only correct when all samples are at the same next position. We therefore maintain a per-sample position cursor $\mathbf{c} \in \mathbb{Z}^{N_b}$ and pass explicit `position_ids` of shape (N_b, L) at every forward pass. For the prompt of each agent, real tokens receive consecutive offsets $\mathbf{c}, \mathbf{c}{+}1, \ldots$ using the cumulative sum of the attention mask, while pad tokens continue the same schedule so no two KV entries share a position id. For the t-th latent step and for text generation, sample b uses position $\mathbf{c}_b + t$. After each stage, $\mathbf{c}$ advances by the number of valid slots actually appended to each sample's cache, using the compressor-reported retained lengths where applicable.

Manual decoding rollout. `model.generate()` exposes no API for injecting per-sample `position_ids` at every decoding step, so both the latent-step loop and text generation are manual decoding loops. At each step we set position_ids $= \mathbf{c} + t$ per sample, extend the attention mask by one, run a single-token forward with explicit `past_key_values`, and sample the next token using temperature 0.6 and top-p 0.95. Per-sample EOS and max-new-tokens conditions are tracked independently, so finished sequences stop advancing $\mathbf{c}$ while others continue.

B SCALING BEHAVIOR AT LARGER BACKBONES

All runs follow Appendix A.5, with the backbone and rank settings specified there for Qwen3-8B and Qwen3-14B.

B.1 QWEN3-8B

Method	Math			Expert QA		Commonsense QA		Coding	
	GSM8K	AIME24	AIME25	GPQA	MedQA	ARC-E	ARC-C	MBPP+	HumanEval+
Full	91.96	66.67	47.78	**52.12**	74.11	**98.39**	**95.02**	**75.57**	**83.33**
L	92.01	65.56	**58.89**	50.25	<u>75.33</u>	98.26	94.48	73.19	79.47
L-OBF	**92.70**	<u>72.22</u>	**58.89**	50.76	**75.56**	98.32	94.43	73.72	80.08
H	92.01	<u>72.22</u>	53.33	50.76	74.11	<u>98.34</u>	<u>94.82</u>	73.99	**83.33**
H-OBF	<u>92.12</u>	65.56	<u>55.56</u>	<u>51.61</u>	74.78	98.33	94.62	<u>74.34</u>	<u>81.10</u>
Gen	89.82	**73.33**	**58.89**	46.87	74.67	97.77	93.37	72.66	79.67

Table 4: **Qwen3-8B accuracy across benchmarks** (%, three-seed mean on A100; setup in Appendix A.5). Best per benchmark is in **bold**, runner-up <u>underlined</u>.

Reading. At 8B, L-OBF improves over L on six of the seven non-AIME tasks, while H-OBF's average change from H is close to zero. The headwise selector is already close to Full at this scale; layerwise eviction leaves a larger average gap (Table 4).

B.2 QWEN3-14B

Method	Math	Expert QA		Commonsense QA		Coding	
	GSM8K	GPQA	MedQA	ARC-E	ARC-C	MBPP+	HumanEval+
Full	91.89	**59.90**	80.67	**98.67**	<u>95.79</u>	77.43	86.99
L	**92.82**	56.68	**81.11**	98.54	95.59	77.60	85.57
L-OBF	92.47	55.33	80.44	98.60	**95.90**	<u>77.78</u>	<u>87.20</u>
H	92.34	56.85	79.22	98.53	95.51	**78.13**	84.76
H-OBF	<u>92.67</u>	<u>57.36</u>	79.89	<u>98.63</u>	95.62	76.90	**88.01**
Gen	91.51	52.96	<u>80.78</u>	98.12	94.14	75.57	83.74

Table 5: **Qwen3-14B accuracy across benchmarks** (%, three-seed mean on H200; setup in Appendix A.5). AIME24 and AIME25 are omitted here, since their 30-problem sets give per-seed swings too large to read at this scale (Appendix A.5). Best per benchmark is in **bold**, runner-up <u>underlined</u>.

Reading. At 14B, both selectors are nearly lossless on most benchmarks. L-OBF is close to L on average, improving on four of seven tasks; H-OBF remains close to H except on HumanEval+, where it recovers more than the eviction gap and finishes above Full.

C THE EFFECTIVENESS OF OBF

We test whether OBF restores accuracy in proportion to what eviction discards. For each (model, benchmark) pair and each selector, define

$$\text{damage} \;=\; \text{Acc}_{\text{Full}} - \text{Acc}_{\text{baseline}}, \qquad \text{recovery} \;=\; \frac{1}{|R|}\sum_{r\in R} \text{Acc}_{\text{OBF}}(r) - \text{Acc}_{\text{baseline}},$$

where R is the `pca_rank` set swept at that scale ($\{2, 4, 8, 16, 32\}$ at 4B, $\{2, 4, 8, 32\}$ at 8B, $\{2, 4, 8\}$ at 14B), baseline $\in$ {H, L}, and OBF $\in$ {H-OBF, L-OBF}. Recovery averages over the whole sweep without rank selection for this analysis. Data are from Qwen3-4B (Table 1), Qwen3-8B (Table 4) and Qwen3-14B (Table 5); AIME24 and AIME25 (hollow markers in Figure 4) are excluded from the fit following Appendix A.5, leaving $N = 21$ non-AIME cells per selector.

Both axes subtract the same eviction baseline, so their correlation alone does not establish proportional recovery. OLS regression of recovery on damage yields:

Selector	Slope	Intercept	Pearson r	p-value
Headwise (H-OBF vs. H)	$+0.45$	≈ 0	$+0.78$	$< 10^{-4}$
Layerwise (L-OBF vs. L)	$+0.28$	≈ 0	$+0.66$	0.001

Both slopes are positive and significant. The fitted intercepts are not significantly different from zero. Headwise OBF recovers roughly half of what headwise eviction removes; layerwise OBF recovers roughly a third.

C.1 BUDGET-SWEEP CHECK

The fit above varies the eviction gap by changing the benchmark. The relay budget varies it directly, since a smaller `kv_budget` discards more of the prompt and widens the gap on a fixed benchmark. Appendix F sweeps $B \in \{4, 8, 16, 32, 64\}$ on four benchmarks at three seeds, so it tests the relationship on runs the fit above never saw. We refit at each budget separately across the four benchmarks.

`kv_budget`	4	8	16	32	64
Pearson r	-0.07	$+0.21$	$\mathbf{+0.92}$	$\mathbf{+0.91}$	-0.94
Slope	-0.16	$+0.32$	$+1.15$	$+0.84$	-1.57

The relationship reproduces at B=16 and B=32, and pooling the three larger budgets (twelve cells) gives $r = +0.79$, slope $+0.80$. The two smallest budgets do not show this pattern. At B=4, H-OBF is below plain eviction on all four benchmarks. At B=8, recovery is mixed: it is positive on GSM8K and MedQA but negative on ARC-C and HumanEval+ (Appendix F). The per-budget fit at B=64 is unreliable on its own, because its four gaps span only 0.9 to 2.3 pp against 1.9 to 6.5 pp at B=32, and one benchmark sets the sign. The positive association appears at intermediate budgets but does not extend to every budget.

D Ablation Analysis of OBF Design Choices

We ablate OBF's design choices in two groups: *component ablations* remove or simplify one internal step of full L-OBF, and *alternative designs* replace one step with a different implementation of the same intent. All runs follow Appendix A.5; AIME24 and AIME25 are excluded following the convention there.

D.1 Component Ablations

- **No Projection**: skip the residual extraction in Section 4.4 that removes from $\mathbf{V}_{\text{del}}$ the component representable in the retained-value span.
- **Max-P**: skip the low-rank principal subspace extraction in Section 4.4 and use all components of the residual.
- **No Scaling**: remove the demand-aware dynamic scaling in Section 4.4.
- **Naive Aggregation**: form the backfill vector by directly averaging the deleted value states, bypassing all three steps.

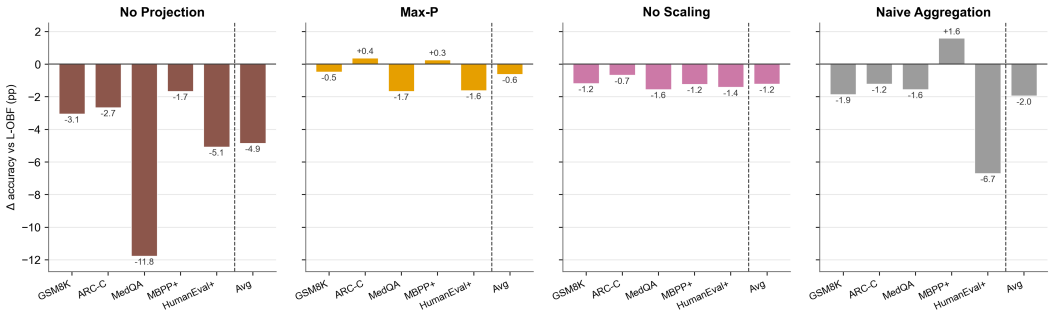


Figure 5: **Component ablations of full L-OBF.** Bars show Δ accuracy (pp) versus the L-OBF reference; the L-OBF reference itself is at zero and omitted. Negative bars mean the variant loses accuracy.

Variant	Avg Δ vs L-OBF (pp)
No Projection	−4.85
Naive Aggregation	−1.95
No Scaling	−1.22
Max-P	−0.63

Table 6: **Component ablations: average Δ accuracy vs L-OBF** across the five benchmarks. Ordered by average degradation.

Analysis. Removing projection produces the largest average drop, and removing demand scaling also lowers accuracy on all five tasks. Naive Aggregation performs worse on average than projection plus scaling. Max-P retains all right singular vectors returned by the residual SVD, without truncation; its smaller average loss indicates that truncation helps less than projection or scaling in this comparison.

D.2 Alternative Designs

- **Attn-inject**: replace the uniform injection $\mathbf{V}_t^\star \leftarrow \mathbf{V}_t + \delta$ with an attention-weighted injection

$$\mathbf{V}_t^\star \leftarrow \mathbf{V}_t + B\,\tilde{w}_t^{\text{keep}}\,\delta, \qquad \tilde{w}_t^{\text{keep}} = \frac{A_t^{(l,h)}}{\sum_{s\in\mathcal{P}_{\text{keep}}} A_s^{(l,h)} + \epsilon},$$

so each retained position receives a share of the backfill proportional to its attention mass during the sending agent's latent reasoning (rather than an equal share).

- **Per-token**: route each deleted residual to retained positions using value-space affinities. With $a_{jt} = \mathrm{softmax}_t(\mathbf{V}_{\mathrm{del}}[j,:]\mathbf{V}_{\mathrm{keep}}[t,:]^\top/\sqrt{d})$ and normalized deleted attention weights w_j, inject

$$\mathbf{V}_t^\star \leftarrow \mathbf{V}_t + \alpha \sum_j w_j a_{jt} (\mathbf{R}_j \mathbf{C}^\top)\mathbf{C}.$$

  Here $\alpha = \tilde{A}_{\mathrm{del}}/(\tilde{A}_{\mathrm{keep}} + \epsilon)$ is the demand scale and $\mathbf{C}$ is the residual basis in Eq. 5. The temporary correction has one vector per retained slot, but the relayed cache shape is unchanged. Since $\sum_t a_{jt} = 1$, the corrections sum to $\alpha\Delta$, versus $B\alpha\Delta$ for uniform injection. This variant changes both allocation and total injection magnitude.

- **EVR-adaptive** (τ=0.35): replace the fixed rank $r = 4$ with an adaptive rank

$$k^\star = \min\left(32,\ \min\left\{k : \sum_{i=1}^{k} \sigma_i^2 \Big/ \sum_i \sigma_i^2 \geq \tau\right\}\right),$$

  where σ_i are the singular values of the residual matrix $\mathbf{R}$. The implementation selects the rank separately for each sample and head, capped at 32. The argument `pca_rank` $= 35$ encodes the 35% energy threshold, not a rank of 35.

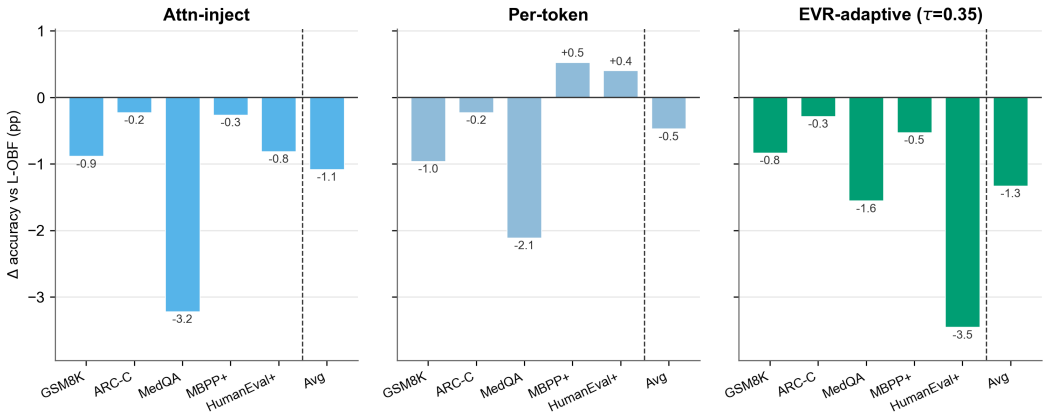


Figure 6: **Alternative-design ablations of full L-OBF.** Bars show Δ accuracy (pp) versus the L-OBF reference; the L-OBF reference itself is at zero and omitted. Negative bars mean the variant loses accuracy.

Variant	Avg Δ vs L-OBF (pp)
EVR-adaptive (τ=0.35)	−1.33
Attn-inject	−1.08
Per-token	−0.47

Table 7: **Alternative designs: average Δ accuracy vs L-OBF** across the five benchmarks. Ordered by average degradation.

Summary. No alternative improves over L-OBF on average across the five tasks at the same relay budget. Per-token injection comes closest. These comparisons favor the default design on average, but do not isolate an allocation effect from the change in total injection magnitude.

E Comparison with External KV Compression Baselines

We compare L-OBF against two other KV compression families on Qwen3-4B: a cache-merging baseline in the spirit of CaM (Zhang et al., 2024), which merges to-be-evicted KV entries into retained ones instead of dropping them, and post-hoc int8/int4 quantization in the spirit of KVQuant (Hooper et al., 2025), applied either to the full KV cache or on top of L-OBF. Neither is the exact code from the cited work; we describe the concrete cache-merging implementation below. All runs follow Appendix A.5. The comparison covers five benchmarks: GSM8K, MedQA, ARC-C, MBPP+, and HumanEval+. ARC-E is omitted because it overlaps with ARC-C. GPQA is omitted from this comparison because its long reasoning outputs make evaluation costly. AIME24/25 are excluded following Appendix A.5.

Cache-merging implementation. At every relay boundary and every (l, h), we do three things:

(i) Select. The same attention selector as the eviction baselines picks the retained set of size B:

$$\mathcal{P}_{\text{keep}} \;=\; \mathrm{TopK}_B\big(\{A_t^{(l,h)}\}_{t\in\mathcal{P}}\big), \qquad \mathcal{P}_{\text{del}} \;=\; \mathcal{P}\setminus\mathcal{P}_{\text{keep}}.$$

(ii) Assign. Each deleted position is routed to its nearest retained slot by key cosine similarity:

$$\pi(j) \;=\; \arg\max_{t\in\mathcal{P}_{\text{keep}}} \frac{\langle \mathbf{K}_j, \mathbf{K}_t\rangle}{\|\mathbf{K}_j\|\,\|\mathbf{K}_t\|}, \qquad j\in\mathcal{P}_{\text{del}}.$$

(iii) Merge. The routed value states are added into the retained slots, weighted by attention mass:

$$\mathbf{V}_t^{\star} \;=\; \mathbf{V}_t \;+\; \sum_{j\,:\,\pi(j)=t} \frac{A_j^{(l,h)}}{A_t^{(l,h)}+\epsilon}\,\mathbf{V}_j, \qquad t\in\mathcal{P}_{\text{keep}}.$$

Keys are unchanged. The retained-set size stays at B, so the row shares the 62.4 MB budget of the other selection-based baselines. This differs from CaM in the routing rule and the weighting but implements the same principle: recycle discarded KV content into retained slots rather than dropping it.

Method	Compression	MedQA	GSM8K	ARC-C	MBPP+	HEval+	Mean	Worst
Full (uncompressed)	1.0×	–	–	–	–	–	–	–
Layerwise	4.6×	−1.78	−2.40	−1.91	+1.15	−3.46	−1.68	−3.46
L-OBF	4.6×	−0.11	−1.24	−1.73	+1.06	**+0.20**	**−0.36**	**−1.73**
CaM-inspired merging	4.6×	−2.89	−1.72	−2.10	**+3.35**	−1.42	−0.96	−2.89
Full + int8	1.9×	**+0.22**	−0.99	**+0.80**	−1.85	−2.44	−0.85	−2.44
Full + int4	3.7×	−0.33	**−0.48**	+0.09	+0.18	−6.71	−1.45	−6.71
L-OBF + int8	**9.0×**	−2.89	−1.74	−1.54	+2.38	−3.05	−1.37	−3.05

Table 8: **KV compression comparisons on Qwen3-4B**, reproduced from Table 2. Accuracy changes (pp) are relative to Full. Compression is the relay-size reduction. Mean is the unweighted average change, and Worst is the minimum change across the five tasks. Best per column is in **bold**.

Analysis. At the 4.6× relay budget, L-OBF has the best Mean, the mildest Worst, and the tightest spread across benchmarks. Quantization and cache merging each score highest on one task but drop sharply on another, most visibly on HumanEval+. Stacking int8 on L-OBF pushes the ratio to 9.0× at a modest average loss, higher than any single-family method here reaches.

F EFFECT OF KV_BUDGET

We sweep kv_budget $B \in \{4, 8, 16, 32, 64\}$ for headwise eviction (H) and H-OBF at the paper's fixed rank r=2, on four benchmarks (GSM8K, ARC-C, MedQA, HumanEval+). All runs follow Appendix A.5. H-OBF here uses the fast SVD path of Appendix H, so the latency axis in Figure 8 already reflects the cheaper factorization.

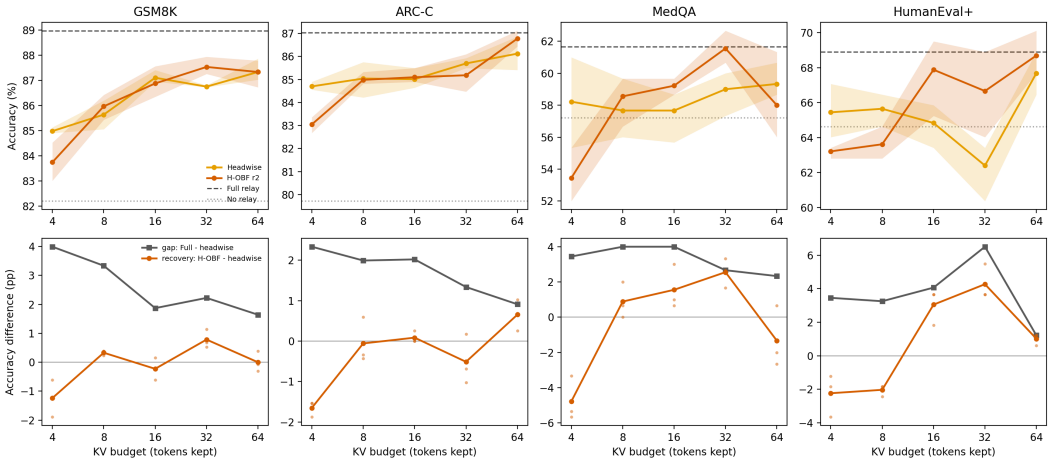


Figure 7: **Accuracy and recovery as functions of kv_budget.** Top row: accuracy of H (yellow) and H-OBF (orange) at $B \in \{4, 8, 16, 32, 64\}$; shaded bands are three-seed standard deviation. Full-relay and no-relay baselines are marked. Bottom row: eviction gap (Full − H, grey) and OBF recovery (H-OBF − H, orange) as functions of budget.

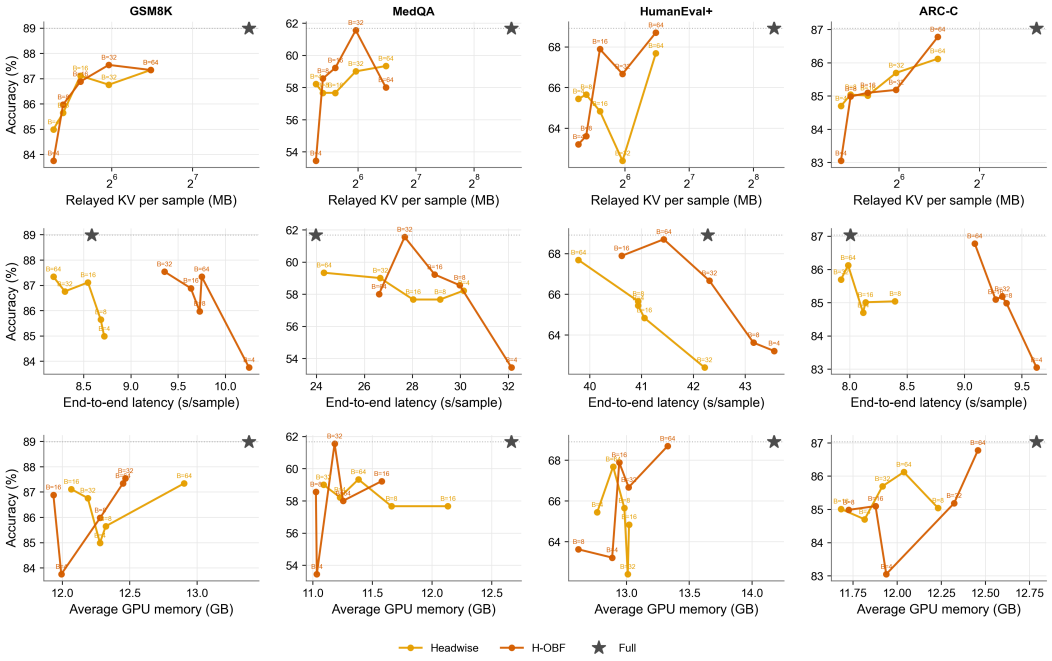


Figure 8: **Accuracy vs cost trade-offs.** For each benchmark, accuracy is plotted against three cost axes: relayed KV per sample (top row, log scale), end-to-end latency (middle row), and average GPU memory (bottom row). Each marker labelled B=k is one budget in $\{4, 8, 16, 32, 64\}$; the star is Full relay.

Accuracy and recovery. At the tested rank, OBF is less effective at very small budgets. At B=4, recovery is negative on all four benchmarks, most severely on MedQA, so injecting into a retained set that small hurts rather than helps. At B=8, recovery is positive on GSM8K and MedQA but remains negative on ARC-C and HumanEval+. It is more often positive at the intermediate budgets B=16 and B=32, where three of four benchmarks improve at each budget. The largest recoveries in this range appear on MedQA and HumanEval+, the two benchmarks where headwise eviction opens the widest gap against Full. Budget is therefore a separate axis from the cross-benchmark recovery analysis of Appendix C, which holds at fixed B=32: within a benchmark, recovery does not follow the gap monotonically as B varies.

This sweep holds `pca_rank` $= 2$ fixed. PCA rank controls the residual subspace used to construct the correction, not the rank of the uniform update matrix $\mathbf{1}_B\delta^\top$, which is at most one. The sweep therefore describes budget sensitivity at this rank; joint rank–budget tuning is not evaluated.

Cost trade-offs. At equal budget H-OBF relays the same KV volume as H and uses comparable GPU memory, but its residual extraction adds latency. Headwise eviction therefore reaches a given accuracy at lower end-to-end cost on GSM8K and ARC-C, while on MedQA and HumanEval+ the accuracy gain is large enough that H-OBF sits above H at every latency it occupies.

The latency axis should be read with one caveat. Our multi-agent chain is simulated within a single process on one device, so the relayed KV cache is handed to the next agent in memory and never crosses a link. Full relay consequently pays no transfer cost in these measurements, and its latency reflects only the extra attention over a longer prompt. Any deployment that places agents on separate devices or nodes would charge Full for the full relay volume, which is roughly an order of magnitude larger than the compressed variants (top row of Figure 8). Relayed KV volume follows from tensor shapes and precision. GPU memory and latency depend on the hardware and runtime, and these measurements exclude network transfer.

G EFFECT OF `pca_rank`

On Qwen3-4B, we sweep `pca_rank` over $\{2, 4, 8, 16, 32\}$ for both L-OBF and H-OBF on all nine benchmarks, with `pca_rank` $= 0$ marking the no-OBF baseline (the plain layerwise or headwise selector alone). All runs follow Appendix A.5.

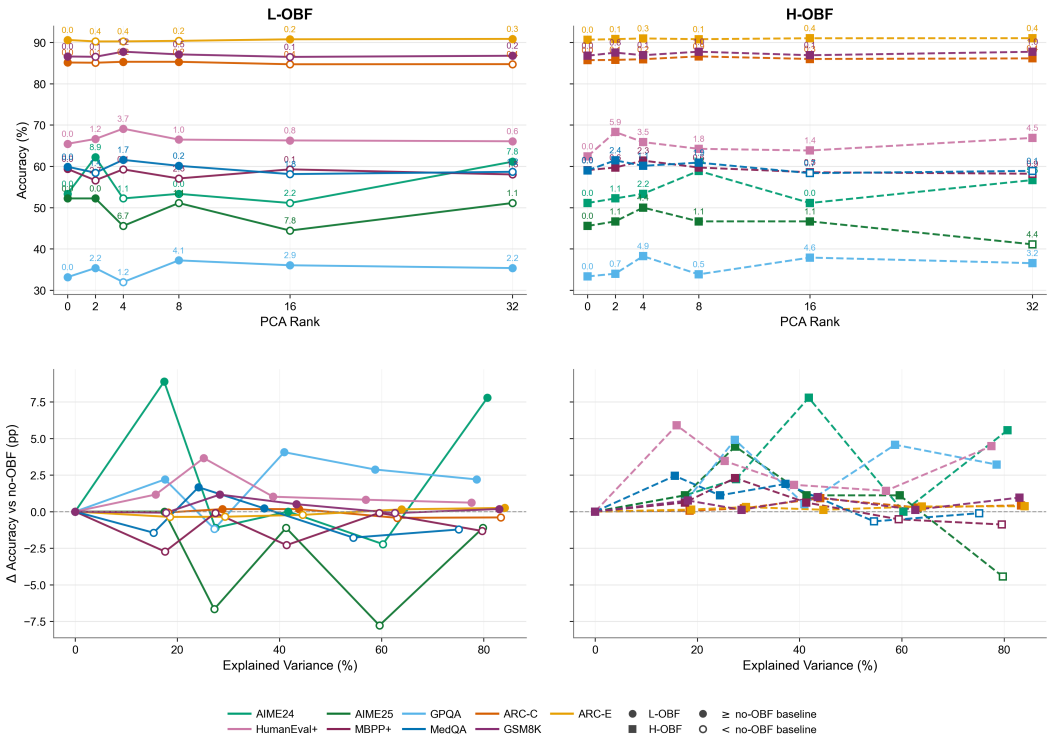


Figure 9: **Effect of `pca_rank` on OBF accuracy across nine benchmarks.** L-OBF (left) and H-OBF (right) across ranks $\{0, 2, 4, 8, 16, 32\}$. Top row: absolute accuracy. Bottom row: gain over the rank-0 no-OBF baseline plotted against the PCA explained variance ratio. Filled markers are at or above the baseline, hollow markers below.

Analysis. At a given `pca_rank`, the PCA explained variance ratio is comparable across benchmarks, so rank and EVR are largely equivalent axes. H-OBF matches or exceeds its no-OBF baseline on most benchmarks and ranks (filled markers on the right panel). Table 9 aggregates the seven non-AIME per-dataset rank orderings via a Borda count with positional weights $5, 4, 3, 2, 1$. Rank 4 has the highest combined score, while H-OBF ties at ranks 4 and 32; rank 16 has the lowest combined score. Table 10 reports the Qwen3-14B L-OBF rank comparison.

`pca_rank`	L-OBF	H-OBF	Combined
4	**27**	**23**	**50**
8	25	21	46
32	19	**23**	42
2	16	22	38
16	18	16	34

Table 9: **Borda-count scores for each `pca_rank` across seven non-AIME benchmarks.** Three seeds receive equal weight within each dataset; exact accuracy ties are ordered by ascending rank. Higher is better; **bold** marks the top rank per column.

Rank	GSM8K	GPQA	MedQA	ARC-E	ARC-C	MBPP+	HumanEval+	Mean
2	92.47	55.33	80.44	98.60	95.90	77.78	87.20	83.96
4	92.60	56.51	80.78	98.51	95.53	77.16	84.55	83.66
8	92.57	56.68	80.00	98.58	95.56	76.72	86.38	83.78

Table 10: **L-OBF rank comparison at Qwen3-14B.** Accuracy (%, three-seed mean) on the same seven benchmarks. Mean is the unweighted average across benchmarks.

H FAST SVD PATH FOR OBF COMPRESSION

Section 4.4 defines the injection direction Δ via the top-k right singular vectors of the residual matrix $\mathbf{R} \in \mathbb{R}^{N \times d}$. In the fast-path experiments, $k = \texttt{pca_rank} \leq 32$. The fast path approximates the required right singular subspace without a full SVD.

Algorithm. Algorithm 4 returns the residual basis used in place of the SVD_k call in Algorithm 3. It uses an oversampled width $w = \min(k + 10, d, N)$ and $T = 12$ cold-start iterations. Here QR returns the orthonormal Q factor of a reduced QR decomposition. The implementation uses FP32, batches the iteration across heads, and computes each head's residual as in the exact path. A CPU Gaussian generator seeded with $1234 + 97l + b$ initializes the batch of head bases for layer index l and batch-item index b; warm starts are disabled.

Algorithm 4 Fast residual principal basis (one KV head).

Require: Residual $\mathbf{R} \in \mathbb{R}^{N \times d}$; requested rank $k \geq 0$
if $k = 0$ **or** $N = 0$ **or** $\|\mathbf{R}\|_F \leq 10^{-10}$ **then**
  return empty basis ▷ no injection
end if
$k \leftarrow \min(k, d, N)$
$w \leftarrow \min(k + 10, d, N)$; $T \leftarrow 12$
$\mathbf{S} \leftarrow \text{QR}(\text{GAUSSIAN}(d, w; \text{fixed seed}))$ ▷ $\mathbf{S} \in \mathbb{R}^{d \times w}$
for $t = 1, \ldots, T$ **do**
  $\mathbf{S} \leftarrow \mathbf{R}^\top(\mathbf{R}\mathbf{S})$ ▷ do not materialize $\mathbf{R}^\top\mathbf{R}$
  if $t \bmod 2 = 0$ **or** $t = T$ **then**
    $\mathbf{S} \leftarrow \text{QR}(\mathbf{S})$
  end if
end for
$\mathbf{A} \leftarrow \mathbf{S}^\top\mathbf{R}^\top(\mathbf{R}\mathbf{S})$; $\mathbf{M} \leftarrow (\mathbf{A} + \mathbf{A}^\top)/2$
$\mathbf{U}_k \leftarrow$ eigenvectors of $\mathbf{M}$ with the k largest eigenvalues
return $\widehat{\mathbf{C}} \leftarrow (\mathbf{S}\mathbf{U}_k)^\top$

The returned basis gives $\widehat{\Delta} = (\bar{\mathbf{r}}\widehat{\mathbf{C}}^\top)\widehat{\mathbf{C}}$ through the remaining steps of Algorithm 3. Each iteration costs $O(Ndw + dw^2)$ for the two matrix products and periodic QR; the final projected eigendecomposition costs $O(Ndw + dw^2 + w^3)$. The total cost is $O((T+1)(Ndw + dw^2) + w^3)$, compared with $O(Nd\min(N, d))$ for a full SVD. A failed numerical decomposition produces no injection in the implementation.

Difference from exact SVD. For a fixed residual $\mathbf{R}$ and weighted mean $\bar{\mathbf{r}}$, the injection depends only on the selected subspace: any orthonormal basis spanning the same subspace gives the same Δ in exact arithmetic. Let $\mathbf{P} = \mathbf{C}^\top\mathbf{C}$ and $\widehat{\mathbf{P}} = \widehat{\mathbf{C}}^\top\widehat{\mathbf{C}}$ be the exact and fast rank-k orthogonal projectors. Their injection directions obey

$$\|\Delta - \widehat{\Delta}\|_2 = \|\bar{\mathbf{r}}(\mathbf{P} - \widehat{\mathbf{P}})\|_2 \leq \|\bar{\mathbf{r}}\|_2 \, \|\mathbf{P} - \widehat{\mathbf{P}}\|_2 = \|\bar{\mathbf{r}}\|_2 \sin\theta_{\max}, \tag{8}$$

where $\theta_{\max}$ is the largest principal angle between the two rank-k subspaces. The inequality follows from the induced matrix norm; the final equality is the projector-distance identity for equal-dimensional subspaces. For the same retained values and demand ratio α, uniform injection into B slots gives, before rounding to the cache dtype,

$$\|\mathbf{V}^{\text{fast}}_{\text{keep}} - \mathbf{V}^{\text{exact}}_{\text{keep}}\|_F = \sqrt{B}\,|\alpha|\,\|\widehat{\Delta} - \Delta\|_2 \leq \sqrt{B}\,|\alpha|\,\|\bar{\mathbf{r}}\|_2 \sin\theta_{\max}. \tag{9}$$

These relations quantify the local effect of subspace approximation. The angle depends on the residual spectrum and solver convergence. Below we evaluate the resulting numerical differences on matched inputs and task accuracy on complete rollouts.

Matched-input numerical check. We compare the returned value caches on a 20-sample GSM8K probe with Qwen3-4B, L-OBF, $B = 32$, and $k = 4$. At each relay boundary, both compressors

receive the same inputs, and only the exact path's output is passed to the next agent. This yields 60 relay boundaries and 2,160 layer comparisons. For boundary m and layer l, we measure

$$e_{m,l} = \frac{\|\mathbf{V}^{\text{fast}}_{m,l} - \mathbf{V}^{\text{exact}}_{m,l}\|_F}{\|\mathbf{V}^{\text{exact}}_{m,l}\|_F + 10^{-12}}, \qquad E_m = \frac{\left(\sum_l \|\mathbf{V}^{\text{fast}}_{m,l} - \mathbf{V}^{\text{exact}}_{m,l}\|_F^2\right)^{1/2}}{\left(\sum_l \|\mathbf{V}^{\text{exact}}_{m,l}\|_F^2\right)^{1/2} + 10^{-12}}. \tag{10}$$

The norms are evaluated in FP32 on the returned cache tensors, including all heads, retained history, and sink entries. Table 11 reports the recorded results. Only 2 of the 2,160 layer comparisons exceed 10^{-3}, and none exceed 10^{-2}; neither path produces non-finite entries or a cache-shape mismatch.

Relative cache error	Median	P90	P99	Maximum
Per layer ($e_{m,l}$)	7.06×10^{-5}	1.37×10^{-4}	2.31×10^{-4}	8.133×10^{-3}
Whole cache (E_m)	—	—	—	1.907×10^{-4}

Table 11: **Fast versus exact numerical agreement on identical inputs.** L-OBF on 20 GSM8K samples; layer statistics cover 2,160 comparisons and the whole-cache maximum covers 60 relay boundaries. Dashes indicate statistics not retained in the original log.

Setup for timing. We compare the original main-experiment exact runs with the original fast-SVD campaign on Qwen3-4B at $B = 32$, using the same five benchmarks and seeds $\{4, 44, 444\}$ for each method. We exclude subsequent budget-sweep runs. *Compression time per sample* is the recorded compression wall clock per rollout. We average over seeds within each benchmark, then equally across benchmarks, and compute aggregate speedups as the ratio of these means. Appendix I uses the same runs and aggregation.

Timing. Figure 10 shows compression time per sample for the exact and fast paths on each benchmark. Averaged across the five benchmarks, the fast path drops compression from 6.95 s to 0.91 s per sample for L-OBF (7.6× overall, 7.1 to 8.2× across benchmarks) and from 7.14 s to 1.32 s for H-OBF (5.4× overall, 5.0 to 5.9×).

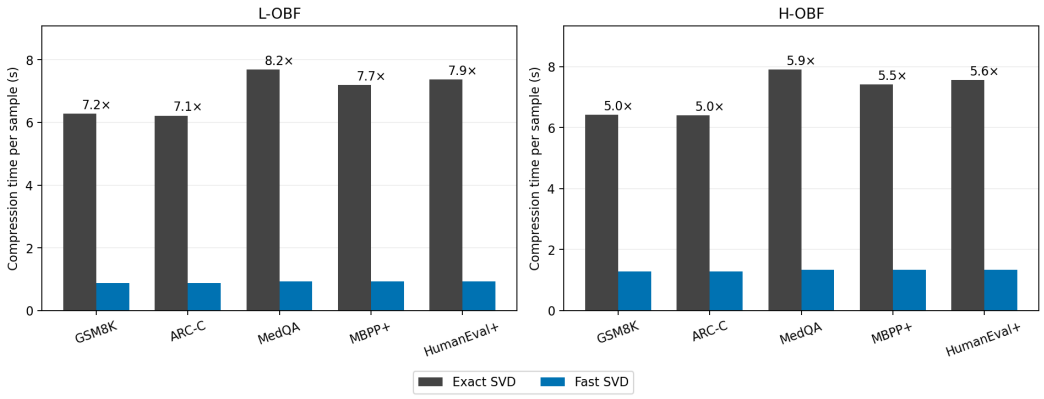


Figure 10: **Fast SVD path compression time.** Per-sample compression time (three-seed mean) for the exact and fast paths on Qwen3-4B (A100 40GB PCIe). Annotations above each pair are the per-benchmark speedup.

The end-to-end impact of this speedup, together with a full comparison against Full and the eviction-only baselines, is reported in Appendix I.

Accuracy verification. Following the matched-input cache check, we compare complete rollouts under stochastic decoding. Figure 11 plots the fast-minus-exact accuracy delta per benchmark against the exact path's own seed noise band (±1 SD).

For H-OBF, the fast path shifts accuracy by −0.26 pp on average, with four of five benchmarks within the exact path's across-seed ±1 SD band.

For L-OBF, the average shift is -1.33 pp, concentrated on HumanEval+ (-4.27 pp seed-averaged; the other four benchmarks lie between -1.33 and $+0.09$ pp). HumanEval+ has the longest generation ($\sim$ 2000 tokens), the smallest test set (164 problems), and a pass@1 metric that is sensitive to single-token substitutions in code, all of which amplify any per-step drift. Increasing the number of subspace iterations can improve approximation accuracy and may help narrow the HumanEval+ gap, at the cost of additional computation. We leave further SVD optimization to future work. We therefore retain the exact SVD for the main-text accuracy results. The fast path is used only in analyses that explicitly identify it, including the budget sweep and the efficiency measurements, and we report its residual accuracy drift here.

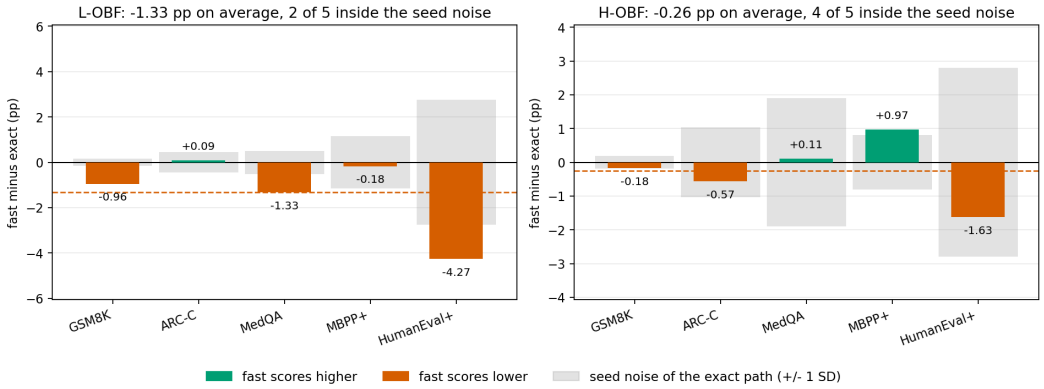


Figure 11: **Fast SVD path accuracy against exact.** Per-benchmark fast-minus-exact accuracy delta (pp, three-seed mean). Grey bands are ± 1 SD of the exact path across its own seeds; bars inside the band fall within that descriptive across-seed range.

I END-TO-END COST BREAKDOWN

We decompose the total cost of one rollout into four quantities: relay bandwidth per non-judger agent, compression work per relay boundary, tokens generated at the final agent, and end-to-end wall clock. All runs follow Appendix A.5, restricted to the five benchmarks with three-seed coverage for every method (GSM8K, ARC-C, MedQA, MBPP+, HumanEval+).

I.1 COST ACROSS COMPRESSION METHODS

Method	Relay (MB)	Relay ratio	Compression (s)	Output tokens	Decode (s)	Throughput (tok/s)	End-to-end (s)	E2E ratio
Full	290.1	1.00×	0.00	1149	23.47	48.97	24.69	1.00×
L	62.4	4.65×	0.05	1247	**23.08**	**54.01**	**24.37**	**0.99**×
L-OBF (fast)	62.4	4.65×	0.91	1274	23.67	53.83	25.79	1.04×
H	62.4	4.65×	0.07	1246	23.59	52.81	24.90	1.01×
H-OBF (fast)	62.4	4.65×	1.32	1237	23.39	52.87	25.92	1.05×
Gen	**35.4**	**8.18**×	0.00	1269	24.77	51.23	25.97	1.05×

Table 12: **End-to-end cost of one rollout across compression methods.** *Relay* is the KV size transferred per non-judger agent; *Decode* is the final agent's text-generation time, excluding compression, and *Throughput* is its output tokens divided by that time; the relay ratio is Full divided by the method, while the E2E ratio is the method divided by Full. OBF uses the fast SVD path (Appendix H). Bold marks the best value in each column: the minimum for costs, the maximum for throughput.

Analysis. Eviction-based methods transfer roughly 62 MB against Full's 290 MB, a $4.7\times$ reduction, and Gen goes further to 35 MB ($8.2\times$). L is slightly faster end-to-end than Full (24.37 vs 24.69 s). Every compressed method sits within 5.2% of Full's wall clock even after paying for compression, with L essentially free and OBF adding about 1 s per sample under the fast SVD path (Appendix H). The *Decode* and *Throughput* columns isolate final-agent generation from compression. Every eviction-based method decodes 8–10% faster than Full in tokens per second, because a shorter relayed prompt makes each decoding step cheaper, so the extra 8–11% of tokens costs no additional decoding time. These timings use a single-process chain on one device, with in-memory cache handoffs and no network transfer. The relay column measures cache volume, while the end-to-end column measures runtime in this co-located setting. GPU memory is reported in Section I.3.

I.2 FAST SVD VS EXACT SVD FOR OBF

The exact and fast SVD paths use the same retention budget, but stochastic rollouts can produce different output lengths. Table 13 focuses on compression time and end-to-end wall clock.

Method	Compression (s)	Compression speedup	End-to-end (s)	E2E ratio
Full	0.00	–	24.69	1.00×
L-OBF (exact)	6.95	1.0×	31.55	1.28×
L-OBF (fast)	**0.91**	**7.6**×	25.79	**1.04**×
H-OBF (exact)	7.14	1.0×	31.81	1.29×
H-OBF (fast)	**1.32**	**5.4**×	25.92	**1.05**×

Table 13: **Fast SVD path against exact SVD for OBF.** *Compression speedup* is exact time divided by fast time. *E2E ratio* divides end-to-end wall clock by Full's.

Analysis. Under the exact SVD, OBF's ~ 7 s per-sample compression cost pushes end-to-end wall clock to $1.28\times$–$1.29\times$ Full. The fast path (Appendix H) cuts compression by $5.4\times$–$7.6\times$ and brings end-to-end to $1.04\times$–$1.05\times$ Full, so the $4.7\times$ bandwidth reduction from Table 12 is delivered at essentially matched wall clock.

I.3 GPU MEMORY

Table 14 completes the cost picture with device GPU memory across the three backbones, separating the time-averaged footprint from the peak, which is what determines whether a chain fits on a card and how many can run concurrently.

	Qwen3-4B (n=26)			Qwen3-8B (n=8)			Qwen3-14B (n=14)		
Method	Mean (GB)	Peak (GB)	ΔPeak	Mean (GB)	Peak (GB)	ΔPeak	Mean (GB)	Peak (GB)	ΔPeak
Full	13.54	17.62	–	21.80	26.45	–	42.99	48.91	–
L	12.62	14.92	−15.3%	20.89	24.54	−7.2%	41.57	46.48	−5.0%
L-OBF	12.59	14.89	−15.5%	20.85	24.60	−7.0%	41.46	46.48	−5.0%
H	12.68	14.95	−15.2%	20.90	24.82	−6.2%	41.56	46.48	−5.0%
H-OBF	12.59	15.01	−14.8%	20.80	24.67	−6.7%	41.39	46.48	−5.0%
Gen	12.66	14.90	−15.4%	21.00	24.51	−7.4%	41.77	46.41	−5.1%

Table 14: **GPU memory across backbones.** The L-OBF memory row uses rank 4 at all scales, including 14B; the reported 14B accuracy row in Table 5 uses rank 2. *Mean* is device GPU memory time-averaged over a run and *Peak* is its sampled maximum (wandb system metrics, one sample every 15–30 s); ΔPeak is relative to Full at the same scale. n counts the (benchmark, seed) cells in which all six methods ran on the same node. Absolute values are not comparable across backbones, which used different cards (A100 for 4B and 8B, H200 for 14B); the ratios are.

Analysis. Compression cuts peak memory by 15% at 4B and by 5–7% at 8B and 14B. The columns use different benchmark mixes: the 4B column covers all nine benchmarks, while the 8B and 14B columns cover the seven with matched coverage, excluding the two AIME sets whose long generations enlarge the cache for every method. The compressed methods have similar measured footprints. OBF preserves the cache shape of its paired eviction selector and adds no retained positions; Gen uses a different retention policy. The table includes all matched cells. One 14B node was shared with an unrelated job for part of the campaign, so its device-level memory measurements include that process. Restricting the 14B comparison to unshared nodes gives a peak-memory reduction of 7.7%, compared with 5.0% across all matched nodes.

J Additional Analysis of Downstream Generation Cost

We examine how accuracy correlates with per-agent runtime cost. Each panel overlays a dashed least-squares fit and reports the Pearson correlation coefficient r. The *Average* panels correlate method-level aggregate accuracy and cost, rather than averaging the per-dataset correlation coefficients.

J.1 Latent-Side Cost

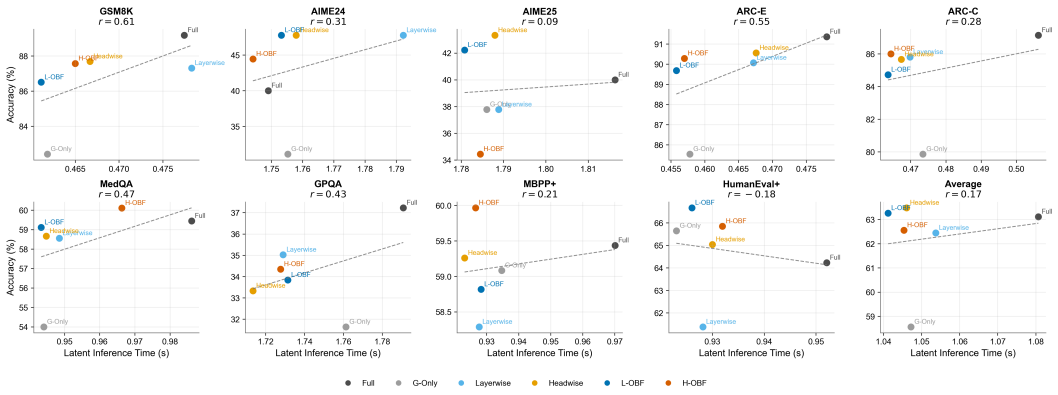


Figure 12: **Accuracy versus latent inference time.** Latent inference time is reported as the average runtime across all agents that perform latent inference.

Analysis. The *Average* panel shows a weak positive association between latent inference time and accuracy across methods ($r = +0.17$). The per-dataset correlations vary in magnitude and are positive except on HumanEval+.

J.2 Text-Side Cost

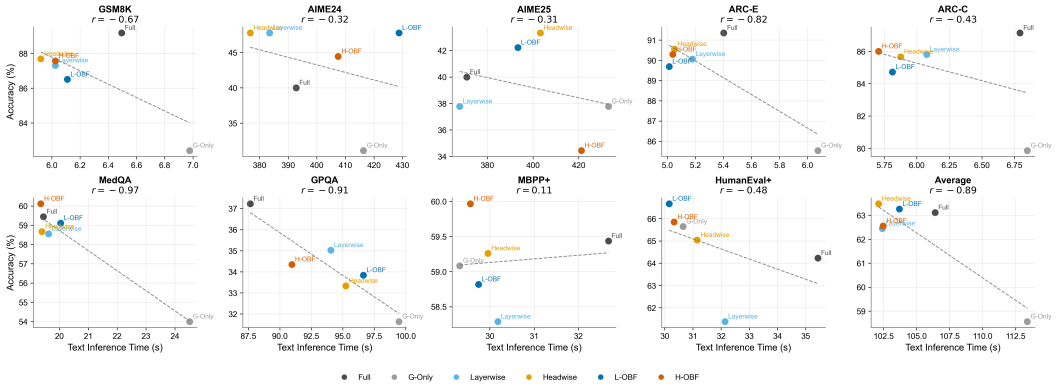


Figure 13: **Accuracy versus text inference time.** This figure measures the runtime of text generation by the final agent.

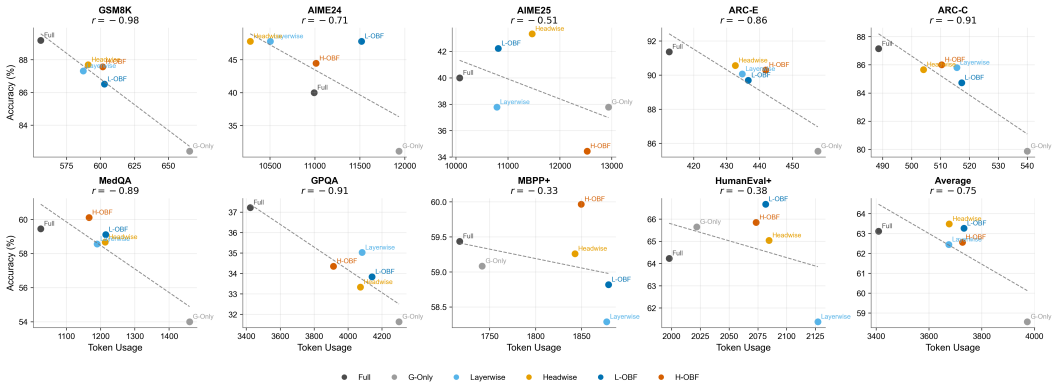


Figure 14: **Accuracy versus final-agent token usage.** Token usage counts only the tokens generated by the final agent.

Analysis. In the *Average* panels, accuracy is negatively correlated with text inference time ($r = -0.89$) and final-agent token usage ($r = -0.75$) across methods. Shorter generations generally accompany higher accuracy on reasoning and QA tasks. The coding results are less consistent.

K Internal Behavior of OBF

K.1 Diagnostic Setup and Representative Example

This appendix examines how OBF behaves internally across layers and heads, complementing the benchmark-level results in the main text. We report a set of diagnostic curves that characterize the magnitude, recoverability, and relative contribution of the injected residual vector throughout the network. These diagnostics use individual Qwen3-4B runs rather than three-seed accuracy averages; their configurations are specified below. Figure 15 shows the diagnostic curves of L-OBF on GSM8K as a running example from seed 444 and the first relay boundary on an NVIDIA A100-PCIE-40GB.

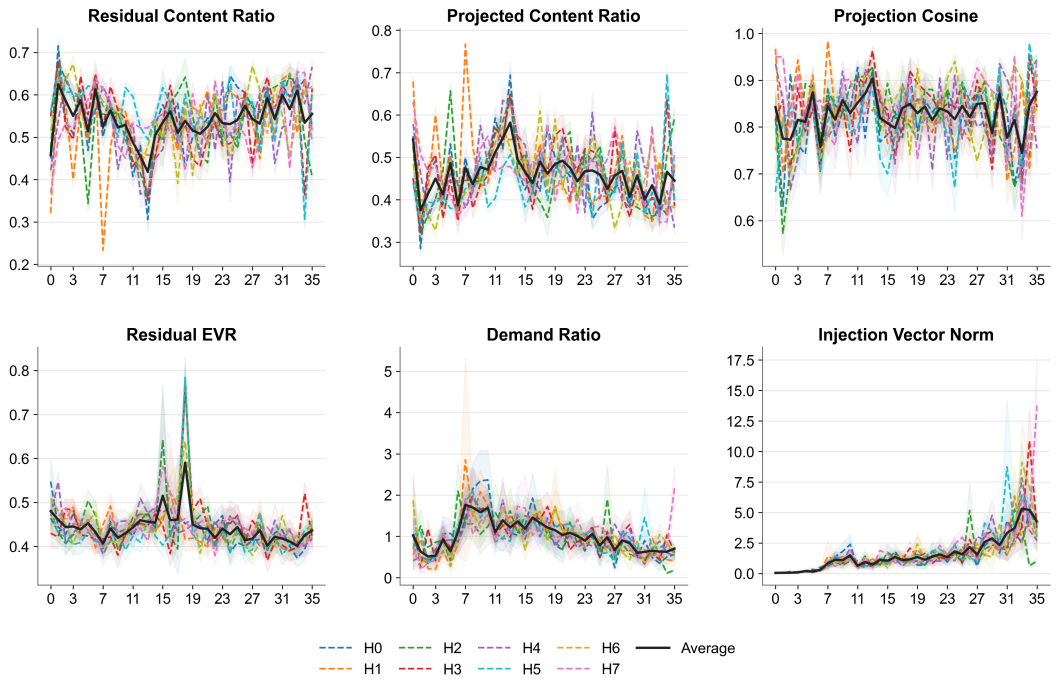


Figure 15: **Representative internal diagnostics of L-OBF on GSM8K.** The six panels report the metrics defined in Appendix K.2; the horizontal axis is the layer index. Colored dashed curves correspond to individual attention head groups, the solid black curve shows the across-head average, and the shaded region indicates variability across samples.

K.2 Diagnostic Metrics

We define the internal diagnostic quantities used throughout this appendix. All metrics are computed for each layer-head pair (l, h) on the residual matrix $\mathbf{R}$.

Projected Content Ratio (PCR) and Residual Content Ratio (RCR). Recall from Eq. 3 that the deleted value states can be decomposed as

$$\mathbf{V}_{\text{del}} = \mathbf{V}_{\parallel} + \mathbf{R}, \qquad \text{where } \mathbf{V}_{\parallel} = (\mathbf{V}_{\text{del}}\mathbf{Q})\mathbf{Q}^{\top}.$$

We define

$$\text{PCR} = \frac{\|\mathbf{V}_{\parallel}\|_F^2}{\|\mathbf{V}_{\text{del}}\|_F^2 + \epsilon}, \qquad \text{RCR} = \frac{\|\mathbf{R}\|_F^2}{\|\mathbf{V}_{\text{del}}\|_F^2 + \epsilon}.$$

Since $\mathbf{V}_{\parallel}$ is the projection of $\mathbf{V}_{\text{del}}$ onto the span of the kept value states, $\mathbf{V}_{\parallel}$ and $\mathbf{R}$ form an orthogonal decomposition of $\mathbf{V}_{\text{del}}$, so $\text{PCR} + \text{RCR} \approx 1$. PCR measures how much of the deleted information is representable by that span; RCR measures how much is not.

Projection Cosine (PC). We also define

$$\mathrm{PC} = \mathrm{CosSim}\left(\sum_{j=1}^{|\mathcal{P}_{\mathrm{del}}|} \mathbf{V}_{\mathrm{del}}[j,:],\ \sum_{j=1}^{|\mathcal{P}_{\mathrm{del}}|} \mathbf{V}_{\parallel}[j,:]\right),$$

which measures whether the representable part remains directionally consistent with the deleted information.

Residual Explained Variance Ratio (REVR). Recall from Section 4.4 that OBF applies SVD to the residual matrix $\mathbf{R}$ and retains its top-k right singular vectors. Let $\sigma_1, \ldots, \sigma_m$ denote the singular values of $\mathbf{R}$. We define

$$\mathrm{REVR} = \frac{\sum_{j=1}^{p} \sigma_j^2}{\sum_{j=1}^{m} \sigma_j^2 + \epsilon}. \tag{11}$$

REVR measures how much of the residual information the top-p principal subspace retains; a high value indicates low-rank structure.

Deleted-versus-Retained Demand Ratio (DR). Recall from Eq. 6 that OBF uses the deleted-versus-retained demand ratio for dynamic scaling. We define

$$\mathrm{DR} = \frac{\tilde{A}_{\mathrm{del}}}{\tilde{A}_{\mathrm{keep}} + \epsilon}.$$

DR measures how much attention falls on deleted positions relative to kept ones.

Injection Vector Norm (IVN). Recall from Eq. 6 that OBF forms the final injected correction δ after dynamic scaling. We define

$$\mathrm{IVN} = \|\delta\|_2.$$

Under the uniform injection rule in Eq. 7, the same δ is added to all kept positions, so IVN captures the overall strength of OBF's correction.

K.3 L-OBF vs. H-OBF

Figure 16 compares L-OBF and H-OBF on GSM8K, AIME2024, ARC-C, MedQA, GPQA, MBPP+, and HumanEval+. These diagnostics use rank 8 for both methods, budget 32, seed 444, 40 latent steps, and the second relay boundary on an A100-PCIE-40GB.

Representability metrics. Across both methods, PCR is about 40%–50%, PC stays around 0.8 or above, and REVR is also about 40%–50%, meaning that roughly half of the deleted information is representable by the kept value states, that representable part is directionally consistent, and the first $p = 8$ residual components capture about half of the residual energy.

Injection-side gap. The gap between L-OBF and H-OBF concentrates in Demand Ratio and Injection Vector Norm, especially in later layers. A plausible reason is that layerwise selection uses a retained set shared across heads: for some heads the kept tokens are not the ones receiving the highest attention, which increases deleted demand and decreases kept demand, thereby raising DR and the injection strength.

K.4 Effect of `kv_budget`

Figure 17 shows a separate diagnostic budget sweep for L-OBF and H-OBF on GPQA, ARC-C, HumanEval+, and MedQA, using seed 444 and 40 latent steps. It is distinct from the three-seed accuracy sweep in Appendix F.

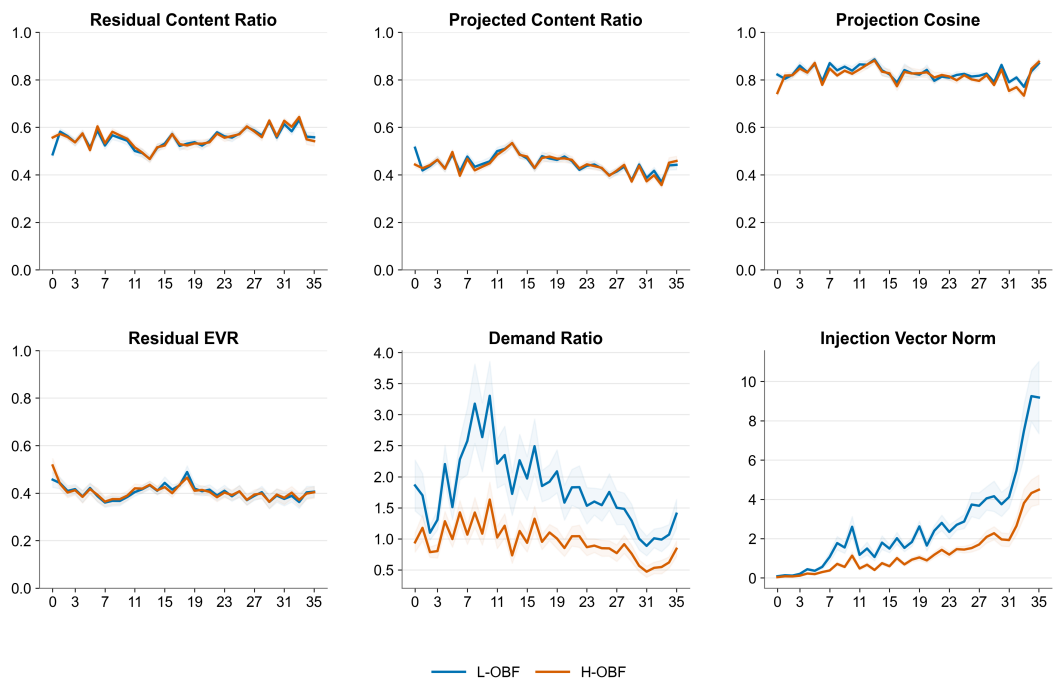


Figure 16: **Internal diagnostics of L-OBF versus H-OBF**, averaged across the seven diagnostic benchmarks, with rank 8 for both methods and seed 444. The first four metrics (PCR, RCR, PC, REVR) are nearly identical between the two methods; L-OBF shows a larger Demand Ratio and Injection Vector Norm, especially in later layers.

First four panels. Increasing the keep budget makes more deleted information representable by the kept value states. The Projected Content Ratio rises from about 0.16 at `kv_budget` $= 8$ to about 0.68 at `kv_budget` $= 64$, with the Residual Content Ratio decreasing accordingly. The Projection Cosine also increases, indicating stronger directional consistency. Residual EVR, by contrast, follows a rough U-shape: at small budgets less deleted information is absorbed by the projection, so more structure remains in the residual and PCA captures a relatively larger fraction of it; at large budgets the residual itself shrinks, which again makes it easier to cover.

Last two panels. Both Demand Ratio and Injection Vector Norm decrease monotonically as the keep budget grows, because a larger budget raises A_{keep} and lowers A_{del} in Eq. 6, so the deleted-versus-retained demand ratio shrinks and less correction is needed. The L-OBF–H-OBF gap from Section K.3 persists across all budgets.

K.5 Effect of `pca_rank`

Figure 18 shows first-boundary L-OBF diagnostics under different PCA ranks, aggregated across the available per-dataset diagnostic runs rather than across the three-seed accuracy sweep. Only Residual EVR and Injection Vector Norm increase monotonically with the PCA rank; the remaining metrics stay nearly constant. REVR rises from about 0.18 at `pca_rank=2` to about 0.82 at `pca_rank=32`, and IVN increases from about 1.65 to about 2.21.

This is expected: changing the PCA rank does not change which tokens are kept or deleted, so the projection step is unchanged and PCR, RCR, PC, and Demand Ratio remain nearly fixed. Retaining more residual principal directions directly raises REVR, and because the injected vector is built from the retained residual subspace, IVN rises with it. Even at `pca_rank=32`, REVR stays below 0.95, so the residual information is spread over a broad subspace rather than concentrated in a few directions.

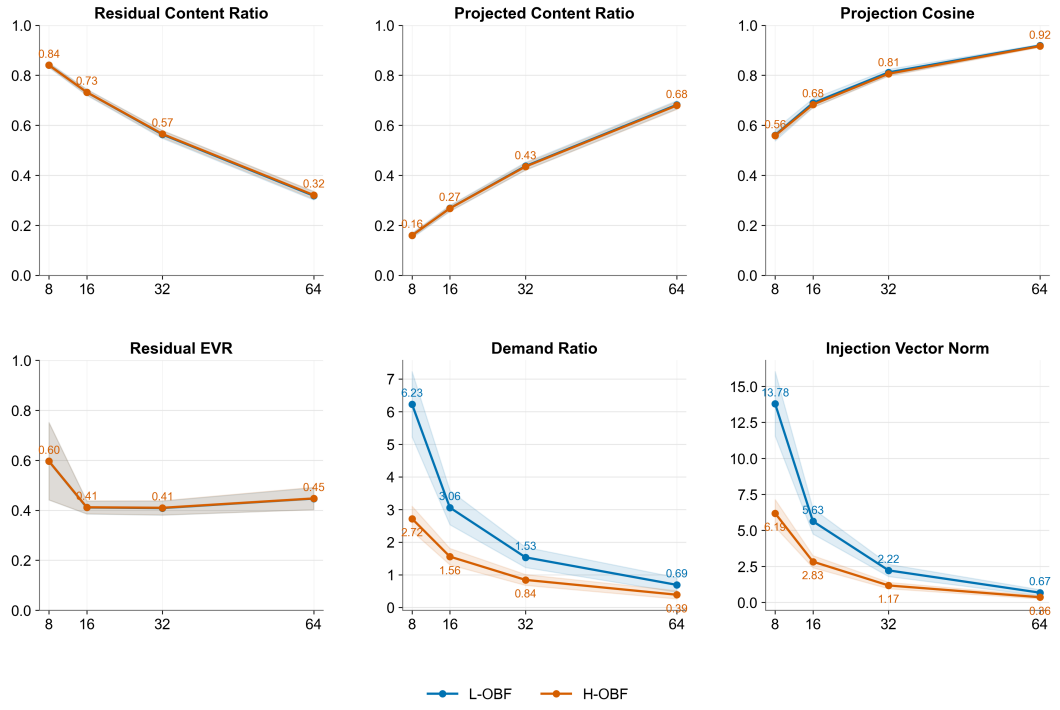


Figure 17: **Internal diagnostics under different keep budgets** (`kv_budget` $\in \{8, 16, 32, 64\}$). As the budget grows, more deleted content becomes representable by the kept value states (PCR↑, RCR↓, PC↑); the L-OBF–H-OBF gap in Demand Ratio and Injection Vector Norm persists but narrows.

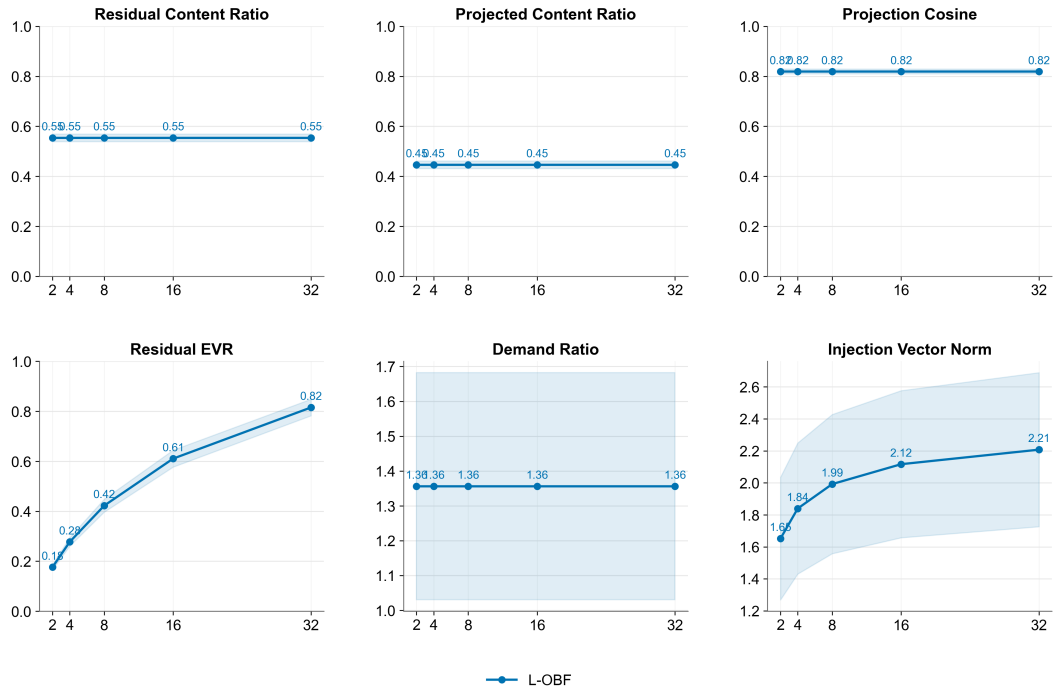


Figure 18: **Internal diagnostics under different PCA ranks** (`pca_rank` $\in \{2, 4, 8, 16, 32\}$). Only Residual EVR and Injection Vector Norm increase monotonically with rank; PCR, RCR, PC, and Demand Ratio are essentially unchanged. REVR remains below 0.95 even at rank 32.

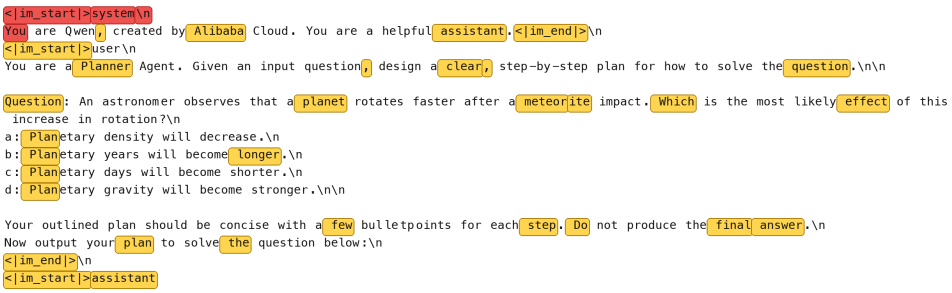
L TOKEN SELECTION VISUALIZATION ACROSS GRANULARITIES

We visualize the prompt tokens selected for relay under the layerwise (L) and headwise (H) granularities defined in Section 4.3.2, on a representative ARC-C sample. For each of the three intermediate agents (Planner, Critic, Refiner) and each granularity, we show an early, middle, and late layer to illustrate how retention varies with depth. Red highlights mark attention sinks; yellow highlights mark tokens ranked by attention mass. Note that this appendix uses the Qwen3-8B backbone (rather than the Qwen3-4B backbone used in the main experiments); all other settings follow Appendix A.5 except for the overrides below.

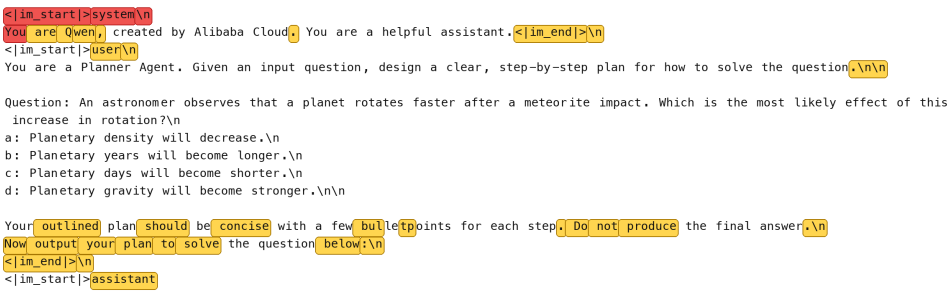
```
{
  "model_name": "Qwen/Qwen3-8B",
  "seed": 888,
  "device": "NVIDIA RTX 6000 Ada Generation GPU"
}
```

L.1 LAYERWISE TOKEN SELECTION

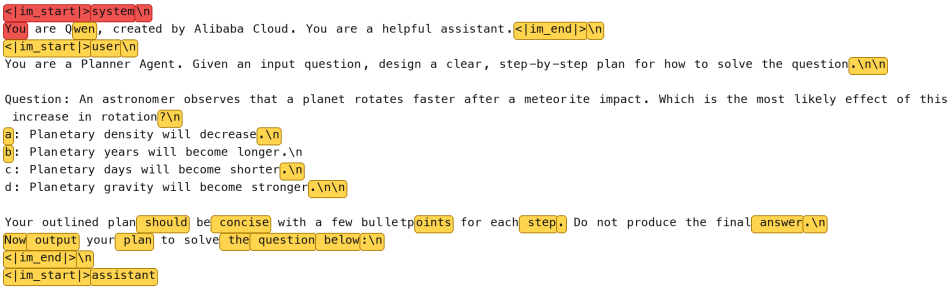
Figures 19, 20, and 21 show layerwise selection for the Planner, Critic, and Refiner agents respectively.



(a) Early layer (layer 0).



(b) Middle layer (layer 17).



(c) Late layer (layer 35).

Figure 19: Layerwise token selection for the Planner agent.

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Critic Agent to evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan.\n
The plan information is provided in latent KV representation format. Review the plan and question and output:\n
(1) original plan contents\n
(2) constructive feedback on the original plan.\n\n

Format your response as follows:\n
Original Plan: [Copy the provided Planner Agent's plan here]\n
Feedback: [Your detailed feedback to improve the plan here]\n\n

Now, output your response below:\n
<|im_end|>\n
<|im_start|>assistant

(a) Early layer (layer 0).

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Critic Agent to evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan.\n
The plan information is provided in latent KV representation format. Review the plan and question and output:\n
(1) original plan contents\n
(2) constructive feedback on the original plan.\n\n

Format your response as follows:\n
Original Plan: [Copy the provided Planner Agent's plan here]\n
Feedback: [Your detailed feedback to improve the plan here]\n\n

Now, output your response below:\n
<|im_end|>\n
<|im_start|>assistant

(b) Middle layer (layer 17).

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

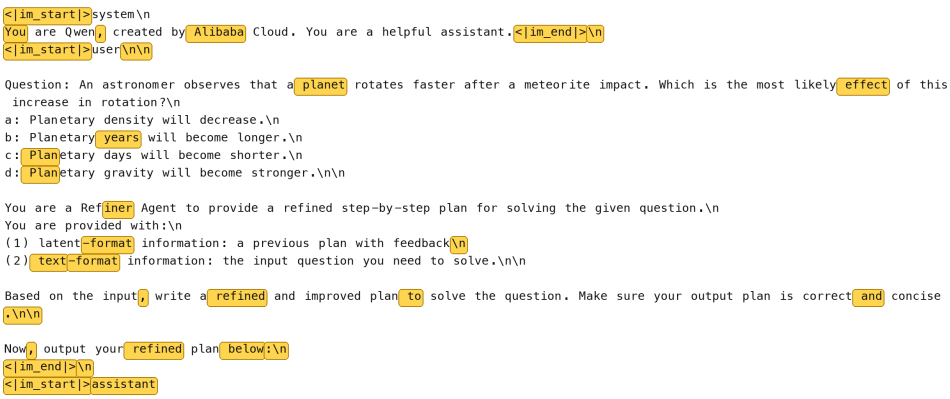
You are a Critic Agent to evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan.\n
The plan information is provided in latent KV representation format. Review the plan and question and output:\n
(1) original plan contents\n
(2) constructive feedback on the original plan.\n\n

Format your response as follows:\n
Original Plan: [Copy the provided Planner Agent's plan here]\n
Feedback: [Your detailed feedback to improve the plan here]\n\n

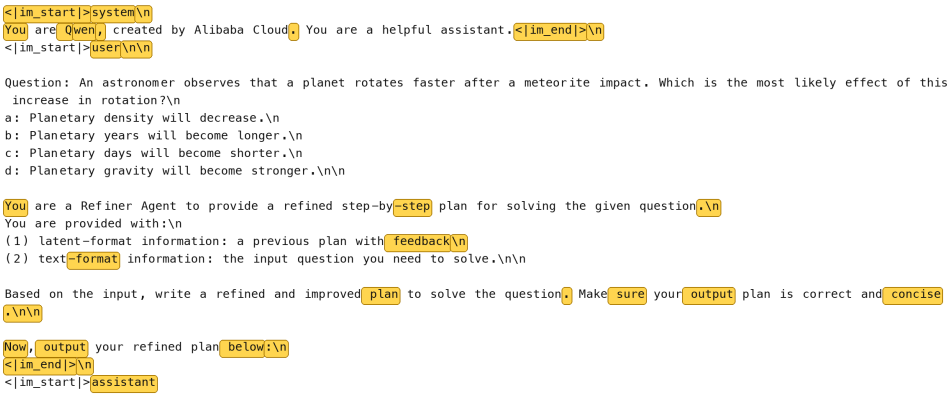
Now, output your response below:\n
<|im_end|>\n
<|im_start|>assistant

(c) Late layer (layer 35).

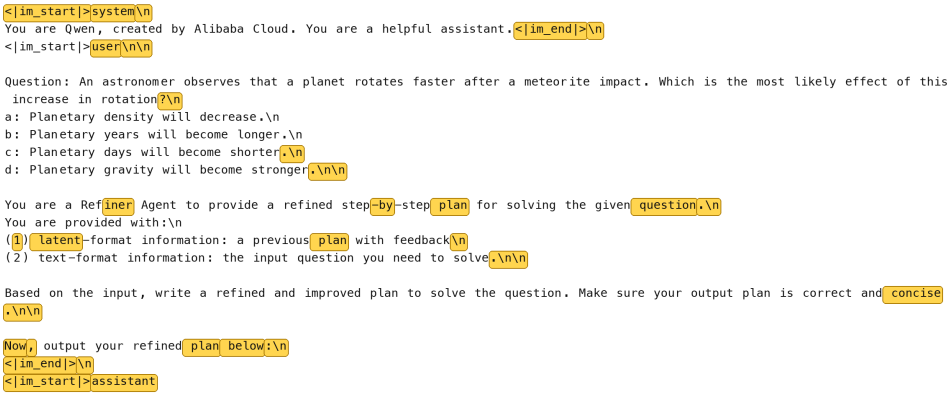
Figure 20: Layerwise token selection for the Critic agent.



(a) Early layer (layer 0).



(b) Middle layer (layer 17).

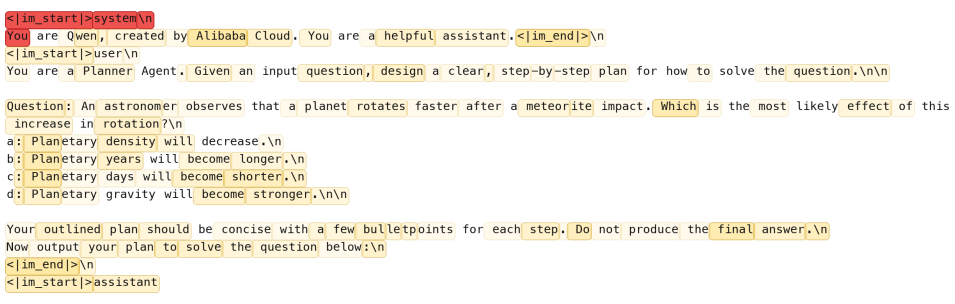


(c) Late layer (layer 35).

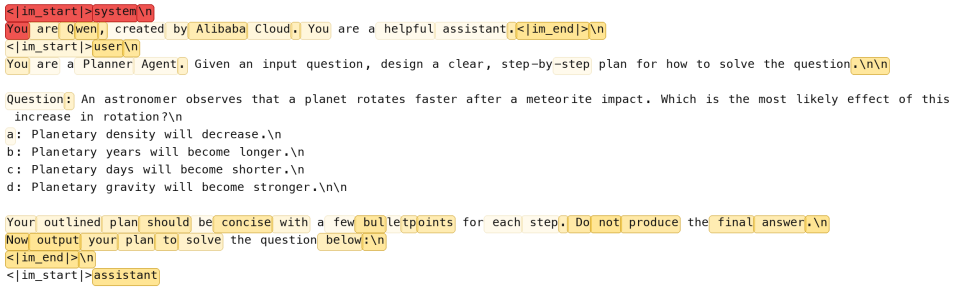
Figure 21: Layerwise token selection for the Refiner agent.

L.2 HEADWISE TOKEN SELECTION

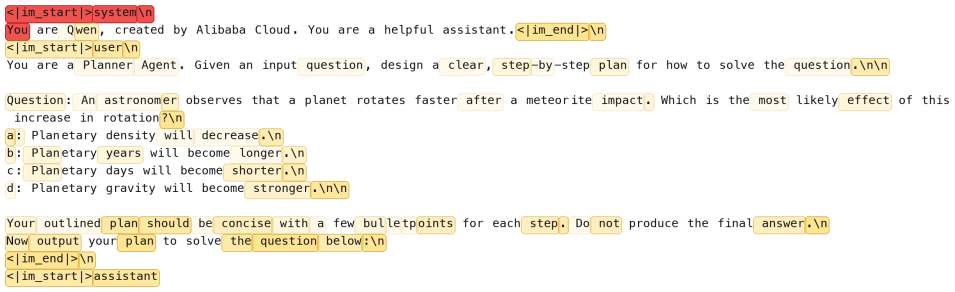
Figures 22, 23, and 24 show headwise selection for the Planner, Critic, and Refiner agents respectively.



(a) Early layer (layer 0).



(b) Middle layer (layer 17).



(c) Late layer (layer 35).

Figure 22: Headwise token selection for the Planner agent.

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Critic Agent to evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan.\n
The plan information is provided in latent KV representation format. Review the plan and question and output:\n
(1) original plan contents\n
(2) constructive feedback on the original plan.\n\n

Format your response as follows:\n
Original Plan: [Copy the provided Planner Agent's plan here]\n
Feedback: [Your detailed feedback to improve the plan here]\n\n

Now, output your response below:\n
<|im_end|>\n
<|im_start|>assistant

(a) Early layer (layer 0).

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Critic Agent to evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan.\n
The plan information is provided in latent KV representation format. Review the plan and question and output:\n
(1) original plan contents\n
(2) constructive feedback on the original plan.\n\n

Format your response as follows:\n
Original Plan: [Copy the provided Planner Agent's plan here]\n
Feedback: [Your detailed feedback to improve the plan here]\n\n

Now, output your response below:\n
<|im_end|>\n
<|im_start|>assistant

(b) Middle layer (layer 17).

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Critic Agent to evaluate the correctness of the input plan for the given question and provide helpful feedback for improving the plan.\n
The plan information is provided in latent KV representation format. Review the plan and question and output:\n
(1) original plan contents\n
(2) constructive feedback on the original plan.\n\n

Format your response as follows:\n
Original Plan: [Copy the provided Planner Agent's plan here]\n
Feedback: [Your detailed feedback to improve the plan here]\n\n

Now, output your response below:\n
<|im_end|>\n
<|im_start|>assistant

(c) Late layer (layer 35).

Figure 23: Headwise token selection for the Critic agent.

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Refiner Agent to provide a refined step-by-step plan for solving the given question.\n
You are provided with:\n
(1) latent-format information: a previous plan with feedback\n
(2) text-format information: the input question you need to solve.\n\n

Based on the input, write a refined and improved plan to solve the question. Make sure your output plan is correct and concise.\n\n

Now, output your refined plan below:\n
<|im_end|>\n
<|im_start|>assistant

(a) Early layer (layer 0).

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Refiner Agent to provide a refined step-by-step plan for solving the given question.\n
You are provided with:\n
(1) latent-format information: a previous plan with feedback\n
(2) text-format information: the input question you need to solve.\n\n

Based on the input, write a refined and improved plan to solve the question. Make sure your output plan is correct and concise.\n\n

Now, output your refined plan below:\n
<|im_end|>\n
<|im_start|>assistant

(b) Middle layer (layer 17).

<|im_start|>system\n
You are Qwen, created by Alibaba Cloud. You are a helpful assistant.<|im_end|>\n
<|im_start|>user\n\n

Question: An astronomer observes that a planet rotates faster after a meteorite impact. Which is the most likely effect of this increase in rotation?\n
a: Planetary density will decrease.\n
b: Planetary years will become longer.\n
c: Planetary days will become shorter.\n
d: Planetary gravity will become stronger.\n\n

You are a Refiner Agent to provide a refined step-by-step plan for solving the given question.\n
You are provided with:\n
(1) latent-format information: a previous plan with feedback\n
(2) text-format information: the input question you need to solve.\n\n

Based on the input, write a refined and improved plan to solve the question. Make sure your output plan is correct and concise.\n\n

Now, output your refined plan below:\n
<|im_end|>\n
<|im_start|>assistant

(c) Late layer (layer 35).

Figure 24: Headwise token selection for the Refiner agent.