DUETKV: COMPLEMENTING EXACT KV RETENTION WITH PROGRESSIVE EPISODIC LATENT TRACES

Anonymous authors
Paper under double-blind review

ABSTRACT

Query-agnostic key–value (KV) cache compression reduces the cache once for reuse across future queries, yet accuracy degrades sharply under tight budgets. Selective eviction preserves the original KV states but retains only a sparse token-aligned subset, while KV compaction constructs new representations at the cost of precision. Recent complementary approaches combine both, yet derive the synthetic component from the retention outcome or only after full context prefill. This leaves open how to construct complementary representations progressively from complete pre-eviction states while preserving a fixed KV budget. We introduce **DuetKV**, which pairs sparse exact retention with progressively constructed *episodic latent traces* under a fixed KV budget. For each context segment, a learned constructor synthesizes traces from its complete pre-eviction hidden states without conditioning on the current retention outcome. Frozen backbone projections map these traces into native KV states that participate alongside retained exact states in standard attention. The constructor is trained offline by self-distillation from a full-cache teacher, while the backbone and base eviction policy remain frozen. We evaluate DuetKV across 4 long-context backbones, 3 base eviction policies, and 12 prefill-intensive tasks. Across policy-budget settings, DuetKV improves the task-averaged score in 17 of 18 cases and recovers Full-KV performance with Fast KVzip at $5\times$ compression. On 170K-token Retr.KV with Qwen2.5-14B-Instruct-1M, it raises Fast KVzip from 31.4 to 80.2 at the same 5.7 GiB peak KV memory with only 1.7% additional prefill latency. The same constructor also improves decoding-intensive reasoning under continuous KV compression without decoding-specific retraining.

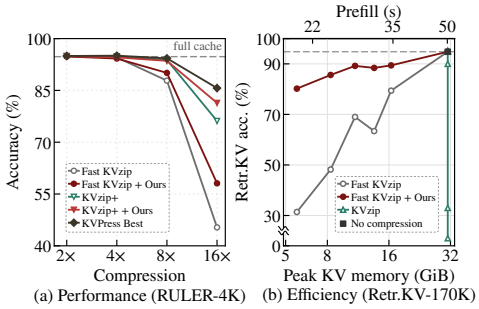
1 INTRODUCTION

Long-context inference relies on key–value (KV) caching to avoid recomputing past attention states, yet its memory footprint grows linearly with sequence length and becomes a primary constraint at scale (Zhang et al., 2023; Xiao et al., 2024; Li et al., 2024). KV eviction reduces this cost by retaining only a budgeted subset of cached states, and successive importance estimators have refined which states to preserve (Zhang et al., 2023; Li et al., 2024; Cai et al., 2025b; Feng et al., 2025). These estimators may score states against a specific downstream query or from the context alone. We focus on the latter, query-agnostic setting, in which a context is compressed once into a cache reusable across subsequent requests (Chen et al., 2026; Kim et al., 2026a;b). Under aggressive compression, however, even strong eviction policies degrade sharply: Fast KVzip falls from 94.6 at $4\times$ compression to 45.3 at $16\times$ on RULER-4K (Figure 1). Selection-only compression is also structurally constrained: improving the selector changes the retained subset, but cannot extend the compressed representation beyond the original token-aligned states.

KV cache compaction lifts this constraint by synthesizing compact states beyond the original ones, at the cost of departing from an entirely exact, token-aligned cache (Zhang et al., 2025; Chari et al., 2025; Li et al., 2026; O'Neill et al., 2026). Complementary formulations instead pair sparse exact retention with compact synthetic states (Zhan et al., 2026; Yang et al., 2026; Baek et al., 2026). Despite this difference, the complement is conditioned on the realized retention outcome or the fully assembled context, rather than built progressively from the representation that precedes each retention decision.

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.

Fuzzy-trace theory posits that verbatim and gist representations are encoded separately yet in parallel, with gist formed directly rather than derived from verbatim memory (Reyna & Brainerd, 1995; Reyna, 2012). Drawing on this principle, we introduce **DuetKV**, a query-agnostic KV compression framework that complements sparse exact retention with progressively constructed *episodic latent traces* under a single KV budget, formed in parallel with exact retention and without conditioning on its contemporaneous outcome. Concretely, for each context segment, the base eviction policy retains a budgeted subset of its KV states, while a learned *latent trace constructor* synthesizes a fixed number of compact traces from the segment's complete pre-eviction hidden states via cross-attention with learnable latent queries. Mapped into the backbone's KV space by its frozen projections, the latent traces are incorporated alongside retained states into the cache, allowing representations at distinct granularities to participate jointly in standard attention. Retention decoupling leaves the base eviction policy intact, and segment-wise construction requires no separate post-prefill pass.



(a) Performance (RULER-4K) (b) Efficiency (Retr.KV-170K)

Figure 1: **Highlighted results. (a)** RULER-4K performance across KV compression ratios on the KVPress benchmark (Devoto et al., 2025) with Qwen3-8B. *KVzip+*+Ours* denotes KVzip+ with the DuetKV constructor transferred from FastKVzip without retraining. **(b)** Accuracy–memory–latency trade-off for 170K-token Retr.KV with Qwen2.5-14B-Instruct-1M in BF16.

DuetKV is trained offline by self-distillation from a frozen full-cache teacher, whose continuations provide the target sequences without human annotation. Given the same context and query, the student runs under a budget-matched cache and is trained to match the teacher's answer-token distributions, updating only the latent trace constructor while keeping the backbone and the base eviction policy frozen. We evaluate DuetKV across 4 long-context backbones, 3 base eviction policies, and cache budgets from 10% to 75%, covering 12 prefill-intensive tasks on SCBENCH (Li et al., 2025) and MRCR (OpenAI, 2025) with contexts up to 170K tokens, and decoding-intensive reasoning on AIME24 (AIME, 2025) and MATH-500 (Hendrycks et al., 2021). On Qwen2.5-7B-Instruct-1M, DuetKV improves the average performance of Fast KVzip, Expected Attention (Devoto et al., 2025), and DuoAttention (Xiao et al., 2025) across 70 valid task–budget cells. At a 10% KV budget ($\rho = 0.1$), DuetKV raises Fast KVzip from 76% to 95% of Full-KV performance on Qwen2.5-14B-Instruct-1M. On Qwen2.5-7B-Instruct-1M, it matches Full-KV performance on average at $\rho = 0.2$ and improves 170K-token Retr.KV from 32.6 to 77.2 at $\rho = 0.1$, with only 1.3–2.5$\%$ additional prefill latency. DuetKV also improves decoding-intensive reasoning, gaining $+13.3$ points on AIME24 and $+6.4$ on MATH-500 at a 0.5K KV budget.

We summarize our contributions as follows:

- We introduce a complementary KV construction paradigm in which latent construction proceeds incrementally alongside eviction, without conditioning on the contemporaneous KV selection outcome or modifying the base eviction policy.
- We instantiate this paradigm in **DuetKV**, a query-agnostic KV compression framework that pairs sparse exact KV retention with progressively constructed *episodic latent traces* under a fixed total KV budget, without awaiting full-context cache materialization.
- We validate DuetKV across four long-context backbones and three base eviction policies in both prefill-intensive and decoding-intensive regimes, transferring the prefill-trained constructor to decoding without adaptation. DuetKV exhibits pronounced gains under aggressive compression; controlled ablations attribute these gains to learned trace content rather than latent cache allocation alone.

2 Related Work

Selective KV Retention. Most KV-cache compression methods retain a subset of the original token-aligned states, differing in how entries are scored, budgets are allocated and retention units are defined. H_2O (Zhang et al., 2023), SnapKV (Li et al., 2024) and ReST-KV (An et al., 2026) estimate

importance using attention statistics or output reconstruction, while PyramidKV (Cai et al., 2025b), Ada-KV (Feng et al., 2026) and ChunkKV (Liu et al., 2025) refine budget allocation or retention granularity. Methods also differ in their access to downstream information. LookaheadKV (Ahn et al., 2026) anticipates response-induced importance, whereas query-agnostic methods construct reusable caches before future queries are known. Among the latter, KVzip (Kim et al., 2026b), ContrastKV (Chen et al., 2026) and FastKVzip (Kim et al., 2026a) use context reconstruction, contrastive signals and learned gating, respectively. Expected Attention (Devoto et al., 2025) and LKV (Zhou et al., 2026) further explore anticipated attention and learned importance or budget allocation. Despite these advances, selection-based methods remain constrained to retaining original KV entries.

KV Compaction Beyond Exact Retention. Another line of work changes the representations stored in the cache rather than only selecting which original entries retent. KV-Distill (Chari et al., 2025) adapts selected representations to incorporate information from unselected tokens, while VECTOR (Lin et al., 2026) reconstructs values for approximated entries. Cartridges (Eyuboglu et al., 2026), Attention Matching (Zweiger et al., 2026) and Latent Context Compilation (Li et al., 2026) construct compact context representations through different forms of distillation or attention approximation. Still (O'Neill et al., 2026) amortizes KV synthesis through learned compactors, and Activation Beacon (Zhang et al., 2025) progressively compresses activations as context units are processed. These approaches establish representation synthesis as an alternative to exact-state selection. DuetKV instead studies how synthesized and exact KV states can coexist within a shared budget during progressive context processing.

Complementary KV Representations. Closer to DuetKV, complementary approaches preserve exact KV information alongside compact representations. Related forms of coexistence also appear in cross-layer merging and head-wise compensation(Liu et al., 2024; Tang et al., 2025). ResKV (Zhan et al., 2026) reconstructs omitted attention contributions with compact residual entries, while IndexMem (Yang et al., 2026) recurrently compresses evicted states into latent memory accessed through residual readouts. GraceKV (Tian et al., 2026) organizes token-level entries and compressed prototypes within a shared multi-resolution budget. RestoreKV (Baek et al., 2026) preserves the base eviction rule and constructs a complementary restore cache, but does so through an additional pass after full-context prefill.

3 METHODOLOGY

DuetKV represents a compressed context using two complementary components: exact KV states retained by a base eviction policy and learned latent traces synthesized from the context before eviction, as illustrated in Fig. 2. The two components share a fixed KV budget and are constructed progressively as the context is processed. We first describe latent construction and its integration into the cache, then introduce the offline distillation objective.

3.1 QUERY-AGNOSTIC KV COMPRESSION

We consider query-agnostic KV compression for a decoder-only Transformer with L layers, H_{kv} KV heads per layer, and hidden width D. A reusable cache is constructed from a compressible context $\mathcal{X}$ before any downstream query is observed. We partition $\mathcal{X}$ into segments $\{\mathcal{X}_1, \ldots, \mathcal{X}_N\}$, where $\mathcal{X}_j$ contains C_j tokens. At layer l, let $\mathbf{H}_j^{(l)} \in \mathbb{R}^{C_j \times D}$ denote the normalized pre-attention hidden states of segment $\mathcal{X}_j$ before eviction.

After processing segment $\mathcal{X}_j$, a trailing local window $\mathcal{W}_j$ remains protected from eviction. The positions newly exposed to eviction form the pruning slice:

$$\mathcal{P}_j = (\mathcal{X}_{\leq j} \setminus \mathcal{W}_j) \setminus (\mathcal{X}_{\leq j-1} \setminus \mathcal{W}_{j-1}), \tag{1}$$

where $\mathcal{X}_{\leq j}$ denotes the context prefix through segment j.

Base eviction policy. Let $\pi = (\sigma_\pi, \mathcal{A}_\pi, \mathcal{S}_\pi)$ denote a base eviction policy, where σ_π assigns importance scores to candidate KV entries, $\mathcal{A}_\pi$ allocates the available cache budget, and $\mathcal{S}_\pi$ selects the entries to retain. The budget may be shared across all compressed layers, allocated separately to each layer, or allocated to each KV head within each layer. We denote the resulting budget groups by $\{\mathcal{G}_g^\pi\}_{g=1}^{G_\pi}$, where $G_\pi \in \{1, L_c, L_c H_{\mathrm{kv}}\}$ and L_c is the number of compressed layers.

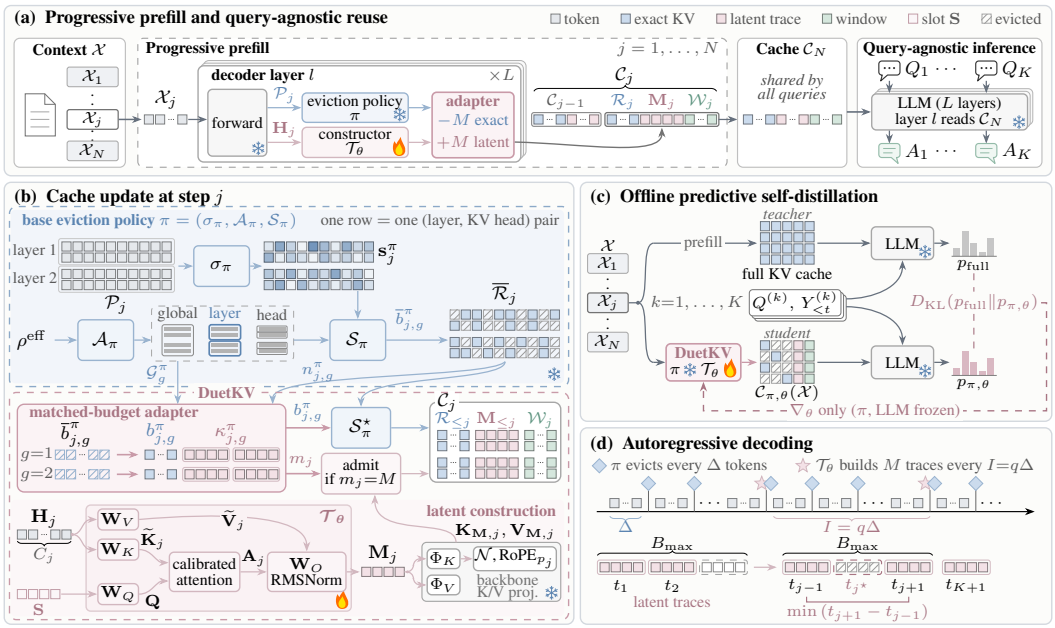


Figure 2: **Overview of DuetKV. (a)** Progressive prefill jointly maintains retained exact KV states and episodic latent traces, producing a reusable query-agnostic cache. **(b)** At each cache update, traces are synthesized from complete pre-eviction hidden states independently of the current retention outcome, then replace part of the exact-KV allocation under the same cache budget. **(c)** The constructor is trained offline by self-distillation from a full-cache teacher while the backbone and base evictor remain frozen. **(d)** The prefill-trained constructor is reused during autoregressive decoding, with latent capacity capped at $B_{\max}$ by strided retention.

At compression step j, the base policy assigns importance scores $\mathbf{s}_j^\pi$ to the candidate KV entries and determines the exact entries it would retain under the target cache ratio ρ:

$$\overline{\mathcal{R}}_j = \mathcal{S}_\pi\left(\mathbf{s}_j^\pi; \rho^{\text{eff}}\right), \tag{2}$$

where ρ^{eff} is the corresponding retention ratio over evictable states, with selection performed under the allocation induced by $\mathcal{A}_\pi$. The number of retained entries in budget group g is $\overline{b}_{j,g}^\pi = |\overline{\mathcal{R}}_j \cap \mathcal{G}_g^\pi|$, and the total is $\overline{b}_j^\pi = \sum_{g=1}^{G_\pi} \overline{b}_{j,g}^\pi$.

DuetKV uses part of this exact-KV allocation for latent traces without changing the base policy's importance estimator or allocation constraints. Importantly, $\overline{\mathcal{R}}_j$ is defined along the current DuetKV execution, rather than a separate base-only run. That is to say, traces admitted at earlier steps may affect subsequent hidden states and importance scores.

3.2 Progressive Latent Construction

Episodic latent trace construction. Given the base policy's exact-KV allocation, DuetKV constructs complementary latent traces from the complete pre-eviction hidden states, independently of the current retention outcome. For each prefill segment $\mathcal{X}_j$, DuetKV constructs M episodic latent traces at each compressed layer from the normalized pre-attention representation $\mathbf{H}_j^{(l)}$ without access to the contemporaneous retention outcome or any selector-derived state. At layer l, M learnable slots $\mathbf{S}^{(l)} \in \mathbb{R}^{M \times D}$ generate the constructor queries:

$$\mathbf{Q}^{(l)} = \mathbf{S}^{(l)}\mathbf{W}_Q^{(l)}, \qquad \widetilde{\mathbf{K}}_j^{(l)} = \mathbf{H}_j^{(l)}\mathbf{W}_K^{(l)}, \qquad \widetilde{\mathbf{V}}_j^{(l)} = \mathbf{H}_j^{(l)}\mathbf{W}_V^{(l)}, \tag{3}$$

where $\mathbf{W}_Q^{(l)}, \mathbf{W}_K^{(l)}, \mathbf{W}_V^{(l)} \in \mathbb{R}^{D \times d_m}$ and $d_m \ll D$. The resulting constructor logits are

$$\mathbf{U}_j^{(l)} = \frac{\mathbf{Q}^{(l)}\left(\widetilde{\mathbf{K}}_j^{(l)}\right)^\top}{\sqrt{d_m}} \in \mathbb{R}^{M \times C_j}. \tag{4}$$

We calibrate the constructor logits by removing per-slot offsets and estimating a shared segment-level scale:

$$\overline{\mathbf{U}}_j^{(l)} = \mathbf{U}_j^{(l)} - \frac{1}{C_j}\mathbf{U}_j^{(l)}\mathbf{1}\mathbf{1}^\top, \qquad \nu_j^{(l)} = \frac{\left\|\overline{\mathbf{U}}_j^{(l)}\right\|_F}{\sqrt{MC_j}}. \tag{5}$$

Here, $\mathbf{1} \in \mathbb{R}^{C_j}$ is the all-ones vector. Since row-wise softmax is invariant to additive shifts, centering removes per-slot offsets from the shared scale estimate. The calibrated attention weights are

$$\mathbf{A}_j^{(l)} = \mathrm{softmax}\left(\gamma \frac{\overline{\mathbf{U}}_j^{(l)}}{\max\left(\nu_j^{(l)}, \epsilon\right)}\right), \tag{6}$$

where the softmax is over token positions, γ is a fixed scalar that sets the logit scale, and $\epsilon > 0$ is a small numerical floor. The attention weights produce M latent traces:

$$\mathbf{M}_j^{(l)} = \mathrm{RMSNorm}_{\boldsymbol{\omega}^{(l)}}\left(\mathbf{A}_j^{(l)}\widetilde{\mathbf{V}}_j^{(l)}\mathbf{W}_O^{(l)}\right) \in \mathbb{R}^{M\times D}, \tag{7}$$

where $\mathbf{W}_O^{(l)} \in \mathbb{R}^{d_m \times D}$ and $\mathrm{RMSNorm}_{\boldsymbol{\omega}^{(l)}}$ denotes RMS normalization with learnable feature-wise scale $\boldsymbol{\omega}^{(l)} \in \mathbb{R}^D$, initialized from the corresponding pre-attention normalization weight.

We map the hidden-space traces into native KV entries using the backbone's frozen projections:

$$\mathbf{K}_{\mathbf{M},j}^{(l)} = \mathrm{RoPE}_{p_j}\left(\mathcal{N}^{(l)}\left(\Phi_K^{(l)}\left(\mathbf{M}_j^{(l)}\right)\right)\right), \qquad \mathbf{V}_{\mathbf{M},j}^{(l)} = \Phi_V^{(l)}\left(\mathbf{M}_j^{(l)}\right), \tag{8}$$

where $\Phi_K^{(l)}$ and $\Phi_V^{(l)}$ are the frozen backbone K/V projections, $\mathcal{N}^{(l)}$ denotes the backbone's key normalization when present, and p_j is the common segment-final positional anchor assigned to traces constructed from $\mathcal{X}_j$.

Matched-budget integration. The latent traces can participate in native attention, but storing them alongside all exact KV entries retained by the base policy would increase the cache size. DuetKV therefore replaces a portion of the exact entries with latent ones under the same budget. At compression step j, DuetKV allocates $m_j \in \{0, M\}$ latent traces to each (layer, KV head) pair from which the base policy evicts at least one exact KV entry. Let $n_{j,g}^\pi$ denote the number of such pairs in native budget group g. The resulting latent and residual exact-KV capacities are

$$\kappa_{j,g}^\pi = m_j n_{j,g}^\pi, \qquad b_{j,g}^\pi = \overline{b}_{j,g}^\pi - \kappa_{j,g}^\pi, \qquad g = 1, \ldots, G_\pi. \tag{9}$$

The total latent capacity is therefore

$$\kappa_j = \sum_{g=1}^{G_\pi} \kappa_{j,g}^\pi = m_j n_j, \qquad n_j = \sum_{g=1}^{G_\pi} n_{j,g}^\pi \le L_c H_{\mathrm{kv}}, \tag{10}$$

with equality when eviction spans all layer–head pairs at step j.

Trace admission requires sufficient capacity in every native budget group while limiting the fraction of the realized budget allocated to latent states:

$$m_j = M\,\mathbb{I}\left[\overline{b}_{j,g}^\pi \ge M n_{j,g}^\pi \;\; \forall g \;\wedge\; M n_j \le \eta_{\max}\overline{b}_j^\pi\right], \tag{11}$$

where $\mathbb{I}[\cdot]$ is the indicator function. We set $\eta_{\max} = 0.40$ throughout. If either condition fails, $m_j = 0$, leaving the native retention set unchanged.

When traces are admitted, the native retention set is restricted to the corresponding residual exact capacities:

$$\mathcal{R}_j = \mathcal{S}_\pi^\star\left(\overline{\mathcal{R}}_j; \left\{b_{j,g}^\pi\right\}_{g=1}^{G_\pi}\right) \subseteq \overline{\mathcal{R}}_j, \tag{12}$$

where $\mathcal{S}_\pi^\star$ retains, within each allocation group, the $b_{j,g}^\pi$ entries with the highest effective scores used by the native rule for selection, with residual ties resolved deterministically by within-group position. This cardinality-controlled restriction preserves all strict score orderings within $\overline{\mathcal{R}}_j$ and never admits an entry rejected by the native rule. Because $\kappa_{j,g}^\pi$ is deducted from $\overline{b}_{j,g}^\pi$, the admitted traces replace exact entries at equal cost, an exchange that reduces Bayes-optimal predictive distortion precisely when, given the remaining cache and query, the traces carry more predictive information than the displaced entries (Appendix A):

$$\left|\mathcal{R}_j \cap \mathcal{G}_g^\pi\right| + \kappa_{j,g}^\pi = \overline{b}_{j,g}^\pi \quad \forall g, \qquad |\mathcal{R}_j| + \kappa_j = \overline{b}_j^\pi. \tag{13}$$

Progressive cache update. Let $(\mathbf{K}^{(l)}_{\mathcal{R},j}, \mathbf{V}^{(l)}_{\mathcal{R},j})$ denote the exact K/V states indexed by $\mathcal{R}_j$, and let $(\mathbf{K}^{(l)}_{\mathcal{W},j}, \mathbf{V}^{(l)}_{\mathcal{W},j})$ denote those in the protected window $\mathcal{W}_j$. Using sequence-wise concatenation $\oplus$, define

$$\mathbf{K}^{(l)}_{\mathcal{R},\le j} = \bigoplus_{i=1}^{j} \mathbf{K}^{(l)}_{\mathcal{R},i}, \qquad \mathbf{K}^{(l)}_{\mathbf{M},\le j} = \bigoplus_{i=1}^{j} \mathbf{K}^{(l)}_{\mathbf{M},i}, \tag{14}$$

with analogous definitions for the values. After processing $\mathcal{X}_j$, the layer-l resident cache is

$$\mathcal{C}^{(l)}_j = \left(\mathbf{K}^{(l)}_{\mathcal{R},\le j} \oplus \mathbf{K}^{(l)}_{\mathbf{M},\le j} \oplus \mathbf{K}^{(l)}_{\mathcal{W},j},\ \mathbf{V}^{(l)}_{\mathcal{R},\le j} \oplus \mathbf{V}^{(l)}_{\mathbf{M},\le j} \oplus \mathbf{V}^{(l)}_{\mathcal{W},j}\right). \tag{15}$$

The episodic states derived from $\mathcal{X}_j$ are admitted only after the forward pass over $\mathcal{X}_j$ and participate in attention from $\mathcal{X}_{j+1}$ onward alongside retained exact states, preserving causal ordering while conditioning the representations from which later traces are formed.

3.3 Offline Self-Distillation from the Full Cache

For a context $\mathcal{X}$ associated with K training queries $\{Q^{(k)}\}_{k=1}^{K}$, we obtain fixed teacher continuations $Y^{(k)} = (y^{(k)}_1, \dots, y^{(k)}_{T_k})$ by offline greedy decoding from the full-cache model and evaluate them with both the full-cache teacher and compressed-cache student under teacher forcing. At continuation step $t \in \{1, \dots, T_k\}$, define

$$p^{(k,t)}_{\text{full}}(\cdot) = p\Big(\cdot \mid \mathcal{X}, Q^{(k)}, Y^{(k)}_{<t}\Big), \qquad p^{(k,t)}_{\pi,\theta}(\cdot) = p\Big(\cdot \mid \mathcal{C}_{\pi,\theta}(\mathcal{X}), Q^{(k)}, Y^{(k)}_{<t}\Big), \tag{16}$$

where $\mathcal{C}_{\pi,\theta}(\mathcal{X})$ denotes the compressed prompt cache produced by the progressive construction in §3.2. We optimize the constructor by minimizing the forward KL divergence, averaged over continuation steps within each query and then across queries:

$$\mathcal{L}(\theta) = \frac{1}{K}\sum_{k=1}^{K}\frac{1}{T_k}\sum_{t=1}^{T_k} D_{\text{KL}}\Big(p^{(k,t)}_{\text{full}} \,\Big\|\, p^{(k,t)}_{\pi,\theta}\Big). \tag{17}$$

The compressed cache is constructed once from $\mathcal{X}$ alone and reused across all associated queries; neither the queries nor their teacher continuations enter its construction. Only θ is optimized, using downstream predictive supervision rather than direct reconstruction of evicted K/V states; the backbone is frozen and the paired base eviction policy π is fixed throughout training. Training preserves the progressive forward dependencies in §3.2 while truncating gradients across segment boundaries; the implementation and validation of this approximation are provided in Appendix B.4.

3.4 Autoregressive Decoding

During autoregressive decoding, DuetKV reuses the prefill-trained constructor and frozen K/V projections without decoding-specific adaptation. Given the compression interval Δ of the paired Fast KVzip policy, trace construction is performed every $I = q\Delta$ generated tokens, $q \in \mathbb{N}$. Under the matched cache budget, latent KV states are capped at $B_{\max}$ per (layer, KV head) pair, beyond which strided retention evicts one interior group per overflow, selecting the most recent one at the first overflow when the groups are equally spaced (Appendix B.5).

4 Experiments

4.1 Experimental Setup

Datasets. Evaluation spans prefill- and decoding-intensive regimes. For prefill-intensive evaluation, we follow FastKVzip (Kim et al., 2026a) with a 12-task suite comprising nine SCBench tasks (Li et al., 2025) spanning synthetic retrieval from RULER (Hsieh et al., 2024), long-context understanding from LongBench (Bai et al., 2024), and code comprehension, complemented by OpenAI MRCR (OpenAI, 2025), SQuAD (Rajpurkar et al., 2016), and GSM8K (Cobbe et al., 2021). Contexts extend to 170K Qwen3 tokens and are evaluated query agnostically: cache construction precedes access to downstream queries, with a single compressed cache reused whenever multiple queries share the same context. For decoding-intensive evaluation, we consider AIME24 (AIME, 2025) and MATH-500 (Hendrycks et al., 2021) under the continuous-compression protocol of R-KV (Cai et al., 2025a), with generation capped at 32K and 16K tokens, respectively.

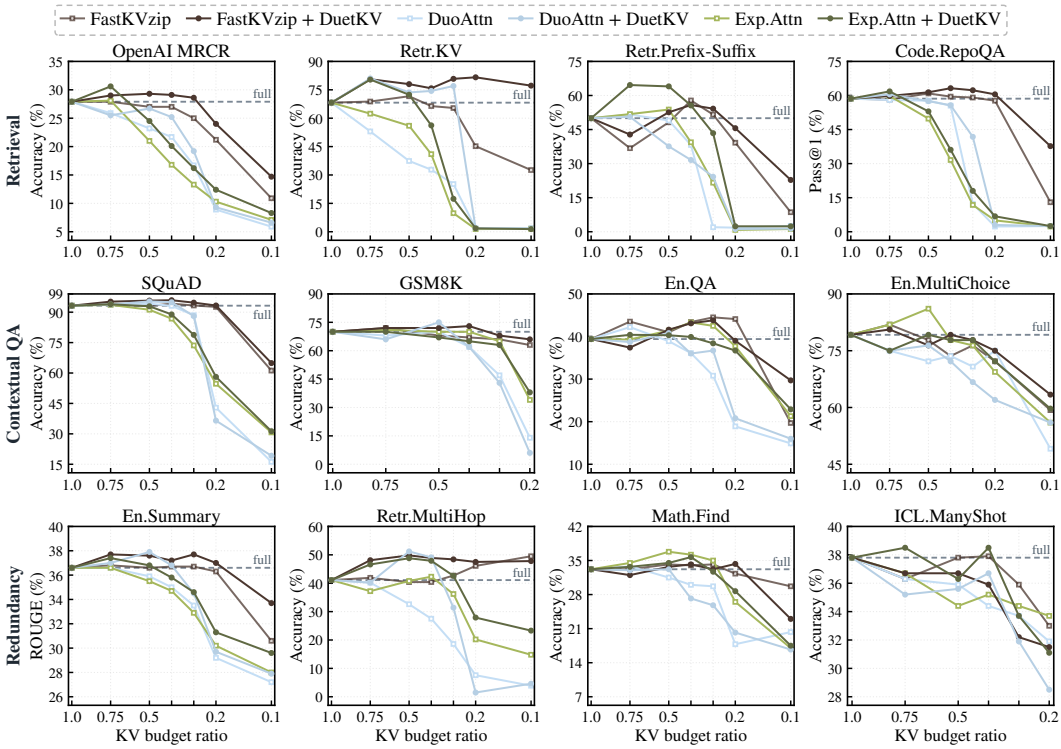


Figure 3: **Matched-budget evaluation across three base evictors and 12 prefill-intensive tasks.** We evaluate Fast KVzip, DuoAttention and Expected Attention with and without DuetKV on the same 70 task–budget cells. Open squares denote the base evictors, filled circles denote their DuetKV-augmented counterparts, and horizontal dashed lines denote Full-KV performance.

Baselines. For prefill-intensive evaluation, we compare DuetKV with Full KV, KVzip (Kim et al., 2026b), SnapKV (Li et al., 2024), Expected Attention (Devoto et al., 2025) coupled with Ada-KV allocation (Feng et al., 2026), DuoAttention (Xiao et al., 2025) using KVzip importance scores, and Fast KVzip (Kim et al., 2026a). For decoding evaluation, we additionally compare against R-KV (Cai et al., 2025a). Unless specified, DuetKV adopts Fast KVzip as its paired importance estimator, with latent-trace capacity deducted from exact retention under a matched total KV budget.

Training details. We train and evaluate DuetKV on Qwen2.5-7B/14B-Instruct-1M (Yang et al., 2025b) and Qwen3-8B (Yang et al., 2025a), and evaluate the Qwen3-8B constructor on Qwen3-8B-FP8 without retraining. We construct a long-context distillation corpus of 1,590 contexts from FineWeb-Edu (Penedo et al., 2024), PG-19 (Rae et al., 2020), NarrativeQA (Kočiskỳ et al., 2018) and GitHub source code, pairing each context with $K = 5$ queries. A content-level scan detects no contamination across the twelve evaluation tasks. Unless specified, we set $M = 4$, $d_m = 128$ and $\gamma = 3$, and process prefill in 16K-token segments with a 4K protected local window. On Qwen2.5-7B-Instruct-1M, the constructor adds 51.9M trainable parameters (0.68% of the frozen 7.62B backbone). Each constructor is trained only at $\rho = 0.1$ and evaluated without retraining at $\rho \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.75\}$. Corpus construction, decontamination, optimization and additional implementation details are provided in Appendix B.

4.2 Main Results

Generalization across evictors and backbones. Figure 3 shows that DuetKV improves Fast KVzip, Expected Attention, and DuoAttention by 3.8, 2.9, and 3.4 points+ on average, respectively, across twelve prefill-intensive tasks and six cache budgets, with the largest gains on retrieval-intensive tasks. For the default Fast KVzip pairing, DuetKV matches Full KV on average at $\rho = 0.2$, while Fast KVzip is already -4.4 points lower; at $\rho = 0.1$, it recovers roughly half of Fast KVzip's 21.2-point deficit. Figure 4 further demonstrates that the improvement over Fast KVzip extends

beyond the primary Qwen2.5-7B-1M backbone: DuetKV outperforms Fast KVzip on Qwen2.5-14B-1M and Qwen3-8B/8B-FP8 across most evaluated cache budgets.

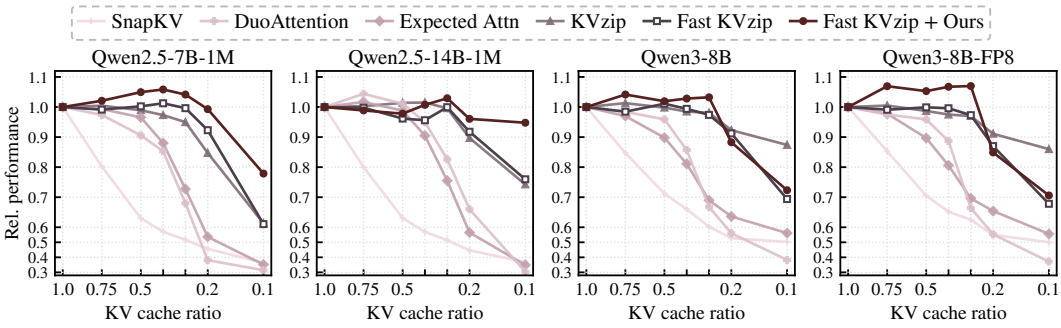


Figure 4: **Performance across backbones.** With Fast KVzip as the base evictor, results are normalized to full-cache performance within each task and averaged over valid tasks at each cache ratio. The Qwen3-8B constructor is transferred to Qwen3-8B-FP8 without retraining.

Decoding-intensive performance. Figure 5 evaluates DuetKV when KV compression continues throughout autoregressive decoding rather than ending after prefill. At larger budgets, both methods approach Full-KV performance, whereas under tighter budgets DuetKV remains substantially more accurate than Fast KVzip, improving AIME24 from 12.1 to 25.4 (+13.3 points) and MATH-500 from 79.4 to 85.8 (+6.4 points) at a 512-token KV budget. These results show that the prefill-trained constructor remains effective under the distinct cache dynamics of autoregressive decoding. *Early Stop* is omitted at 0.5K as some prompts exceed this length.

4.3 Ablation Studies

Table 1 evaluates DuetKV on Qwen2.5-7B-Instruct-1M with Fast KVzip as the base evictor. All variants except DuetKV+ preserve the total KV-cache budget of the paired Fast KVzip baseline.

Predictive contribution of the latent traces. Two complementary controls probe the two sides of the matched-capacity exchange (Theorem A.2). *Random* replaces each episodic latent trace with a norm-matched Gaussian vector while preserving its cache cost and frozen K/V projections, reducing the average score from 52.9 to 48.2; at $\rho = 0.1$, the retrieval gain over Fast KVzip falls from +21.9 to +1.9 points. DuetKV+ instead retains the learned traces while restoring the base eviction policy's full exact-KV allocation, but raises the average by only 0.3 points, from 52.9 to 53.2. The gain therefore depends primarily on learned trace content, while restoring the displaced exact-KV capacity provides little additional benefit.

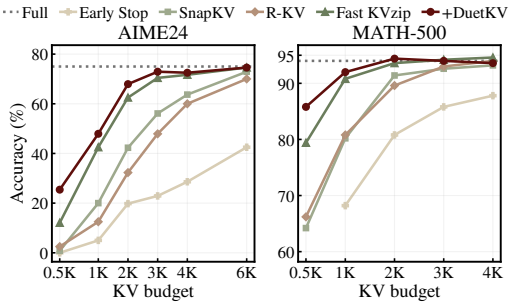


Figure 5: **Decoding-intensive performance.** Accuracy on AIME24 and MATH-500 across cache budgets with Qwen3-8B. AIME24 results are averaged over 16 random seeds.

Latent trace construction. We ablate three aspects of the constructor: its input support, trace cardinality, and attention calibration. The *Evicted* variant restricts its inputs to positions discarded by at least one KV head under Fast KVzip, which at $\rho = 0.1$ excludes only 2.4% of segment positions, and reduces the average score from 52.9 to 51.9; at $\rho = 0.1$, retrieval falls from 38.1 to 35.7. Because the restriction also broadens the learned aggregation, this result does not isolate the effect of input support (Appendix C.2). Under a fixed total cache budget, $M = 2, 4, 8$ yield average scores of 51.8, 52.9, and 51.5, respectively, showing that increasing trace cardinality is not monotonically beneficial as additional traces displace exact KV states. Finally, the *Uncalib.* variant removes logit calibration from the constructor cross-attention, reducing the average from 52.9 to

Table 1: Per-budget attribution and design ablations on Qwen2.5-7B-Instruct-1M with FastKVzip as the base policy. DuetKV+ retains the full exact-KV allocation and additionally stores the learned latent states.

	ρ	Baselines		Method and ablations						
		Full KV	FastKVzip	**DuetKV**	**DuetKV+**	Random	Evicted	Uncalib.	M=2	M=8
Retrieval	0.75	51.2	47.9	**53.0**$_{+5.1}$	**53.1**$_{+5.2}$	47.1$_{-0.8}$	49.4$_{+1.5}$	52.5$_{+4.6}$	53.8$_{+5.9}$	51.1$_{+3.2}$
	0.5	51.2	51.9	**55.3**$_{+3.4}$	**55.1**$_{+3.2}$	51.0$_{-0.9}$	55.9$_{+4.0}$	55.3$_{+3.4}$	54.4$_{+2.5}$	54.6$_{+2.7}$
	0.4	51.2	52.7	**56.0**$_{+3.3}$	**56.1**$_{+3.4}$	51.7$_{-1.0}$	57.0$_{+4.3}$	56.1$_{+3.4}$	55.1$_{+2.4}$	55.2$_{+2.5}$
	0.3	51.2	50.3	**56.5**$_{+6.2}$	**56.1**$_{+5.8}$	47.6$_{-2.7}$	56.1$_{+5.8}$	57.1$_{+6.9}$	56.8$_{+6.5}$	55.4$_{+5.1}$
	0.2	51.2	40.8	**52.9**$_{+12.1}$	**52.2**$_{+11.4}$	40.5$_{-0.3}$	52.3$_{+11.5}$	52.7$_{+11.8}$	52.8$_{+12.0}$	52.6$_{+11.8}$
	0.1	51.2	16.3	**38.1**$_{+21.9}$	**38.3**$_{+22.0}$	18.1$_{+1.9}$	35.7$_{+19.5}$	37.1$_{+20.8}$	37.4$_{+21.2}$	34.3$_{+18.1}$
Contextual QA	0.75	70.5	72.8	**71.3**$_{-1.6}$	**72.1**$_{-0.8}$	71.4$_{-1.4}$	72.0$_{-0.9}$	72.2$_{-0.6}$	69.3$_{-3.5}$	72.0$_{-0.8}$
	0.5	70.5	70.0	**71.4**$_{+1.4}$	**71.2**$_{+1.2}$	70.4$_{+0.4}$	71.0$_{+1.0}$	70.6$_{+0.6}$	70.5$_{+0.5}$	72.3$_{+2.3}$
	0.4	70.5	69.5	**72.8**$_{+3.3}$	**72.5**$_{+2.9}$	69.4$_{-0.1}$	70.6$_{+1.1}$	71.6$_{+2.1}$	69.3$_{-0.3}$	70.8$_{+1.2}$
	0.3	70.5	70.1	**71.1**$_{+1.0}$	**72.2**$_{+2.2}$	67.1$_{-3.0}$	69.7$_{-0.4}$	70.6$_{+0.5}$	67.1$_{-2.9}$	68.5$_{-1.6}$
	0.2	70.5	68.0	**68.3**$_{+0.3}$	**69.5**$_{+1.5}$	65.6$_{-2.4}$	67.9$_{-0.1}$	67.7$_{-0.3}$	66.7$_{-1.2}$	66.4$_{-1.6}$
	0.1	70.6	46.7	**52.7**$_{+5.9}$	**58.2**$_{+11.5}$	40.1$_{-6.6}$	51.6$_{+4.9}$	51.1$_{+4.4}$	53.5$_{+6.8}$	48.6$_{+1.9}$
Redundancy	0.75	37.2	37.0	**38.6**$_{+1.6}$	**38.5**$_{+1.5}$	38.1$_{+1.0}$	36.4$_{-0.7}$	37.3$_{+0.2}$	37.4$_{+0.4}$	37.4$_{+0.3}$
	0.5	37.2	37.2	**39.4**$_{+2.2}$	**39.2**$_{+2.0}$	37.1$_{-0.1}$	37.3$_{+0.1}$	37.2$_{-0.0}$	37.7$_{+0.4}$	37.1$_{-0.1}$
	0.4	37.2	37.5	**39.1**$_{+1.6}$	**39.1**$_{+1.6}$	36.2$_{-1.3}$	37.9$_{+0.4}$	37.0$_{-0.5}$	37.8$_{+0.3}$	37.2$_{-0.3}$
	0.3	37.2	37.4	**37.9**$_{+0.5}$	**37.8**$_{+0.5}$	37.8$_{+0.4}$	37.0$_{-0.3}$	36.5$_{-0.8}$	37.0$_{-0.4}$	37.4$_{+0.0}$
	0.2	37.2	36.9	**37.6**$_{+0.7}$	**37.7**$_{+0.8}$	37.2$_{+0.2}$	36.6$_{-0.3}$	35.7$_{-1.3}$	37.1$_{+0.2}$	36.7$_{-0.2}$
	0.1	37.0	36.6	**34.8**$_{-1.7}$	**35.5**$_{-1.1}$	37.1$_{+0.5}$	34.8$_{-1.8}$	33.5$_{-3.1}$	35.8$_{-0.8}$	35.1$_{-1.5}$
	Avg.	52.9	49.1	**52.9**$_{+3.8}$	**53.2**$_{+4.1}$	48.2$_{-0.8}$	51.9$_{+2.8}$	52.0$_{+3.0}$	51.8$_{+2.8}$	51.5$_{+2.5}$

52.0. We use $M = 4$ with calibrated logits throughout the main experiments. Appendix C.2 further characterizes the associated aggregation breadth and effective rank.

4.4 Efficiency

Figure 6 reports prefill and decoding latency against peak KV memory on a single H100. Because DuetKV replaces exact KV states with latent traces under the same cache budget, its peak KV memory remains essentially identical to that of the paired base evictor. At $\rho = 0.1$, trace construction adds only 1.3–2.5% prefill latency, while decoding latency changes by -0.4% to $+1.6\%$. DuetKV also constructs traces progressively, retaining only the compressed prefix and the in-flight segment rather than materializing the full prompt cache before compression. At 170K tokens and $\rho = 0.1$ on Qwen2.5-14B-Instruct-1M, KVzip requires 31.0 GiB peak KV memory and 165.6 s of prefill time, compared with 5.7 GiB and 19.1 s for DuetKV. Details are provided in Appendix B.2.

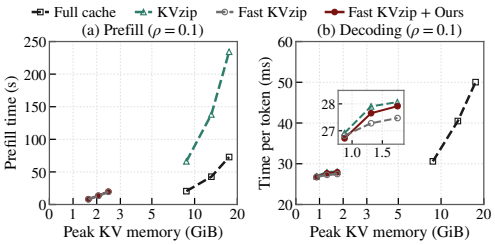


Figure 6: **Inference efficiency.** Prefill and decoding latency versus peak KV memory for Qwen2.5-7B-Instruct-1M at a 10% KV budget on a single H100 GPU. Points denote context lengths of 160K, 240K and 320K tokens.

5 Conclusion

We introduced DuetKV, a query-agnostic KV compression framework that pairs sparse exact retention with learned episodic latent traces. Constructed progressively from complete pre-eviction representations without conditioning on retention outcomes, the traces enter the backbone's native attention alongside retained exact states under a fixed total KV budget. Across multiple backbones and base eviction policies, and in both prefill- and decoding-intensive settings, DuetKV substantially improves performance under aggressive compression with modest inference overhead. Ablations further show that the gains arise from learned trace content rather than cache allocation alone, highlighting the benefit of combining exact retention with learned latent construction for KV cache compression.

AI USE STATEMENT

We used large language models (LLMs), including LLM-based coding assistants, to support code implementation and experimental execution, including the development of training and evaluation pipelines and the scheduling and monitoring of experimental runs. LLMs also assisted with drafting and polishing the manuscript and identifying relevant literature. All cited works and attributed claims were verified against their primary sources, and all AI-assisted code, experimental outputs and manuscript text were reviewed by the authors, who take full responsibility for the final content of the paper.

ETHICS STATEMENT

This work studies KV-cache compression for efficient long-context inference with pretrained language models. Our experiments use publicly available models and established benchmarks and involve no collection of human-subject or private user data. As a lossy inference-time transformation, DuetKV may alter model outputs relative to uncompressed inference, warranting careful evaluation in safety-critical or otherwise sensitive applications. Reducing inference-time memory requirements may lower deployment costs, but does not mitigate potential misuse, bias, or other risks associated with the underlying models.

REPRODUCIBILITY STATEMENT

The appendix provides all experimental details required for reproduction and complete proofs of the theoretical results. An anonymized implementation accompanies the submission; code, trained checkpoints, and evaluation outputs will be released upon publication.

REFERENCES

Jinwoo Ahn, Ingyu Seong, Akhil Kedia, Junhan Kim, Hyemi Jang, Kangwook Lee, and Yongkweon Jeon. LookaheadKV: Fast and accurate KV cache eviction by glimpsing into the future without generation. In *International Conference on Learning Representations*, 2026.

AIME. AIME problems and solutions, 2025. URL `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

Yongqi An, Chang Lu, Kuan Zhu, Tao Yu, Chaoyang Zhao, Hong Wu, Ming Tang, and Jinqiao Wang. ReST-KV: Robust KV cache eviction with layer-wise output reconstruction and spatial-temporal smoothing. In *International Conference on Learning Representations*, 2026.

Changwoo Baek, Seungjun Shin, and Kyeongbo Kong. Restorekv: Recovering full-cache behavior under aggressive query-agnostic kv cache eviction. *arXiv preprint arXiv:2608.01247*, 2026.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3119–3137. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.172.

Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-KV: Redundancy-aware KV cache compression for reasoning models. In *Advances in Neural Information Processing Systems*, volume 38, 2025a.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025b. URL `https://openreview.net/forum?id=ayi7qezU87`.

Vivek Chari, Guanghui Qin, and Benjamin Van Durme. Kv-distill: Nearly lossless learnable context compression for llms. *arXiv preprint arXiv:2503.10337*, 2025.

Xingchi Chen, Peiyuan Zong, Ziqiang Gao, Qing Li, Yong Jiang, Fa Zhu, and Hui Li. Contrastkv: Robust kv cache eviction via contrastive signal fusion for multi-query generalization. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9216–9229, 2026.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Atri Rudra, James Y Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study. In *International Conference on Learning Representations*, volume 2026, pp. 42642–42687, 2026.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-3773.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:113152–113188, 2026.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Jang-Hyun Kim, Dongyoon Han, and Sangdoo Yun. Fast kvzip: Efficient and accurate llm inference with gated kv eviction. *CoRR*, abs/2601.17668, January 2026a. URL `https://doi.org/10.48550/arXiv.2601.17668`.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W Lee, Sangdoo Yun, and Hyun Oh Song. Kvzip: Query-agnostic kv cache compression with context reconstruction. *Advances in Neural Information Processing Systems*, 38:167563–167591, 2026b.

Tomáš Kočiskỳ, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, et al. Scbench: A kv cache-centric analysis of long-context methods. In *International Conference on Learning Representations*, volume 2025, pp. 66063–66093, 2025.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Zeju Li, Yizhou Zhou, and Qiang Xu. Latent context compilation: Distilling long context into compact portable memory. In *Third Conference on Language Modeling*, 2026. URL `https://openreview.net/forum?id=svRq7IdBGi`.

Yuping Lin, Jiayuan Ding, Yue Xing, Pengfei He, Jiliang Tang, and Subhabrata Mukherjee. A simple plug-in for improving eviction-based kv cache compression. *arXiv preprint arXiv:2605.23258*, 2026.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031, 2024.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. ChunkKV: Semantic-preserving KV cache compression for efficient long-context LLM inference. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Charles O'Neill, Alex Sandomirsky, Harry Partridge, Mudith Jayasekara, and Max Kirkby. Still: Amortized kv cache compaction in a single forward pass. *arXiv preprint arXiv:2606.07878*, 2026.

OpenAI. Multi-round coreference resolution (MRCR). OpenAI long-context evaluation, 2025.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SylKikSYDH.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392. Association for Computational Linguistics, 2016. doi: 10.18653/v1/D16-1264.

Valerie F Reyna. A new intuitionism: Meaning, memory, and development in fuzzy-trace theory. *Judgment and Decision making*, 7(3):332–359, 2012.

Valerie F Reyna and Charles J Brainerd. Fuzzy-trace theory: An interim synthesis. *Learning and individual Differences*, 7(1):1–75, 1995.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Danning Ke, Shikuan Hong, Yiwu Yao, and Gongyi Wang. Razorattention: Efficient kv cache compression through retrieval heads. In *International Conference on Learning Representations*, volume 2025, pp. 16632–16646, 2025.

Haolin Tian, Yuzhe Liu, and Tonghan Wang. Every cache entry earns its place: Global allocation of resolution and coverage for kv cache compression. *arXiv preprint arXiv:2608.07001*, 2026.

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

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations*, volume 2025, pp. 37228–37253, 2025.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1m technical report. *CoRR*, abs/2501.15383, January 2025b. URL https://doi.org/10.48550/arXiv.2501.15383.

Xintong Yang, Hao Gu, Binxing Xu, Lujun Li, Bei Liu, Jiacheng Liu, Qiyuan Zhu, Yike Guo, and Sirui Han. Indexmem: Learned KV-cache eviction with latent memory for long-context LLM inference. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=SKXkGLbXdB.

Yuhang Zhan, Lisi Chen, and Shuo Shang. Reskv: Reconstructing omitted attention contributions for fixed-budget kv cache compression. *arXiv preprint arXiv:2607.29591*, 2026.

Peitian Zhang, Zheng Liu, Shitao Xiao, Ninglu Shao, Qiwei Ye, and Zhicheng Dou. Long context compression with activation beacon. In *International Conference on Learning Representations*, volume 2025, pp. 101932–101948, 2025.

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. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023. doi: 10.52202/075280-1506.

Enshuai Zhou, Yifan Hao, Chao Wang, Rui Zhang, Di Huang, Jiaming Guo, Xing Hu, Zidong Du, Qi Guo, and Yunji Chen. Lkv: End-to-end learning of head-wise budgets and token selection for llm kv cache eviction. *arXiv preprint arXiv:2605.06676*, 2026.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast KV compaction via attention matching. In *Forty-third International Conference on Machine Learning*, 2026. URL `https://openreview.net/forum?id=t01SKTj8pP`.

Appendix Overview

Appendix A provides a theoretical analysis of the matched-capacity formulation in §3. Appendix B provides experimental and implementation details for reproducibility, including the distillation data, optimization, efficiency measurements, the truncated-gradient implementation and its validation, and the decoding protocol. Appendix C extends the ablations in §4.3 with a sensitivity analysis of the protected-window size and an analysis of latent-trace construction, structure and readout.

A Theoretical Analysis

We analyze DuetKV through predictive distortion, matched-capacity exchange, latent expressivity, predictive self-distillation, and budget scaling: what a query-agnostic representation irreducibly loses, when exchanging exact states for latent ones is informationally favorable, whether the frozen interface can realize such a representation, how the training objective relates to that criterion, and how its allocation scales.

Let X denote the input context $\mathcal{X}$ of §3.1, viewed as a random variable, and U the information available at prediction time after query-agnostic cache construction. For downstream question answering, U contains the query. For autoregressive self-distillation at step t, we write $U_t = (Q, Y_{<t})$, where $Y_{<t}$ is the teacher-forced continuation prefix. Let $p_F(y \mid x, u)$ denote the conditional prediction distribution induced by the frozen full-cache model.

We represent a generic query-agnostic compressor as

$$Z = \mathcal{C}(X, \Xi), \tag{18}$$

where Ξ denotes any internal randomness of the compressor and satisfies $\Xi \perp (U, Y) \mid X$, a degenerate Ξ recovering the deterministic case to which DuetKV belongs. Together with the frozen full-cache model, the compressor induces the joint distribution

$$P_F(x, u, y, z) = P_{X,U}(x, u)\, p_{\mathcal{C}}(z \mid x)\, p_F(y \mid x, u). \tag{19}$$

Hence $p_{\mathcal{C}}(z \mid x, u) = p_{\mathcal{C}}(z \mid x)$ and

$$Y - (X, U) - Z. \tag{20}$$

All entropies, mutual informations, and expectations below are taken under the corresponding induced joint distribution. Cache capacity is measured in $(\text{layer}, \text{KV-head}, \text{position})$ cells, consistent with the operational accounting used by DuetKV. Let $\mathbb{L}_{\text{cmp}} \subseteq [L]$ denote the cache-compressed layers and $L_c = |\mathbb{L}_{\text{cmp}}|$; all exact–latent capacity exchanges occur only on these layers. Uncompressed layers are part of the conditioning state and are identical in the counterfactual representations compared below. Within each compressed layer, each KV head stores vectors of dimension d_h, so that a single position occupies one cell per $(\text{layer}, \text{KV head})$ pair. For a collection $\mathcal{A}$ of cache entries we write $\text{cost}_g(\mathcal{A})$ for the number of cells it occupies in allocation group g and $\text{cost}(\mathcal{A})$ for the total.

Notation. Table 2 summarizes the core notation used throughout the analysis, while result-specific symbols are defined where they first appear.

Table 2: Core notation used in the theoretical analysis.

Symbol	Definition
j	Prefill-segment index. The corresponding compression step follows the processing of $\mathcal{X}_j$.
X, U, Y	Input context; prediction-time information unavailable during cache construction; prediction target.
Z	Generic query-agnostic compressed representation of X.
p_F	Conditional prediction distribution induced by the frozen full-cache model.
$\mathcal{D}(Z, r)$, $\mathcal{D}^\star(Z)$	Population predictive distortion for a generic predictor $r(y \mid z, u)$ operating on Z, and its infimum over all such predictors, respectively.
H_F, I_F	Entropy and mutual information under the induced joint distribution.
$\mathbb{L}_{\mathrm{cmp}}$, L_c	Set of cache-compressed layers and its cardinality.
$\overline{\mathcal{R}}_j$, $\mathcal{R}_j$	Sets of exact cache cells, indexed by compressed layer, KV head, and position: those the base policy would retain before latent-capacity deduction, and the subset retained by DuetKV after it.
$\mathcal{E}_j$	Exact cache cells displaced by latent-capacity allocation: $\mathcal{E}_j = \overline{\mathcal{R}}_j \setminus \mathcal{R}_j$.
$\mathcal{M}_j$	Projected latent KV representation constructed from $\mathcal{X}_j$ and admitted after that segment is processed. Calligraphic $\mathcal{M}_j$ denotes the projected key–value states; the boldface $\mathbf{M}_j^{(l)}$ of §3.2 denotes the hidden-space traces from which they are obtained.
$\mathcal{C}_j^{\mathrm{sh}}$	Cache state shared by the two representations in the local exchange associated with segment j.
$Z_{\mathrm{sel},j}$, $Z_{\mathrm{hyb},j}$	Matched representations $(\mathcal{C}_j^{\mathrm{sh}}, \mathcal{E}_j)$ and $(\mathcal{C}_j^{\mathrm{sh}}, \mathcal{M}_j)$, respectively.
$n_{j,g}^{\pi}$, κ_j	Number of compressed (layer, KV head) pairs in group g that receive traces at segment j, and the cache-cell charge of the resulting trace group.
$\mathcal{G}_g^{\pi}$, $\overline{b}_{j,g}^{\pi}$	Allocation group g induced by the paired base policy π, and its realized native exact-KV cardinality before latent-capacity deduction.
ΔI_j	Conditional predictive information carried by $\mathcal{M}_j$ minus that carried by the exact entries it displaces, both given the shared state.

A.1 Predictive Distortion and Matched-Capacity Exchange

For a generic predictor $r(y \mid z, u)$ operating on a compressed representation, define its population predictive distortion relative to the full-cache model as

$$\mathcal{D}(Z, r) = \mathbb{E}\left[D_{\mathrm{KL}}\left(p_F(Y \mid X, U) \,\|\, r(Y \mid Z, U)\right)\right]. \tag{21}$$

Introducing the generic predictor separates the predictive information retained by Z from the particular readout used to exploit that representation.

Proposition A.1 (Irreducible predictive distortion)**.** *For any compressed representation Z satisfying Eq. equation 20,*

$$\mathcal{D}^\star(Z) := \inf_r \mathcal{D}(Z, r) = I_F(Y; X \mid Z, U), \tag{22}$$

where the infimum is attained by

$$r^\star(y \mid z, u) = p_F(y \mid z, u). \tag{23}$$

Proof. Introducing $p_F(Y \mid Z, U)$ into Eq. equation 21 gives

$$\begin{aligned}\mathcal{D}(Z, r) = {} & \mathbb{E} D_{\mathrm{KL}}\left(p_F(Y \mid X, U) \| p_F(Y \mid Z, U)\right) \\ & + \mathbb{E} D_{\mathrm{KL}}\left(p_F(Y \mid Z, U) \| r(Y \mid Z, U)\right).\end{aligned} \tag{24}$$

By Eq. equation 20, $p_F(Y \mid X, Z, U) = p_F(Y \mid X, U)$, so the first term equals $I_F(Y; X \mid Z, U)$. The second term is non-negative and vanishes for $r = p_F(Y \mid Z, U)$. □

Proposition A.1 identifies the representation-level loss of full-cache predictive fidelity. Under log loss, $I_F(Y; X \mid Z, U)$ is precisely the predictive dependence on the full context that remains unresolved by Z. This quantity is distinct from geometric K/V reconstruction error and, without additional assumptions linking teacher fidelity to task loss, does not by itself determine downstream accuracy.

We next instantiate this representation-level criterion for the local capacity exchange induced when DuetKV augments a frozen base evictor.

Policy-agnostic matched-capacity construction. We inherit the policy interface and budget adapter from §3.1–3.2. The paired base policy is $\pi = (\sigma_\pi, \mathcal{A}_\pi, \mathcal{S}_\pi)$, comprising its fixed importance estimator, allocation rule, and native selection rule. At the compression step associated with $\mathcal{X}_j$, applying the native rule before latent-capacity deduction yields the within-run counterfactual set of exact cache cells

$$\overline{\mathcal{R}}_j = \mathcal{S}_\pi\left(\mathbf{s}_j^\pi; \rho^{\text{eff}}\right), \qquad \overline{b}_{j,g}^\pi = \left|\overline{\mathcal{R}}_j \cap \mathcal{G}_g^\pi\right|. \tag{25}$$

Here $\{\mathcal{G}_g^\pi\}_{g=1}^{G_\pi}$ are the policy's native allocation groups, with $G_\pi \in \{1, L_c, L_c H_{\text{kv}}\}$ according to whether the policy allocates globally, per layer, or per (layer, KV head); the policies we pair with instantiate the first two. The counterfactual is evaluated on the realized DuetKV trajectory; it is not obtained by rerunning the base policy in isolation. If the trace group associated with $\mathcal{X}_j$ is admitted, let $n_{j,g}^\pi$ and n_j be the counts of Eqs. equation 9 and equation 10: the (layer, KV head) pairs in group g, and in total, from which π evicts at least one exact KV entry. Then

$$\kappa_{j,g}^\pi = \text{cost}_g(\mathcal{M}_j) = m_j n_{j,g}^\pi, \qquad \kappa_j = \sum_{g=1}^{G_\pi} \kappa_{j,g}^\pi = m_j n_j, \tag{26}$$

so that each group is charged exactly the cells its own traces occupy; admission failure gives $m_j = 0$ and $\kappa_{j,g}^\pi = 0$ for every g. The matched-budget adapter $\mathcal{S}_\pi^\star$ of Eq. equation 12 then returns a *nested restriction* of the native outcome, that is, a set $\mathcal{R}_j$ satisfying

$$\mathcal{R}_j \cap \mathcal{G}_g^\pi \subseteq \overline{\mathcal{R}}_j \cap \mathcal{G}_g^\pi \qquad \forall g \in [G_\pi], \tag{27}$$

$$\left|\mathcal{R}_j \cap \mathcal{G}_g^\pi\right| + \kappa_{j,g}^\pi = \overline{b}_{j,g}^\pi \qquad \forall g \in [G_\pi]. \tag{28}$$

Deduction therefore removes only cells the native policy would otherwise have retained, and preserves its allocation granularity. Both conditions hold by construction rather than by assumption on π: Eq. equation 12 defines $\mathcal{S}_\pi^\star$ as a restriction of $\overline{\mathcal{R}}_j$ to the reduced group cardinalities. They are also all the analysis requires, presupposing neither a global ranking nor a scalar scoring representation of $\mathcal{S}_\pi$ and leaving how σ_π is computed entirely open. The charge need not be uniform across groups: uniformity, $\kappa_{j,g}^\pi = \kappa_j / G_\pi$, holds exactly when every compressed (layer, KV head) pair loses exact entries, so that $n_j = L_c H_{\text{kv}}$. It is a property of the eviction pattern, and nothing below depends on it.

Define the exact KV entries displaced by latent-capacity allocation as

$$\mathcal{E}_j = \overline{\mathcal{R}}_j \setminus \mathcal{R}_j. \tag{29}$$

Nestedness and Eq. equation 28 imply

$$\left|\mathcal{E}_j \cap \mathcal{G}_g^\pi\right| = \kappa_{j,g}^\pi \quad \forall g, \qquad |\mathcal{E}_j| = \sum_{g=1}^{G_\pi} \kappa_{j,g}^\pi = \kappa_j. \tag{30}$$

Together with Eq. equation 26 this gives identical group-wise cost for the two exchanged components,

$$\text{cost}_g(\mathcal{E}_j) = \text{cost}_g(\mathcal{M}_j) = \kappa_{j,g}^\pi \quad \forall g, \qquad \text{cost}(\mathcal{E}_j) = \text{cost}(\mathcal{M}_j) = \kappa_j. \tag{31}$$

The comparison therefore exchanges the displaced native-retained entries for an equally charged latent representation.

Event-wise representation exchange. Consider a segment $\mathcal{X}_j$ whose associated compression step admits a latent trace group. Its projected latent KV representation is

$$\mathcal{M}_j = \left(\left(\mathbf{K}^{(l)}_{\mathbf{M},j}, \mathbf{V}^{(l)}_{\mathbf{M},j} \right) \right)_{l \in \mathbb{L}_{\mathrm{cmp}}}, \tag{32}$$

restricted to the KV heads at which π evicts, which are the n_j pairs charged above. Thus, $\mathcal{M}_j$ contains only the projected latent KV states constructed from $\mathcal{X}_j$, excluding latent KV states accumulated from preceding segments. The construction and eviction domains also differ: the paired policy acts on the contemporaneous pruning slice $\mathcal{P}_j$ of Eq. equation 1, part of which belongs to the preceding segment because protection delays eviction, so that $\mathcal{X}_j$ and $\mathcal{P}_j$ have equal cardinality under a constant-width window without being the same set. $\mathcal{M}_j$ is therefore not a summary of the entries displaced at step j.

To isolate the current exchange, let

$$\mathcal{C}^{\mathrm{sh}}_j = \text{the complete resident state common to the two alternatives compared below;} \tag{33}$$

only this commonality enters the proof. In DuetKV it comprises, for each compressed layer $l \in \mathbb{L}_{\mathrm{cmp}}$,

$$\mathcal{C}^{\mathrm{sh},(l)}_j = \left(\mathbf{K}^{(l)}_{\mathcal{R},\le j} \oplus \mathbf{K}^{(l)}_{\mathbf{M},\le j-1} \oplus \mathbf{K}^{(l)}_{\mathcal{W},j},\ \mathbf{V}^{(l)}_{\mathcal{R},\le j} \oplus \mathbf{V}^{(l)}_{\mathbf{M},\le j-1} \oplus \mathbf{V}^{(l)}_{\mathcal{W},j} \right), \tag{34}$$

which is the resident cache $\mathcal{C}^{(l)}_j$ of Eq. equation 15 with $\mathcal{M}_j$ withheld, the latent index running to $j-1$ rather than to j. It is accompanied by $\mathcal{C}^{\mathrm{unc}}_j$, the resident state of the uncompressed layers: exact entries retained through segment j, latent states admitted earlier, and the protected window $\mathcal{W}_j$, whose width is set by the chunked-prefill schedule shared by every paired policy rather than by any one of them. When protection advances, entries of $\mathcal{W}_{j-1}$ surviving pruning are absorbed into $\mathcal{R}_j$ and the remainder discarded. The shared state excludes $\mathcal{M}_j$.

We compare

$$Z_{\mathrm{sel},j} = \left(\mathcal{C}^{\mathrm{sh}}_j, \mathcal{E}_j \right), \qquad Z_{\mathrm{hyb},j} = \left(\mathcal{C}^{\mathrm{sh}}_j, \mathcal{M}_j \right). \tag{35}$$

The first restores the exact KV entries displaced by the current latent allocation, whereas the second is the representation realized by DuetKV. Their preceding trajectories are identical, and they differ only in whether the exchanged capacity is assigned to $\mathcal{E}_j$ or $\mathcal{M}_j$. Equation equation 31 ensures that the two alternatives have identical cache cost.

Because $\mathcal{C}^{\mathrm{sh}}_j$, $\mathcal{E}_j$, and $\mathcal{M}_j$ are determined by the context and the realized DuetKV trajectory without access to prediction-time information U,

$$Y - (X, U) - \left(\mathcal{C}^{\mathrm{sh}}_j, \mathcal{E}_j, \mathcal{M}_j \right). \tag{36}$$

These three are jointly defined on one realized trajectory, so every information quantity in the local comparison below is evaluated under their common induced joint distribution. Thus $Z_{\mathrm{sel},j}$ restores the capacity displaced at step j while holding the preceding trajectory fixed; latent states admitted earlier remain in both alternatives and may already have altered the policy scores at j.

Theorem A.2 (Event-wise matched-capacity representation exchange)**.** *For the representations in Eq. equation 35,*

$$\mathcal{D}^{\star}\left(Z_{\mathrm{sel},j}\right) - \mathcal{D}^{\star}\left(Z_{\mathrm{hyb},j}\right) = I_F\left(Y; \mathcal{M}_j \mid \mathcal{C}^{\mathrm{sh}}_j, U\right) - I_F\left(Y; \mathcal{E}_j \mid \mathcal{C}^{\mathrm{sh}}_j, U\right). \tag{37}$$

Consequently,

$$\mathcal{D}^{\star}\left(Z_{\mathrm{hyb},j}\right) < \mathcal{D}^{\star}\left(Z_{\mathrm{sel},j}\right) \iff I_F\left(Y; \mathcal{M}_j \mid \mathcal{C}^{\mathrm{sh}}_j, U\right) > I_F\left(Y; \mathcal{E}_j \mid \mathcal{C}^{\mathrm{sh}}_j, U\right). \tag{38}$$

Proof. Applying Proposition A.1 to Eq. equation 35 gives

$$\mathcal{D}^{\star}\left(Z_{\mathrm{sel},j}\right) = H_F\left(Y \mid \mathcal{C}^{\mathrm{sh}}_j, \mathcal{E}_j, U\right) - H_F(Y \mid X, U), \tag{39}$$

$$\mathcal{D}^{\star}\left(Z_{\mathrm{hyb},j}\right) = H_F\left(Y \mid \mathcal{C}^{\mathrm{sh}}_j, \mathcal{M}_j, U\right) - H_F(Y \mid X, U). \tag{40}$$

Subtracting cancels the common full-context term. Adding and subtracting $H_F(Y \mid \mathcal{C}^{\mathrm{sh}}_j, U)$ gives

$$\begin{aligned} &H_F\left(Y \mid \mathcal{C}^{\mathrm{sh}}_j, \mathcal{E}_j, U\right) - H_F\left(Y \mid \mathcal{C}^{\mathrm{sh}}_j, \mathcal{M}_j, U\right) \\ &\quad = I_F\left(Y; \mathcal{M}_j \mid \mathcal{C}^{\mathrm{sh}}_j, U\right) - I_F\left(Y; \mathcal{E}_j \mid \mathcal{C}^{\mathrm{sh}}_j, U\right), \end{aligned} \tag{41}$$

which proves Eq. equation 37. Equation equation 38 follows directly. □

Equation equation 37 is an information identity and does not require equal capacity algebraically. Matched capacity makes the comparison operationally meaningful under DuetKV's fixed cache budget. The relevant quantity is therefore the conditional predictive information carried by $\mathcal{M}_j$ beyond the shared state, relative to that carried by the exact KV states it replaces. Define

$$\Delta I_j = I_F\left(Y; \mathcal{M}_j \mid \mathcal{C}_j^{\mathrm{sh}}, U\right) - I_F\left(Y; \mathcal{E}_j \mid \mathcal{C}_j^{\mathrm{sh}}, U\right). \tag{42}$$

Under matched capacity, $\Delta I_j > 0$ is exactly the condition under which replacing $\mathcal{E}_j$ with $\mathcal{M}_j$ reduces Bayes-optimal predictive distortion for the local exchange associated with $\mathcal{X}_j$.

Two limits of this statement should be read with it. It compares *Bayes-optimal* distortions, whereas both representations are consumed by the same frozen readout, so a favorable exchange at the representation level need not be realized by that readout; §A.3 separates the two effects. And it is *event-local*: the choice made at step j enters subsequent attention and can alter later hidden states and importance scores, so the sign of ΔI_j does not by itself determine the cumulative effect of a trajectory of such exchanges.

Although ΔI_j is not directly observable, the controls of §4.3 probe its two mechanisms operationally. Randomized traces remove context-dependent trace content while preserving trace count, cache charge, and per-slot scale, and so test whether the gain requires learned content rather than the cache cells the traces occupy. Restoring $\mathcal{E}_j$ alongside $\mathcal{M}_j$ instead abandons the capacity match and measures the empirical effect of restoring the displaced exact states while the learned traces are retained. Neither control identifies a conditional mutual information, and the second conditions on $\mathcal{M}_j$ rather than on $\mathcal{C}_j^{\mathrm{sh}}$ alone.

Decoupling from the contemporaneous selection outcome. Theorem A.2 identifies the information criterion governing the exchange and exposes a limitation of constructing the projected latent KV representation solely from information already contained in the shared cache.

For prefill segment $\mathcal{X}_j$, DuetKV constructs at layer l

$$\mathbf{M}_j^{(l)} = \mathcal{T}_\theta\left(\mathbf{H}_j^{(l)}\right), \tag{43}$$

where $\mathcal{T}_\theta$ is the per-layer constructor of §3.2 and $\mathbf{H}_j^{(l)}$ is the normalized pre-attention representation of the complete current segment. The latent trace constructor does not condition on the contemporaneous selection outcome for $\mathcal{X}_j$, that is, on the retained set, the discarded set, or any residual derived from that outcome. The frozen backbone K/V projections subsequently map these episodic latent traces to $(\mathbf{K}_{\mathbf{M},j}^{(l)}, \mathbf{V}_{\mathbf{M},j}^{(l)})$, which collectively form $\mathcal{M}_j$. The decoupling concerns functional dependence on the contemporaneous selection outcome; the constructor and the importance estimator may read different representations, the latter operating on key–value states or on offline statistics rather than on $\mathbf{H}_j^{(l)}$.

This decoupling is event-local rather than statistical: the constructor does not condition on the contemporaneous selection outcome, although traces admitted at earlier steps enter subsequent attention and may therefore affect later hidden representations and importance scores.

Corollary A.3 (Shared-state-only re-encoding)**.** *Suppose that the current projected latent KV representation is almost surely a deterministic function of the shared cache state,*

$$\mathcal{M}_j = \phi\left(\mathcal{C}_j^{\mathrm{sh}}\right) \qquad \textit{a.s.} \tag{44}$$

for some measurable map ϕ. Then

$$I_F\left(Y; \mathcal{M}_j \mid \mathcal{C}_j^{\mathrm{sh}}, U\right) = 0, \tag{45}$$

and therefore

$$\Delta I_j = -I_F\left(Y; \mathcal{E}_j \mid \mathcal{C}_j^{\mathrm{sh}}, U\right) \le 0. \tag{46}$$

Equality holds if and only if

$$I_F\left(Y; \mathcal{E}_j \mid \mathcal{C}_j^{\mathrm{sh}}, U\right) = 0. \tag{47}$$

Proof. Conditioning on $\mathcal{C}_j^{\mathrm{sh}}$ renders $\mathcal{M}_j$ deterministic, so $I_F(Y; \mathcal{M}_j \mid \mathcal{C}_j^{\mathrm{sh}}, U) = 0$; substituting into Eq. equation 42 gives Eq. equation 46, whose right-hand side is non-positive and vanishes exactly under Eq. equation 47. □

Corollary A.3 therefore states that a latent representation deterministically re-encoded from the shared cache cannot compensate, in the Bayes-optimal sense, for displaced exact entries carrying non-zero conditional predictive information. Constructing $\mathbf{M}_j^{(l)}$ from the complete current-segment representation $\mathbf{H}_j^{(l)}$ allows it to depend on information beyond $\mathcal{C}_j^{\mathrm{sh}}$ and so avoids this restriction; it does not guarantee $\Delta I_j > 0$, whose sign remains a property of the learned representation and the data distribution. Such a reparameterization may nevertheless reduce realization mismatch under the fixed Transformer readout, a distinction §A.3 makes precise.

A.2 Expressivity of Synthesized KV Representations

Theorem A.2 states the criterion but not whether it is reachable. We therefore turn to the representation interface itself and ask three complementary questions: whether synthesis enlarges the cache representation class beyond source selection, whether the frozen K/V projection imposes a linear reachability bottleneck, and whether native attention can distinguish the configurations that multiple synthesized keys make available.

Relaxing the source-subset constraint. Consider one layer and one KV head with realized source pairs $\{(\mathbf{k}_i, \mathbf{v}_i)\}_{i=1}^n$. A selection-only cache of capacity b belongs to

$$\mathfrak{S}_b = \left\{ [(\mathbf{k}_{i_r}, \mathbf{v}_{i_r})]_{r=1}^b : i_1, \dots, i_b \text{ are distinct elements of } [n] \right\}, \tag{48}$$

where brackets denote indexed cache slots rather than an unordered set.

For $0 \le \kappa \le b$, define the relaxed hybrid envelope

$$\mathcal{H}_{b,\kappa}^{\mathrm{free}} = \Big\{ [(\mathbf{k}_{i_r}, \mathbf{v}_{i_r})]_{r=1}^{b-\kappa} \parallel \left[(\tilde{\mathbf{k}}_s, \tilde{\mathbf{v}}_s) \right]_{s=1}^{\kappa} : \\ i_1, \dots, i_{b-\kappa} \text{ distinct}, \quad (\tilde{\mathbf{k}}_s, \tilde{\mathbf{v}}_s) \in \mathbb{R}^{d_h} \times \mathbb{R}^{d_h} \Big\}, \tag{49}$$

where $\parallel$ denotes concatenation of cache slots.

Proposition A.4 (Free-codeword relaxation)**.** *For every $0 \le \kappa \le b$,*

$$\mathfrak{S}_b \subseteq \mathcal{H}_{b,\kappa}^{\mathrm{free}}. \tag{50}$$

For $\kappa \ge 1$ the inclusion is strict, the ambient KV space being uncountable while the source sequence realizes finitely many pairs.

Proof. Take any element of $\mathfrak{S}_b$ and designate κ of its indexed slots as free codewords. Retaining the remaining $b - \kappa$ source slots and instantiating the free slots with the designated source pairs reproduces the original cache. For $\kappa \ge 1$, at least one free slot may instead instantiate a KV pair outside the finite set realized by the source sequence, establishing strict inclusion. □

Selection is confined to KV atoms instantiated by the source sequence; synthesis relaxes that constraint. $\mathcal{H}_{b,\kappa}^{\mathrm{free}}$ is an outer envelope rather than DuetKV's hypothesis class, the constructor realizing only a constrained subset of it. Which synthesized states remain reachable is decided by the frozen backbone interface.

Frozen KV projection interface. Episodic latent traces enter native attention through the backbone's frozen K/V projections. We therefore examine whether the affine projection stage introduces an additional linear rank bottleneck before subsequent normalization and positional transformation.

Under a column-vector convention, write the joint affine projection at layer l as

$$\begin{bmatrix} \mathbf{k} \\ \mathbf{v} \end{bmatrix} = \Psi^{(l)} \mathbf{m} + \mathbf{b}^{(l)}, \qquad \Psi^{(l)} = \begin{bmatrix} \Phi_K^{(l)} \\ \Phi_V^{(l)} \end{bmatrix} \in \mathbb{R}^{2H_{\mathrm{kv}} d_h \times D}. \tag{51}$$

Lemma A.5 (Surjectivity of the frozen affine KV projection)**.** *If*

$$\operatorname{rank} \Psi^{(l)} = 2H_{\mathrm{kv}} d_h, \tag{52}$$

then the affine map in Eq. equation 51 is surjective onto $\mathbb{R}^{2H_{\mathrm{kv}}d_h}$. *Every target* $\mathbf{y}^\star$ *admits a minimum-norm preimage satisfying*

$$\|\mathbf{m}^\star\|_2 \le \frac{\left\|\mathbf{y}^\star - \mathbf{b}^{(l)}\right\|_2}{\sigma_{\min}\left(\Psi^{(l)}\right)}. \tag{53}$$

Proof. Full row rank makes $\Psi^{(l)}$ surjective. The minimum-norm preimage is

$$\mathbf{m}^\star = \left(\Psi^{(l)}\right)^+ \left(\mathbf{y}^\star - \mathbf{b}^{(l)}\right), \tag{54}$$

and

$$\left\|\left(\Psi^{(l)}\right)^+\right\|_2 = \frac{1}{\sigma_{\min}\left(\Psi^{(l)}\right)}. \tag{55}$$

The stated bound follows. □

Numerical verification. The premise of Lemma A.5 is a property of an individual backbone's frozen weights and is therefore checked per model. For Qwen2.5-7B-Instruct-1M, $D = 3584$, $H_{\mathrm{kv}} = 4$ and $d_h = 128$ give $\Psi^{(l)} \in \mathbb{R}^{1024\times 3584}$; for Qwen3-8B, $D = 4096$, $H_{\mathrm{kv}} = 8$ and $d_h = 128$ give $\Psi^{(l)} \in \mathbb{R}^{2048\times 4096}$. Using double-precision singular-value decomposition and the numerical-rank criterion

$$\sigma_i > \max(m, n)\, \varepsilon_{\mathrm{fp64}}\, \sigma_{\max}, \tag{56}$$

every layer of both models attains full numerical row rank, with $\sigma_{\min} \in [0.044, 0.181]$ across the 28 layers of the first and $\sigma_{\min} \in [0.004, 0.366]$ across the 36 layers of the second. The smallest $\sigma_{\min}$ exceeds its threshold by 1.2×10^{10} and 4.8×10^{8} respectively, so the rank determination is far from the numerical boundary in both. The preimage bound of Eq. equation 53 scales as $1/\sigma_{\min}$ and is correspondingly looser on Qwen3-8B, whose worst-conditioned layer has $\sigma_{\max}/\sigma_{\min} \approx 2.3 \times 10^3$ against 1.1×10^2 for Qwen2.5-7B-Instruct-1M.

At a fixed position, RoPE applies an invertible orthogonal transformation to the key coordinates while leaving the value coordinates unchanged and therefore preserves the rank of the joint projected representation. Key normalization, when present, is nonlinear and lies outside the scope of Lemma A.5. The lemma therefore establishes only that the affine K/V projection introduces no additional linear rank bottleneck. It does not imply surjectivity of the complete latent-to-cache mapping, which remains constrained by normalization, positional transformation, the latent trace constructor, parameter sharing, and causal inputs.

The lemma also settles a granularity mismatch between synthesis and selection. Eviction decides each (layer, KV head, position) cell separately, whereas one latent trace passes through $\Psi^{(l)}$ and thereby fixes the key and value of *every* KV head of that layer at once, so the free codewords of Eq. equation 49 are available per head only if the corresponding joint target is reachable. Surjectivity onto $\mathbb{R}^{2H_{\mathrm{kv}}d_h}$ is exactly that condition: the coupling across heads costs no reachability in the projected space, and what remains constrained is the constructor, not the interface.

Reachability in projected KV space is still insufficient unless native attention can distinguish the resulting configurations.

Attention-readout expressivity. For a set G of simultaneously visible keys, define its contribution to the attention log-partition function as

$$\ell_G(\mathbf{q}) = \log \sum_{i\in G} \exp\left(\frac{\mathbf{q}^\top \mathbf{k}_i}{\sqrt{d_h}}\right). \tag{57}$$

Proposition A.6 (Curvature of grouped attention mass)**.** *Let*

$$\alpha_i(\mathbf{q}) = \frac{\exp(\mathbf{q}^\top \mathbf{k}_i/\sqrt{d_h})}{\sum_{r\in G}\exp(\mathbf{q}^\top \mathbf{k}_r/\sqrt{d_h})}. \tag{58}$$

Then

$$\nabla_{\mathbf{q}} \ell_G(\mathbf{q}) = \frac{1}{\sqrt{d_h}} \sum_{i \in G} \alpha_i(\mathbf{q}) \mathbf{k}_i, \tag{59}$$

$$\nabla^2_{\mathbf{q}} \ell_G(\mathbf{q}) = \frac{1}{d_h} \operatorname{Cov}_{\alpha(\mathbf{q})} [\mathbf{k}_i] \succeq 0. \tag{60}$$

On any open region where the attention-weighted key covariance is non-zero, no single key can reproduce ℓ_G throughout that region.

Proof. The gradient of the log-sum-exp is the softmax-weighted mean of its linear components. Differentiating this mean yields their softmax-weighted covariance. A single key contributes a linear function of $\mathbf{q}$ and therefore has identically zero Hessian. □

For the group of M synthesized keys $\{\tilde{\mathbf{k}}_r\}_{r=1}^M$, write

$$\beta_r(\mathbf{q}) = \frac{\exp(\mathbf{q}^\top \tilde{\mathbf{k}}_r / \sqrt{d_h})}{\sum_{s=1}^M \exp(\mathbf{q}^\top \tilde{\mathbf{k}}_s / \sqrt{d_h})}, \qquad r \in [M], \tag{61}$$

for the weights of Eq. equation 58 on that group, and ℓ_M for its log-partition contribution in the sense of Eq. equation 57. The same identity gives

$$\nabla^2_{\mathbf{q}} \ell_M(\mathbf{q}) = \frac{1}{d_h} \operatorname{Cov}_{\beta(\mathbf{q})} \left[\tilde{\mathbf{k}}_r \right]. \tag{62}$$

The curvature vanishes for $M = 1$ and whenever all synthesized keys coincide. Distinct synthesized keys can instead induce non-zero directional curvature, yielding log-partition geometry that no single key reproduces on such a region.

Coincident traces are nevertheless not equivalent to a single trace because multiplicity changes the log-partition function. If $\tilde{\mathbf{k}}_1 = \cdots = \tilde{\mathbf{k}}_M = \tilde{\mathbf{k}}$, then

$$\ell_M(\mathbf{q}) = \log M + \frac{\mathbf{q}^\top \tilde{\mathbf{k}}}{\sqrt{d_h}}. \tag{63}$$

Coincidence therefore removes directional covariance without eliminating the multiplicity contribution.

This result concerns only the key-dependent log-partition geometry: it neither characterizes value-side approximation nor separates synthesized keys from an equally sized set of exact keys, which carries its own covariance. What it establishes is that representing a segment by several slots admits query-dependent geometry that collapsing it into one representative state cannot. The calibration of Eq. equation 5 fixes the offsets and scale of the slot logits but guarantees neither trace diversity nor a benefit from increasing M; both remain properties of the learned constructor and are evaluated empirically.

Together, Proposition A.4, Lemma A.5, and Proposition A.6 separate three requirements on the interface: synthesis removes the source-subset constraint in an outer relaxation, the frozen affine projection adds no linear rank bottleneck, and several slots admit query-dependent geometry unavailable to a single representative key. None establishes that the trained constructor attains $\Delta I_j > 0$; that depends on optimization and the data distribution. We next relate the training objective to predictive distortion.

A.3 Predictive Self-Distillation

Let

$$Z_\theta = \mathcal{C}_{\pi,\theta}(X) \tag{64}$$

denote the compressed cache produced by the frozen base evictor together with the trainable latent trace constructor θ. Here, $\pi = (\sigma_\pi, \mathcal{A}_\pi, \mathcal{S}_\pi)$ denotes the fixed paired base policy, including its importance estimator, allocation rule, and native selection rule. At autoregressive step t, let

$$U_t = (Q, Y_{<t}), \tag{65}$$

where $Y_{<t}$ is the teacher-forced continuation prefix.

For DuetKV, the generic predictor r introduced in §A.1 is instantiated by the frozen Transformer conditioned on Z_θ. Denote the resulting conditional distribution by $p_{\pi,\theta}(y \mid Z_\theta, U_t)$. The population forward-distillation objective is

$$\mathcal{L}_t(\theta) = \mathbb{E} D_{\mathrm{KL}}\left(p_F(Y_t \mid X, U_t) \,\|\, p_{\pi,\theta}(Y_t \mid Z_\theta, U_t)\right). \tag{66}$$

For every fixed θ, the decomposition underlying Proposition A.1 yields

$$\begin{aligned}\mathcal{L}_t(\theta) = &\underbrace{I_F\left(Y_t; X \mid Z_\theta, U_t\right)}_{\text{representation deficiency}} \\ &+ \underbrace{\mathbb{E} D_{\mathrm{KL}}\left(p_F(Y_t \mid Z_\theta, U_t) \,\|\, p_{\pi,\theta}(Y_t \mid Z_\theta, U_t)\right)}_{\text{realization mismatch}}.\end{aligned} \tag{67}$$

The first term is the irreducible predictive distortion induced by the compressed representation. Theorem A.2 characterizes how the matched-capacity exchange between displaced exact KV states and the current projected latent KV representation changes this representation-level term. The second measures the mismatch between the conditional distribution available from that representation and the distribution realized by the frozen Transformer.

The population forward-distillation objective therefore decomposes exactly into representation deficiency and realization mismatch. This decomposition motivates output-level self-distillation rather than direct reconstruction of evicted K/V states: episodic latent traces need not reproduce any particular discarded KV entry, but must preserve predictive information in a form accessible to the frozen attention stack.

Equation equation 67 is a population identity for each fixed θ; the implemented loss of §3.3 is its finite-sample counterpart. To bound activation memory, constructor computation is reconstructed segment-wise and gradients are truncated through the realized prefill trajectory, so Eq. equation 67 characterizes the population objective rather than full backpropagation through all preceding compression decisions.

A.4 Budget Scaling

The matched-capacity construction establishes local equality between the exact KV states displaced and the latent KV states admitted at each applicable compression step. We conclude by characterizing how these local exchanges accumulate with context length.

Distinct compression steps act on disjoint domains. By Eq. equation 1 the pruning slices satisfy

$$\mathcal{P}_i \cap \mathcal{P}_j = \varnothing, \qquad i \neq j, \tag{68}$$

and a retention decision, once taken for a processed domain, is not revisited at a later step. Since $\overline{\mathcal{R}}_j \subseteq \mathcal{P}_j$ and $\mathcal{R}_j \subseteq \overline{\mathcal{R}}_j$, the native and post-deduction exact sets are likewise pairwise disjoint across steps, and so are the displaced sets $\mathcal{E}_j$. Cumulative capacity is therefore additive over steps rather than a sum over overlapping cache snapshots. For a context prefix ending after step t, define

$$B_{\mathrm{native}}(\le t) = \left|\biguplus_{j \le t} \overline{\mathcal{R}}_j\right| = \sum_{j \le t} \overline{b}^{\pi}_j, \qquad \overline{b}^{\pi}_j = \sum_{g=1}^{G_\pi} \overline{b}^{\pi}_{j,g}, \tag{69}$$

$$B_{\mathrm{exact}}(\le t) = \left|\biguplus_{j \le t} \mathcal{R}_j\right|, \tag{70}$$

$$B_{\mathrm{latent}}(\le t) = \sum_{j \le t} \kappa_j = \sum_{j \le t} m_j n_j, \tag{71}$$

which reduces to $L_c H_{\mathrm{kv}} \sum_{j \le t} m_j$ when every compressed (layer, KV head) pair is charged. These three are already linked by an exact identity. At every step $\overline{\mathcal{R}}_j = \mathcal{R}_j \uplus \mathcal{E}_j$ by Eq. equation 29, and

by Eq. equation 68 these sets are pairwise disjoint across steps; taking the disjoint union over $j \le t$ and counting cells gives the prefix-level ledger

$$B_{\text{exact}}(\le t) + B_{\text{latent}}(\le t) = \sum_{j \le t} \overline{b}_j^{\pi} = B_{\text{native}}(\le t). \tag{72}$$

The right-hand side is the cumulative allocation of the unmodified base policy before latent-capacity deduction, evaluated along the same realized DuetKV trajectory rather than by executing that policy independently; the ledger covers the evictable cells of the compressed layers, protected-window and uncompressed-layer states being common to both sides. DuetKV therefore preserves the native budget not only at each compression event but also in the cumulative cache-cell accounting over the processed prefix.

Whenever $B_{\text{native}}(\le t) > 0$, the cumulative latent share is

$$f_{\text{latent}}(\le t) = \frac{B_{\text{latent}}(\le t)}{B_{\text{native}}(\le t)}. \tag{73}$$

A trace group is admissible only if its charge fits within every native group and its share of the event budget stays below the cap,

$$\kappa_{j,g}^{\pi} \le \overline{b}_{j,g}^{\pi} \quad \forall g, \qquad \frac{\kappa_j}{\overline{b}_j^{\pi}} \le \eta_{\max}. \tag{74}$$

The first condition keeps every post-deduction group cardinality non-negative; the second bounds the event-wise latent fraction. Together they are the admission rule of Eq. equation 11. Under the uniform charge the first collapses to

$$\min_g \overline{b}_{j,g}^{\pi} \ge \frac{L_c H_{\text{kv}} M}{G_\pi}, \tag{75}$$

a specialization to uniform charging rather than a property of $\mathcal{A}_\pi$. Both conditions read off the realized group budgets; for a generic π no scalar threshold in context length is equivalent to them.

The admission cap also gives a policy-independent cumulative bound. For each admitted step, $\kappa_j \le \eta_{\max} \overline{b}_j^{\pi}$; for a rejected step, $\kappa_j = 0$. Summing over steps yields

$$B_{\text{latent}}(\le t) \le \eta_{\max} B_{\text{native}}(\le t), \qquad f_{\text{latent}}(\le t) \le \eta_{\max}. \tag{76}$$

For intuition, write

$$N_j^{\text{ev}} = |\mathcal{P}_j| \tag{77}$$

for the number of positions exposed to eviction at compression step j; under a protected window of constant width, $N_j^{\text{ev}} = C_j$ for every segment after the first. Consider a regular regime in which $N_j^{\text{ev}} = N^{\text{ev}}$ and $n_j = n$, every trace group is admitted, and the native policy realizes an aggregate retained fraction r_{ret} over the evictable cells of the compressed layers. Then $\overline{b}_j^{\pi} \approx r_{\text{ret}} N^{\text{ev}} L_c H_{\text{kv}}$ and Eq. equation 73 reduces to

$$f_{\text{latent}} \approx \frac{n}{L_c H_{\text{kv}}} \cdot \frac{M}{N^{\text{ev}} r_{\text{ret}}}, \tag{78}$$

where $n/(L_c H_{\text{kv}}) \in (0, 1]$ is the fraction of compressed (layer, KV head) pairs that receive traces, equal to 1 under uniform charging. The retention fraction r_{ret} is the one the native policy realizes, and it need not equal the control parameter ρ^{eff} of Eq. equation 25: policy-specific grouping, tie handling, or cardinality constraints can quantize the realized allocation. Equation equation 78 is a steady-regime approximation rather than an admission rule, admission being governed by Eq. equation 74. Within that regime the latent fraction approaches a constant for long contexts processed in fixed-size segments, both allocations growing linearly in the number of admitted segments, while the ledger of Eq. equation 72 holds exactly at every prefix and in every regime. Because decoding-time traces are charged to the same budget, the ledger extends unchanged to autoregressive decoding.

B Additional Experimental Details

B.1 Distillation Data

Corpus construction and split integrity. We construct a distillation corpus of 2,371 contexts from 4,109 source documents: 1,728 contexts contain a single document, while 643 concatenate two to seventeen documents. PG-19, FineWeb-Edu, NarrativeQA, and source code contribute 981, 723, 387, and 280 contexts, respectively. We partition the corpus at the source-document level into 2,017 training, 217 validation, and 137 test contexts, keeping source documents disjoint across splits. To control evaluation contamination, we exclude the FineWeb-Edu documents used to train the frozen Fast KVzip gate and all repositories represented in Code.RepoQA, then screen the remaining training corpus against all twelve evaluation panels using content-level chunk matching to detect shared text under altered surface forms; no evaluation sample is flagged as contaminated.

Query and target construction. We pair each context with $K = 5$ queries, yielding 11,855 context–query pairs. The 387 NarrativeQA contexts use their dataset-provided human-written queries, while for the remaining 1,984 contexts we generate five queries with Qwen2.5-7B-Instruct-1M according to domain-specific taxonomies: factual, summary, multi-hop, method, and comparison for comprehension; multiple-choice, counting, ordering, negation, and entailment for reasoning; and symbol, overview, cross-file, data-flow, and branch for code. Each query is generated from evidence drawn from a local window, two distant windows, or excerpts distributed throughout the context to diversify evidence locations and discourage positional shortcuts; the resulting queries are screened for lexical grounding and cross-document template similarity before being fixed and shared across backbones. Distillation targets are generated separately for each backbone by greedy full-cache rollout. NarrativeQA reference answers are retained only for reference and are not used as distillation targets.

Training-set filtering. We filter the training split according to the retention budget available outside the protected local window. Let n denote the context length, $C = 16{,}000$ the prefill segment capacity, and $W = 4{,}096$ the nominal size of the protected local window. We define

$$W_{\text{eff}}(n) = \begin{cases} \lfloor 0.02\, n \rfloor, & n < C, \\ W, & n \geq C, \end{cases} \qquad \rho_{\text{comp}}(n) = \frac{\rho n - W_{\text{eff}}(n)}{n - W_{\text{eff}}(n)}, \tag{79}$$

where ρ is the nominal retention ratio and $\rho_{\text{comp}}(n)$ is the induced retention ratio over the compressible region. A context is eligible if

$$\rho n > W_{\text{eff}}(n) \qquad \text{and} \qquad \rho_{\text{comp}}(n) \geq 0.02. \tag{80}$$

The first condition ensures positive retention capacity outside the protected window, while the second excludes compression regimes more aggressive than those encountered at evaluation, where $\rho_{\text{comp}} \in [0.046, 0.191]$.

Because $W_{\text{eff}}(n)$ scales with n below one full segment, the criterion does not impose a minimum context length. At the training ratio $\rho = 0.1$, it excludes only $n \in [16{,}000, 50{,}176)$, accounting for all 427 removed contexts while retaining 236 single-segment contexts. After filtering, 1,590 contexts remain: 707 from FineWeb-Edu, 503 from PG-19, 221 from source code, and 159 from NarrativeQA, with a median length of 102,304 tokens and a maximum of 175,896. Filtering is restricted to training, while all benchmark instances are evaluated as provided.

B.2 Implementation and Optimization

Configuration. The constructor uses $M = 4$ layer-specific slots, shared across prefill segments, with bottleneck width $d_m = 128$ and logit calibration (Eq. equation 6) at target scale $\gamma = 3$. Separate constructor parameters are instantiated at each layer, yielding 51.88M trainable parameters on Qwen2.5-7B-Instruct-1M (0.68% of the 7.62B backbone), 127.06M on Qwen2.5-14B-Instruct-1M (0.86% of 14.77B) and 76.23M on Qwen3-8B (0.93% of 8.19B). Prefill uses 16K-token segments with a nominal 4K protected local window, and the matched KV budget defined in §3.2 is enforced identically during training and evaluation.

Across the reported budgets, the constraint $Mn_j \le \eta_{\max}\overline{b}_j^{\pi}$ in Eq. equation 11 ($\eta_{\max} = 0.40$) never binds during training or on nine of the twelve evaluation tasks. It binds on 99.83% of GSM8K segments at $\rho = 0.1$, whose median context length of 83 tokens is far below the 16K-token segment length, and on at most 6.35% of segments in all other reported settings. Segments for which it binds fall back to the base policy alone.

Optimization. Each constructor is trained only at $\rho = 0.1$ with seed 0, for 300 optimization steps on Qwen2.5-7B/14B-Instruct-1M and 400 steps on Qwen3-8B. The contexts and queries are shared across backbones, while teacher continuations are generated separately by greedy decoding from the corresponding full-cache backbone. Training is data-parallel across 8 H100 GPUs, with each device processing one context and its $K = 5$ queries, so each update covers 8 contexts and 40 continuations. For each context, the K continuation losses are backpropagated sequentially to the latent K/V states, with their gradients accumulated according to the averaging in Eq. equation 17 and propagated through the constructor in a second pass (Appendix B.4). Gradients are not accumulated across optimization steps.

We use AdamW with $(\beta_1, \beta_2) = (0.9, 0.99)$, a peak learning rate of 10^{-3}, gradient clipping at 1.0 and linear warmup for 10 steps on Qwen2.5-7B/14B-Instruct-1M and 12 steps on Qwen3-8B, followed by cosine decay to 5×10^{-5}. Weight decay of 0.01 is applied only to the key, value and output projections. The slots $\mathbf{S}$, the query projection $\mathbf{W}_Q$ and the normalization scale $\boldsymbol{\omega}$ are exempt. Training the Qwen2.5-7B-Instruct-1M constructor takes 206 minutes (3.43 hours) on 8 H100 GPUs, with a median step time of 41 seconds and peak allocated memory of 30.9 GB per GPU.

B.3 Efficiency Measurements

All latencies are measured on H100 GPUs with PyTorch and FlashAttention-2, in a layout that physically removes evicted entries. Prefill times are medians of five runs, and decoding times are medians of 64 per-token latencies after eight warm-up steps. Because absolute prefill time varies by up to 2.7% across nominally identical GPUs, both arms of each comparison are run in one process on one device. At $\rho = 0.3$, trace construction adds 0.9–1.5% to prefill latency over the plotted contexts and 2.9% at 32K, as the overhead decreases with context length. Decoding latency changes by -1.4% to $+1.0\%$. The constructor adds 198 MiB of resident parameters, independent of context length. KVzip (Kim et al., 2026b) is measured with its authors' code, prefill in our fork and decoding in the released implementation: its scoring pass costs $2.2\times$ a standard prefill, matching the factor its authors report on other hardware, and the two harnesses agree to within 0.27% on a shared configuration.

Decoding speed-up. The dense full-cache baseline in Figure 6b uses a different attention kernel from the compressed cache. Passing the uncompressed cache through the same variable-length kernel gives 24.8, 30.3 and 37.3 ms/token at 160K, 240K and 320K, so DuetKV's speed-ups of $1.14\times$, $1.46\times$ and $1.79\times$ over the dense baseline become $0.93\times$, $1.10\times$ and $1.34\times$ against this kernel-matched one, and the gain from a smaller cache emerges only beyond 240K.

B.4 Truncated-Gradient Implementation and Validation

Two-pass gradient evaluation. As illustrated in Figure 7, each committed latent trace is detached from the autograd graph during prefill, truncating gradient paths through its influence on later hidden representations, their derived exact K/V states and subsequent episodic traces. Its gradient path from downstream predictions through the cached latent K/V states is retained. Hard retention decisions remain non-differentiable irrespective of this truncation.

To evaluate the resulting truncated gradient g_{trunc} without retaining the full prefill graph, we use a two-pass procedure. In the first pass, the cached $\mathbf{K}^{(l)}_{\mathbf{M},j}$ and $\mathbf{V}^{(l)}_{\mathbf{M},j}$ are treated as differentiable leaves, and the K training queries are replayed independently. Their contributions to the objective in Eq. equation 17 are accumulated in FP32 buffers as

$$\mathbf{G}^{(l)}_{\mathbf{K},j} = \frac{\partial \mathcal{L}}{\partial \mathbf{K}^{(l)}_{\mathbf{M},j}}, \qquad \mathbf{G}^{(l)}_{\mathbf{V},j} = \frac{\partial \mathcal{L}}{\partial \mathbf{V}^{(l)}_{\mathbf{M},j}}. \tag{81}$$

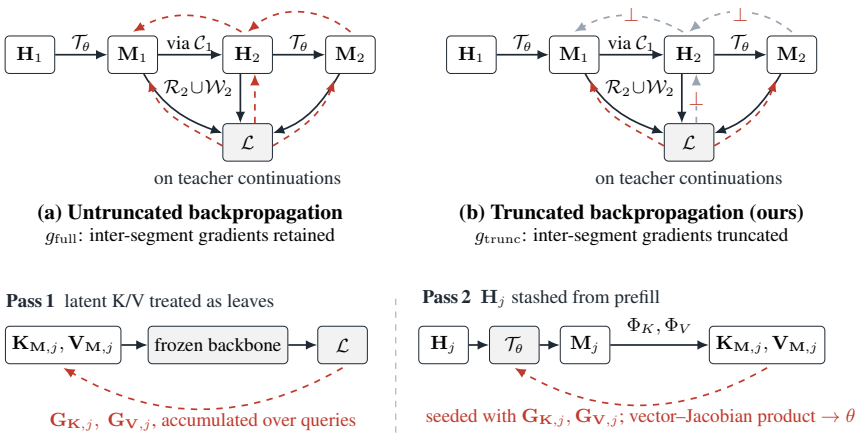


Figure 7: **Forward computation and gradient flow.** $\mathbf{H}_j$ denotes the hidden states of prefill segment $\mathcal{X}_j$ and $\mathbf{M}_j$ the latent traces constructed from them by $\mathcal{T}_\theta$. The arrow labeled $\mathcal{R}_2 \cup \mathcal{W}_2$ carries the exact key–value states of $\mathcal{X}_2$, comprising its score-retained set and protected window. Solid arrows denote forward dependencies, dashed arrows gradient paths: red where included in the panel's gradient, grey ($\perp$) where truncated. The forward computation is identical in (a) and (b): an admitted trace enters the cache $\mathcal{C}_j$ only after its source segment, and then conditions later segment representations, from which both the exact key–value states and the subsequent traces are derived. Gradient therefore reaches an earlier trace along a direct path, through its admitted latent states $\mathbf{K}_{\mathbf{M},j}, \mathbf{V}_{\mathbf{M},j}$, and along indirect paths across segment boundaries. **(a)** Retaining all of these paths gives g_{full}, but its backward activation cost grows with the segment count. **(b)** DuetKV retains the direct path and truncates the indirect ones by treating the realized prefill trajectory as fixed during optimization. **(c)** The resulting g_{trunc} is evaluated in two passes: the gradients $\mathbf{G}_{\mathbf{K},j}, \mathbf{G}_{\mathbf{V},j}$ with respect to those latent states, treated as leaves, are accumulated over the K training queries, then propagated into θ by recomputing one layer–segment pair at a time. This factorization introduces no additional gradient approximation. Dependencies that carry no constructor gradient even under untruncated backpropagation are omitted, as are layer indices and the latent projection's key normalization and positional anchor.

These gradients therefore include both the averaging over continuation steps within each query and the averaging across queries in Eq. equation 17. Summing the corresponding vector–Jacobian products gives

$$g_{\text{trunc}} = \sum_l \sum_j \left[\left(\frac{\partial \mathbf{K}_{\mathbf{M},j}^{(l)}}{\partial \theta} \right)^\top \mathbf{G}_{\mathbf{K},j}^{(l)} + \left(\frac{\partial \mathbf{V}_{\mathbf{M},j}^{(l)}}{\partial \theta} \right)^\top \mathbf{G}_{\mathbf{V},j}^{(l)} \right], \tag{82}$$

where l ranges over the L_c compressed layers and the Jacobians are taken with the constructor inputs $\mathbf{H}_j^{(l)}$ held fixed, as implied by the truncation. The second pass evaluates these products by recomputing one layer–segment pair at a time from the stashed constructor inputs and rotary tables, seeding each backward pass with $(\mathbf{G}_{\mathbf{K},j}^{(l)}, \mathbf{G}_{\mathbf{V},j}^{(l)})$. Under a synthetic FP32 loss, this factorization matches one-shot backpropagation on the same truncated graph to a relative ℓ_2 error of 5.2×10^{-8}.

Effect of gradient truncation. The two-pass factorization changes only how the truncated gradient is evaluated; the inter-segment truncation itself remains an approximation. We quantify this approximation by retaining inter-segment constructor gradients during prefill to obtain the untruncated gradient g_{full} and comparing it with g_{trunc} using the relative ℓ_2 difference

$$\frac{\|g_{\text{full}} - g_{\text{trunc}}\|_2}{\|g_{\text{full}}\|_2}$$

and cosine similarity.

context (tokens)	segment (tokens)	segments	relative ℓ_2	$\cos(g_{\text{full}}, g_{\text{trunc}})$	untruncated peak
4,000	2,048	2	26.9%	+0.963	25.6 GB
8,200	2,048	5	32.4%	+0.956	45.2 GB
8,200	4,096	3	37.6%	+0.929	36.9 GB

Despite relative ℓ_2 differences of 26.9–37.6%, g_{trunc} remains strongly aligned with g_{full}, with cosine similarities of 0.929–0.963. Because untruncated backpropagation is already memory-intensive at these short context lengths, we use it only for this diagnostic; all training uses the truncated gradient evaluated by the two-pass procedure.

B.5 Decoding Protocol and Latent Retention

Table 3 summarizes the decoding configuration. DuetKV uses the same constructor trained for prefill, without decoding-specific adaptation. For a native Fast KVzip compression interval Δ, trace construction is performed every $I = q\Delta$ generated tokens, provided that the corresponding compression boundary triggers eviction.

Table 3: Decoding configuration. Settings above the divider are inherited from the paired Fast KVzip policy and shared with the baseline; those below are specific to DuetKV.

Backbone	Qwen3-8B, thinking mode enabled
Sampling	temperature 0.6, nucleus $p = 0.95$
Generation budget	32,768 (AIME24), 16,384 (MATH-500)
Forced-answer offset	2,048 tokens
Compression interval Δ	128 generated tokens
Cache budget	AIME24: 512–6144; MATH-500: 512–4096 (entries per KV head and layer)
Replicates	AIME24: 30 problems $\times$ 16 seeds; MATH-500: one seed
Trace-construction interval I	AIME24: $1{,}024 = 8\Delta$; MATH-500: $512 = 4\Delta$
Construction condition	native eviction at the corresponding boundary
Latent cap $B_{\max}$	16 entries per (layer, KV head) pair, independent of cache budget
Latent retention	strided

Strided latent retention. Each construction event contributes M latent entries per (layer, KV head) pair, which accumulate across successive events up to the fixed capacity $B_{\max}$. Let $K = B_{\max}/M$ denote the maximum number of construction events represented at full capacity. When a new construction event would exceed this limit, DuetKV retains the traces from the oldest and most recent events and removes those associated with the interior event whose removal induces the smallest merged temporal gap. Specifically, for the $K+1$ construction times $t_1 < \cdots < t_{K+1}$, the discarded event is indexed by

$$j^\star = \arg\min_{2 \le j \le K} (t_{j+1} - t_{j-1}) . \tag{83}$$

The corresponding M latent entries are removed together. Repeating this procedure after each capacity-exceeding construction maintains temporally distributed coverage of the generated history while keeping latent capacity fixed at $B_{\max}$.

C Additional Ablations and Analysis

We extend the main ablations in Section 4.3 along two dimensions: robustness to the compression configuration and the behavior of the learned latent traces. We first test whether a constructor trained with the default protected local window transfers to a different inference-time window, and then examine how latent traces aggregate source states, what representational structure they acquire, and how the frozen backbone uses them during downstream inference. Unless otherwise stated, all experiments use Qwen2.5-7B-Instruct-1M with Fast KVzip as the base evictor under matched total KV-cache budgets.

C.1 Sensitivity to the Protected Local Window

We reduce the protected local window from 4096 to 2048 at inference time, while keeping the segment size and total cache budget fixed and reusing the constructor trained at $W = 4096$. To ensure comparable eviction regimes, we exclude SQuAD and GSM8K, whose contexts are shorter than one prefill segment and therefore yield the same effective window after implementation-level rescaling. Together with ICL.ManyShot at $\rho = 0.1$, which is absent from the main evaluation because its retained set is then determined entirely by the protected window, this leaves 59 evaluation cells jointly comparable across the two window sizes.

Table 4 shows that the mean gain over Fast KVzip changes from $+4.03$ at $W = 4096$ to $+3.68$ at $W = 2048$, with retrieval remaining the largest source of improvement ($+8.65$ versus $+7.90$). The reduction is concentrated at the two tightest budgets, $\rho \in \{0.2, 0.1\}$, where the gain narrows from $+7.63$ to $+5.87$. This narrowing reflects a larger improvement in Fast KVzip rather than a decline in DuetKV: reducing W raises DuetKV from 43.48 to 44.67 and Fast KVzip from 35.84 to 38.80. Two cells at $\rho = 0.2$ are partially degenerate, affecting 5.8% of MRCR samples and 33.3% of ICL.ManyShot samples; excluding them leaves 57 cells and changes the aggregate gains only to 4.15 and 3.79, respectively. Thus, the gains learned under the default window largely persist when the protected window is halved at inference time without retraining.

Table 4: Sensitivity to the protected local window under matched cache budgets. Results are averaged over the 59 cells jointly comparable between $W = 4096$ and $W = 2048$; Δ denotes the paired mean gain over Fast KVzip at the same window, with gains above 5 points in bold. MRCR uses the full $1{,}476$-sample protocol at both window sizes. The aggregate is not directly comparable to Table 1, which additionally includes SQuAD and GSM8K.

		$W = 4096$ (default)			$W = 2048$		
Subset	Cells	Fast KVzip	DuetKV	Δ	Fast KVzip	DuetKV	Δ
All	59	43.57	47.61	$+4.03$	44.31	47.99	$+3.68$
$\rho \in \{0.2, 0.1\}$	19	35.84	43.48	$+\mathbf{7.63}$	38.80	44.67	$+\mathbf{5.87}$
Retrieval	24	43.31	51.97	$+\mathbf{8.65}$	44.82	52.71	$+\mathbf{7.90}$
Contextual QA	12	56.45	57.24	$+0.79$	57.34	57.43	$+0.09$
Redundancy	23	37.12	38.03	$+0.91$	36.99	38.14	$+1.14$

C.2 Analysis of the Learned Traces

We analyze the learned traces on 20 Retr.KV contexts with a median length of 169K tokens, each partitioned into eleven segments and evaluated at $\rho = 0.1$ with Qwen2.5-7B-Instruct-1M under the segment size and protected window used in §4.2. This panel aligns the diagnostics with the ablations they are intended to explain. Unless stated otherwise, construction- and representation-side statistics are averaged over all valid $(\text{context}, \text{segment}, \text{layer})$ tuples. The retained-cache layouts used for these diagnostics and for the read-out analysis contain the same logical entries and differ only in accumulation order. Table 5 summarizes the evaluated checkpoints, including four trained exclusively for this analysis.

Aggregation breadth. For the constructor attention weights $\mathbf{a}$ of a latent slot, we define its effective source support as

$$N_{\text{eff}} = \frac{1}{\sum_i a_i^2}. \tag{84}$$

By construction, γ specifies the target RMS scale of the centered, normalized constructor logits in Eq. equation 6, which the constructor realizes exactly: the measured scales are 2.00, 3.00, and 4.00 for the three checkpoints differing only in γ, with no variation across contexts. The corresponding aggregation breadth changes substantially: increasing the target from 2 to 3 and 4 reduces N_{eff} from $2{,}980$ to 711 and 161, respectively, with no overlap across contexts and an across-context standard deviation of at most 5.2. Removing calibration instead drives the measured scale to 282 and concentrates aggregation on nearly a single source position ($N_{\text{eff}} = 12.6$ of $15{,}378$ positions; mean maximum attention weight 0.99), which no other probed checkpoint does.

Table 5: **Constructor diagnostics on the probed checkpoints.** N_{eff} is the effective number of source tokens a trace aggregates and the effective rank is that of a segment's M traces, both measured on real forward passes of Qwen2.5-7B-Instruct-1M over 20 Retr.KV contexts of median length 169,099 tokens, at $\rho = 0.1$ with the evaluation chunk and window. Every entry is averaged over each (segment, layer) pair of each context. The upper block lists the checkpoints whose accuracy Table 1 reports; the lower block lists checkpoints trained only for this analysis, whose accuracy is not reported. γ is the block-wise calibration target of §3. Both cosines are measured on the synthesized *keys*, which carry a per-segment positional anchor; the anchor lowers the across-segment cosine, so $\cos^K_{\text{seg}}$ understates how similar the segments' traces are. The anchor-free value-side comparison is given in the text.

	Variant	M	d_m	γ	N_{eff}	rank	$\cos^K_{\text{slot}}$	$\cos^K_{\text{seg}}$
In Tab. 1	**DuetKV**	4	128	3	711	1.38	0.969	0.619
	Evicted	4	128	3	838	1.34	0.977	0.620
	Uncalib.	4	128	–	13	1.69	0.933	0.617
	M=2	2	128	3	643	1.19	0.975	0.619
	M=8	8	128	3	896	1.55	0.966	0.628
Probe only	γ=2	4	128	2	2,980	1.23	0.985	0.624
	γ=4	4	128	4	161	1.47	0.961	0.615
	d_m=64	4	64	3	808	1.37	0.972	0.617
	d_m=256	4	256	3	878	1.28	0.984	0.609

Under the same calibration target, the *Evicted* constructor of §4.3 observes 97.6% of the segment positions at $\rho = 0.1$ yet aggregates 838 of them, compared with 711 for the default constructor. This intervention therefore changes both the available input support and the learned aggregation breadth, so its downstream effect cannot be attributed solely to restricting constructor access.

Within-segment diversity. Within each segment, the learned traces occupy a substantially lower-dimensional representation than their nominal cardinality suggests. The effective rank remains below 2 for every checkpoint in Table 5; along the trace-cardinality intervention of Table 1, increasing M fourfold from 2 to 8 raises the rank only from 1.19 to 1.55. The per-trace normalization of Eq. equation 7, together with the frozen key projection and shared segment anchor of Eq.equation 8, cannot increase the rank of the trace set, so this low-dimensional structure carries into the synthesized keys. Proposition A.6 further relates their second-order query dependence to a key covariance that vanishes in the collinear limit.

Downstream performance peaks at the intermediate cardinality, with aggregate scores of 51.8, 52.9, and 51.5 for $M = 2, 4, 8$, respectively. Because increasing M also consumes capacity that would otherwise retain exact KV states, the measured rank is consistent with, but does not by itself explain, the absence of further improvement at the largest cardinality.

Cross-segment sharing. Across segments, the synthesized values exhibit a mean pairwise cosine similarity of 0.993, compared with 0.987 for the corresponding per-segment means of the source values. Removing the component shared across segments leaves residuals accounting for 5.7% and 9.0% of their norms, respectively, with an across-context standard deviation of 0.0004. We report this comparison on values because synthesized keys undergo a segment-specific positional transformation whereas source keys retain their original positions, making the analogous key-side comparison difficult to interpret. The source segments are therefore already highly similar in value space, with the learned traces exhibiting an additional, consistent increase in cross-segment sharing.

This shared structure does not render the learned content dispensable. The *Random* control in §4.3 replaces each learned trace with a Gaussian vector of the same per-slot norm while preserving the number of traces, their cache charge, and the frozen projection pathway, reducing the aggregate score by 4.7 points (Table 1). Because this control perturbs the shared and segment-specific components jointly, however, it does not identify which component, or which interaction between them, accounts for the downstream predictive benefit.

Downstream utilization. At $\rho = 0.1$, latent traces receive 16.3% of context-directed attention while occupying 0.34% of retained cache entries; at $\rho = 0.2$, the corresponding shares are 11.6% and 0.15%. Both quantities use the normalization in Figure 8 and are computed exactly because the key segments of each head are contiguous under a single `softmax_lse`. Utilization varies by two orders of magnitude across depth: at $\rho = 0.1$, trace attention ranges from 0.77% at layer 17 to 80.5% at layer 24, and at $\rho = 0.2$ from 0.39% to 50.6% at the same layers. Layers 20, 21, and 24 receive the highest trace attention at both budgets, with a secondary maximum at layers 9–11; every layer assigns nonzero attention to the traces.

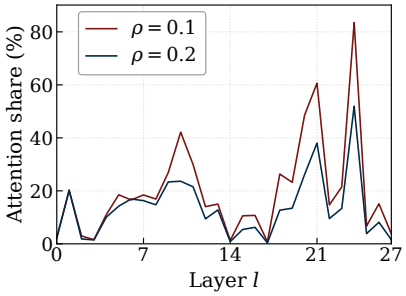


Figure 8: **Layer-wise attention to latent traces** on Retr.KV from SCBench with Qwen2.5-7B-Instruct-1M. Trace utilization varies substantially with depth and concentrates in later layers at both compression ratios.

Relative to the paired baseline, and measured over the full softmax rather than the context-directed group, the retained-context share decreases by 2.77 points while latent traces receive 2.36 points, leaving their combined attention share nearly unchanged. Yet the two forms of capacity are not interchangeable: restoring the exact capacity displaced by the traces improves the aggregate score by only 0.3 points, whereas replacing the learned trace content costs 4.7 points (§4.3). This asymmetry is consistent with the beneficial matched-budget exchange characterized by Theorem A.2: the latent pathway is actively read by the frozen backbone rather than merely occupying cache capacity.

Query-dependent routing. On Retr.KV, each query has a single evidence segment among the eleven segments of its context. We measure routing over 95 contexts, excluding the five whose evidence lies within the always-retained window. The context, rather than the query, is the unit of analysis because all five queries associated with a context share the same evidence segment.

Trace attention exhibits a strong query-independent concentration on the final segment, which receives 39.3% of trace attention at $\rho = 0.1$ and 43.4% at $\rho = 0.2$. This concentration is confined to the final segment rather than varying systematically with segment position: the other ten segments receive 4.0–7.7% at $\rho = 0.1$ and 3.7–7.7% at $\rho = 0.2$, with no ordering by position. We do not ascribe a mechanism to this concentration: the final segment is simultaneously the shortest, the one containing the protected window, and the one whose positional anchor lies closest to the query. Because the evidence segment can never be this final segment, its raw attention share, 8.6% at $\rho = 0.1$ and 7.7% at $\rho = 0.2$, is structurally biased downward relative to the 9.1% expected under uniform allocation, making a uniform null inappropriate.

Position-controlled comparisons nevertheless reveal evidence-dependent routing. Within each context, the evidence segment receives $+2.84$ and $+2.29$ points more trace attention than the mean of the other non-final segments at $\rho = 0.1$ and 0.2, respectively, with both differences separated from zero. Matching instead by segment index against contexts whose evidence occurs elsewhere yields nearly identical enrichments of $+2.83$ and $+2.28$ points. After excluding the final segment, top-1 trace attention falls on the evidence segment in 36.8% of contexts at $\rho = 0.1$ and 33.7% at $\rho = 0.2$, compared with 10% under chance.

A negative control further separates this query-dependent enrichment from the final-segment concentration. Re-evaluating 24 contexts at $\rho = 0.1$ with a query borrowed from another context, whose key does not occur in the current text, leaves the final-segment share essentially unchanged ($38.8\% \rightarrow 38.5\%$) but reduces evidence-segment enrichment from $+2.47$ points, separated from zero, to $+0.18$ points, which is not. The paired reduction of 2.29 points is itself separated from zero, and top-1 selection returns to chance (8.3%). The borrowed query also fails to increase attention to the segment containing its evidence in the donor context (-0.86 points), ruling out a fixed positional preference carried by the query. Thus, query-dependent routing toward the evidence-bearing segment coexists with, and is smaller than, the query-independent concentration on the final segment.