SWIFT: SECOND-ORDER WEIGHTED IMPORTANCE FOR TOKENS

Anonymous authors
Paper under double-blind review

ABSTRACT

KV-cache eviction reduces the memory and bandwidth cost of long-context inference, but its effectiveness depends on identifying cached entries whose physical removal minimally changes model behavior. Existing methods typically address the physical removal perturbation, downstream sensitivity, and global budget allocation separately, making eviction scores difficult to compare across layers and KV heads. We introduce SWIFT, a task-free second-order KV-cache eviction method grounded in a self-fidelity objective. Using only the cached context, full-coverage observations define fidelity against detached outputs of the uncompressed model, while exact softmax-renormalized removal perturbations and matrix-free scalar probes estimate second-order output damage without materializing the Hessian. These globally comparable scores enable SWIFT to allocate a shared physical KV budget across layers, KV heads, and tokens. Experiments cover $1{,}792$ controlled interventions and comparisons across three 7–8B model families and four long-context benchmarks. SWIFT achieves Pearson and Spearman correlations of 0.9957 and 0.9948 with measured eviction damage and ranks first or joint-first in 22 of 24 primary model–benchmark–retention comparisons. At 20% retention, it provides $5\times$ physical KV compression, delivers a 9.7–14.1% steady-state decoding speedup, and realizes approximately 80% of the empirically attainable historical-KV latency reduction.

1 INTRODUCTION

Large language models (LLMs) increasingly rely on long contexts for document understanding, repository-scale code analysis, multi-turn interaction, and persistent streaming applications (Bhaskar et al., 2025; Xiao et al., 2024). During autoregressive decoding, key–value (KV) caching avoids recomputing attention states for previous tokens, but stores those states in GPU memory. Its footprint grows linearly with sequence length and batch size, while repeated accesses consume memory bandwidth and increase decoding latency (Zhang et al., 2023; Jo & Shin, 2024). KV-cache eviction addresses this bottleneck by physically removing selected entries. Reliable eviction therefore depends on estimating the importance of each cached token before it is removed.

Existing policies infer importance from different signals. Structure-based policies retain fixed subsets. StreamingLLM and LM-Infinite preserve initial and recent tokens (Xiao et al., 2024; Han et al., 2024), while RazorAttention and DuoAttention assign different cache policies to different attention heads (Tang et al., 2024; Xiao et al., 2025). Content-adaptive policies instead respond to the current context. H2O, Scissorhands, A2SF, Keyformer, and SnapKV derive saliency from attention statistics or observation windows (Zhang et al., 2023; Liu et al., 2023; Jo & Shin, 2024; Adnan et al., 2024; Li et al., 2024). PyramidKV and Ada-KV additionally allocate budgets across layers or attention heads (Cai et al., 2024; Feng et al., 2026). K-Norm and KeyDiff use key magnitude or geometry (Devoto et al., 2024; Park et al., 2026), while CAOTE and LAVa score contributions to the attention output (Goel et al., 2025; Shen et al., 2025). Recent methods move toward more direct fidelity signals. KVGrad uses gradient-based global importance scoring (Kwak et al., 2026), OBCache adopts a curvature-inspired local objective (Gu et al., 2025), KVzip reconstructs cached context (Kim et al., 2026), and KVSlimmer performs asymmetric KV merging (Liu et al., 2026).

Despite this progress, reliable physical eviction requires three properties. The score should reflect the perturbation induced by actual KV removal, including the resulting softmax renormalization.

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.

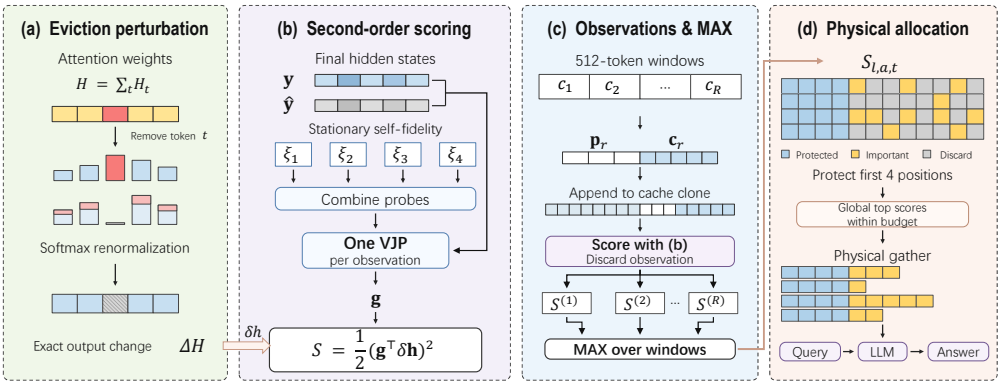

Figure 1: Overview of SWIFT. (a) Exact softmax-renormalized perturbation caused by evicting one KV pair. (b) Task-free stationary self-fidelity scoring with matrix-free second-order probes. (c) Full-context coverage using repeated observation windows and max aggregation. (d) Global physical-budget allocation followed by KV gathering for downstream queries.

It should measure how that perturbation affects the model output rather than only a local attention statistic. It should also remain comparable across layers, KV heads, and tokens so that a shared physical budget can be allocated globally. Existing methods typically capture only a subset of these requirements, such as local attention utility, downstream sensitivity, or budget allocation.

We introduce SWIFT, a task-free second-order method for KV-cache eviction. SWIFT constructs full-coverage observations using only the cached context, without task-specific questions, target continuations, answers, or labels. It defines a self-fidelity loss against detached outputs from the uncompressed model. At this operating point, the objective is stationary, so its first-order term vanishes and the leading approximation to eviction damage is second order. Matrix-free scalar probes estimate this damage for each candidate perturbation. Combined with the exact softmax-renormalized perturbation from physical KV removal, these estimates enable global cache allocation. The selected cache can therefore be constructed before future queries and reused across requests.

We validate SWIFT at three levels. In a controlled diagnostic with $1{,}792$ singleton interventions, its saliency achieves log-scale Pearson and Spearman correlations of 0.9957 and 0.9948 with measured eviction damage. Across three 7–8B model families and four long-context benchmarks, SWIFT ranks first or joint-first in 22 of 24 primary comparisons. At 20% retention, it provides $5\times$ physical KV compression, delivers a 9.7–14.1% steady-state decoding speedup, and realizes approximately 80% of the empirically attainable historical-KV latency reduction.

Our main contributions are summarized as follows:

- We formulate physical KV eviction as a second-order self-fidelity problem. SWIFT uses the exact softmax-renormalized perturbation at the attention endpoint and estimates its downstream output damage under a stationary fidelity objective.
- We develop a task-free full-coverage estimator that aggregates repeated observations of the cached context. Matrix-free scalar probes produce globally comparable second-order scores without materializing the Jacobian or Hessian, making the estimator practical for long-context eviction.
- We allocate a shared physical KV budget over the retained candidates and realize the resulting masks through physical KV gathering. Across three 7–8B models and four long-context benchmarks, SWIFT improves fidelity and task performance while delivering measured physical KV compression and steady-state decoding speedups.

2 SWIFT

2.1 Token Removal as an Attention Perturbation

For one attention head, let $Q \in \mathbb{R}^{n\times d}$ contain the observation queries and $K, V \in \mathbb{R}^{N\times d}$ contain all KV pairs visible to them. As shown in Fig. 1(a), SWIFT models physical KV removal through

exact softmax renormalization:

$$A = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}} + M\right), \qquad H = AV. \tag{1}$$

where M is the causal attention mask. The attention output decomposes over visible positions as

$$H = \sum_{t=1}^{N} H_t, \qquad H_t = A_{:,t} V_{t,:}. \tag{2}$$

Here H_t is the contribution of cached token t under the original attention distribution. It is not the perturbation caused by removing that token because removal changes the softmax normalizer.

For a historical token t, let $H^{(-t)}$ denote the attention output after the token is removed. At query position i, the remaining attention weights are renormalized by $1 - A_{it}$, giving

$$H^{(-t)}_{i,:} = \frac{H_{i,:} - A_{it} V_{t,:}}{1 - A_{it}}. \tag{3}$$

Thus the exact local perturbation induced by eviction is

$$\Delta H_{t,i} = H^{(-t)}_{i,:} - H_{i,:} = \frac{A_{it}}{1 - A_{it}} \left(H_{i,:} - V_{t,:}\right). \tag{4}$$

Equation (4), rather than $-H_t$, is the intervention scored by SWIFT. It exactly accounts for the softmax compensation at the attention endpoint. The derivation for removing an arbitrary set of tokens is given in the supplementary material.

2.2 Second-Order Weighted Token Importance

For fixed input $\mathbf{w}$ and an attention state $\mathbf{h}$ affected by KV eviction, define

$$\mathbf{y} = f(\mathbf{w}, \mathbf{h}) \in \mathbb{R}^C, \qquad \hat{\mathbf{y}} = \mathrm{sg}(\mathbf{y}), \tag{5}$$

where f maps $\mathbf{h}$ to the selected outputs with the input and model parameters fixed, and sg denotes stop-gradient. In our implementation, $\mathbf{y}$ contains the final hidden representations at the observation positions. Figure 1(b) summarizes this task-free scoring stage, where the dense hidden state and its detached reference define the self-fidelity objective. We define the normalized self-fidelity loss

$$\mathcal{L}(\mathbf{h}) = \frac{1}{2C} \left\|\mathbf{y} - \hat{\mathbf{y}}\right\|_2^2 = \frac{1}{2C} \sum_{c=1}^{C} (y_c - \hat{y}_c)^2. \tag{6}$$

The detached reference is formed at the dense operating point and held fixed when evaluating a local perturbation. Since $\mathbf{y} = \hat{\mathbf{y}}$ at this point, the fidelity residual and its first derivative vanish:

$$\frac{\partial \mathcal{L}}{\partial h_i} = \frac{1}{C} \sum_{c=1}^{C} (y_c - \hat{y}_c) \frac{\partial y_c}{\partial h_i} = 0. \tag{7}$$

The mixed second derivative is therefore

$$\frac{\partial^2 \mathcal{L}}{\partial h_i \, \partial h_j} = \frac{1}{C} \sum_{c=1}^{C} \frac{\partial y_c}{\partial h_i} \frac{\partial y_c}{\partial h_j}. \tag{8}$$

With $J = \partial \mathbf{y} / \partial \mathbf{h}$, Eq. (8) gives $\nabla^2_{\mathbf{h}} \mathcal{L} = C^{-1} J^\top J$. Hence, for an arbitrary attention-output perturbation $\delta \mathbf{h}$, the Taylor expansion at the dense operating point becomes

$$\Delta \mathcal{L} \approx \frac{1}{2} \delta \mathbf{h}^\top \nabla^2_{\mathbf{h}} \mathcal{L} \, \delta \mathbf{h} = \frac{1}{2C} \delta \mathbf{h}^\top J^\top J \delta \mathbf{h} = \frac{1}{2C} \left\|J \delta \mathbf{h}\right\|_2^2. \tag{9}$$

This quadratic form applies to any differentiable intervention variable. SWIFT instantiates it at the attention endpoint affected by physical KV eviction.

Explicitly constructing J or the Hessian is unnecessary. For $P = 4$ independent Rademacher vectors $\boldsymbol{\xi}_p \in \{-1, +1\}^C$, SWIFT forms

$$\bar{\boldsymbol{\xi}} = \frac{1}{\sqrt{P}} \sum_{p=1}^{P} \boldsymbol{\xi}_p, \qquad \phi = \frac{1}{\sqrt{C}} \bar{\boldsymbol{\xi}}^\top \mathbf{y}, \qquad \mathbf{g} = \nabla_{\mathbf{h}} \phi. \tag{10}$$

Since $\mathbb{E}[\bar{\boldsymbol{\xi}}\bar{\boldsymbol{\xi}}^\top] = I$, SWIFT directly defines the saliency of the candidate perturbation as

$$S = \frac{1}{2}\left(\mathbf{g}^\top \delta\mathbf{h}\right)^2, \qquad \mathbb{E}[S] = \frac{1}{2C}\left\|J\delta\mathbf{h}\right\|_2^2. \tag{11}$$

This matrix-free estimate preserves all cross-coordinate terms within the candidate perturbation and requires one vector–Jacobian product. The inner product is evaluated directly from attention probabilities, values, and the output projection, without materializing either the Hessian, the Jacobian, or all candidate perturbations.

2.3 Task-Free Full-Coverage Observations

A single terminal observation can miss evidence distributed across a long context. As shown in Fig. 1(c), we partition the history $\mathbf{x}$ into non-overlapping 512-token segments $\{\mathbf{c}_r\}_{r=1}^{R}$ and construct the task-free repeat input

$$\mathbf{w}_r = [\mathbf{p}_r; \mathbf{c}_r]. \tag{12}$$

The instruction prefix $\mathbf{p}_r$ asks the model to repeat the corresponding segment. For $r > 1$, it additionally includes an 8-token anchor from the end of $\mathbf{c}_{r-1}$. Each $\mathbf{w}_r$ is appended only to a clone of the full dense context cache and discarded after scoring. It never enters the final KV budget.

For layer l, KV head a, historical position t, and observation r, the single-token perturbation from Eq. (4) is formed for all query heads sharing KV head a, passed through the output projection, and vectorized as $\delta\mathbf{h}_{l,a,t}^{(r)}$. Restoring the indices suppressed in the general derivation gives

$$S_{l,a,t}^{(r)} = \frac{1}{2}\left(\mathbf{g}_l^{(r)\top}\delta\mathbf{h}_{l,a,t}^{(r)}\right)^2. \tag{13}$$

The inner product jointly aggregates all observation positions, feature coordinates, and GQA query heads associated with a candidate before squaring. Different KV heads remain separate candidates and are not merged into one layer–token score.

The repeat text constructs observations rather than a supervised objective: SWIFT never differentiates reconstruction cross-entropy or uses future questions, continuations, or answers during scoring. We aggregate evidence from complete context coverage by

$$S_{l,a,t} = \max_{r\in\{1,\ldots,R\}} S_{l,a,t}^{(r)}. \tag{14}$$

The maximum retains strong deletion evidence from any context region instead of diluting localized evidence across unrelated observations.

2.4 Physical KV-Pair Allocation

As shown in Fig. 1(d), SWIFT globally ranks all layer–KV-head–token candidates under a shared physical budget. We reserve the first four sink-token positions in every layer and KV head, then retain the highest-scoring remaining pairs until the total number of retained pairs reaches the target budget. This one-shot allocation allows different layers and KV heads to retain different subsets of the history. In our experiments, the saliency scores are supplied to the official physical-cache implementation of KVzip, which physically gathers the retained KV pairs. We report the realized physical retention in addition to the requested target ratio.

For a length-N context, SWIFT processes $R = \lceil N/512 \rceil$ observations sequentially, releases each computation graph after scoring, and stores only $O(LH_{\mathrm{KV}}N)$ saliency values. The resulting scores are computed once and reused across retention ratios and subsequent queries over the same context.

3 Experiments

We evaluate SWIFT along five axes: saliency consistency, language-model fidelity, long-context generalization, multi-query robustness, and physical deployment. Experiments span three 7–8B model families and representative KV-cache compression baselines.

All methods start from the same dense checkpoints and frozen model-specific inputs. Selection uses only the context, while task questions and reference answers remain hidden. We compare compressed methods at matched realized physical KV retention and use the Full cache only as an uncompressed reference. Unless stated otherwise, evaluation physically gathers the retained KV pairs rather than applying a logical attention mask.

We compare with StreamingLLM (Xiao et al., 2024), SnapKV (Li et al., 2024), PyramidKV (Cai et al., 2024), KeyDiff (Park et al., 2026), OBCache (Gu et al., 2025), KVGrad (Kwak et al., 2026), KVzip (Kim et al., 2026), KVzip+ (Jegou & Jeblick, 2026), and KVSlimmer (Liu et al., 2026), together with Random eviction.

3.1 Saliency Consistency Validation

We first isolate the accuracy of SWIFT's second-order damage estimator in a controlled diagnostic. Using Qwen3-0.6B (Yang et al., 2025) and a 512-token C4 context (Raffel et al., 2020), we protect the first four and final 32 positions and uniformly sample 64 positions from the remaining history. For every layer and sampled position, we mask that position across all attention heads and measure the resulting continuation-logit damage. This produces $28 \times 64 = 1{,}792$ matched layer–position interventions. We compare the measured damage with the prediction obtained from the exact softmax-renormalized perturbation and a high-fidelity 32-probe estimate.

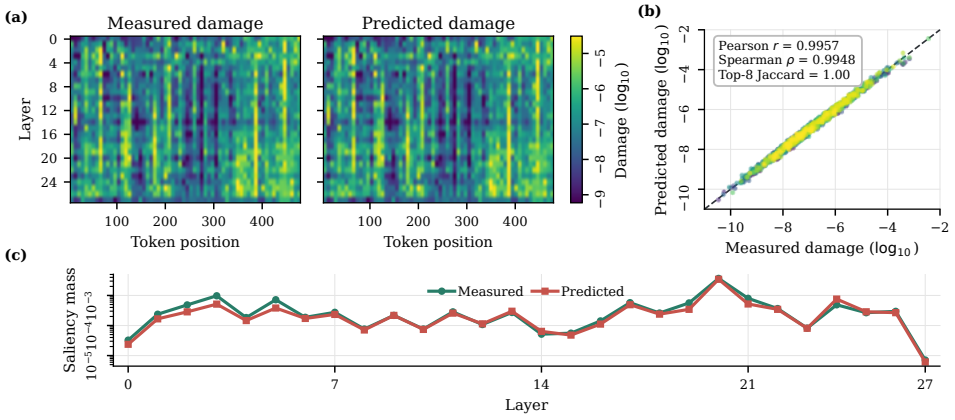

Figure 2: Consistency of SWIFT's second-order damage estimator on Qwen3-0.6B. We uniformly sample 64 removable positions from a 512-token C4 context and intervene on each position across all 28 layers. (a) Measured and predicted damage over the resulting 1,792 layer–position interventions. (b) Pointwise agreement between the two maps. (c) Their aggregate saliency mass across layers.

Figure 2(a–b) shows close agreement between predicted and measured damage. Across all 1,792 interventions, SWIFT achieves a log-scale Pearson correlation of 0.9957 and a Spearman correlation of 0.9948. The median same-layer Top-8 Jaccard is 1.00, showing that the estimator recovers the highest-damage positions within a typical layer. The analytical attention perturbation also has a median relative error of only 8.16×10^{-5} against the measured endpoint change.

Figure 2(c) further reveals a $518.4\times$ range in measured saliency mass across layers. Layers with greater mass require more KV capacity to avoid large output damage, while low-mass layers can tolerate more aggressive eviction. This heterogeneity argues against a uniform per-layer quota and motivates SWIFT's global ranking of layer–KV-head–token candidates. Subsequent experiments test whether this local agreement yields faithful compression and robust downstream accuracy.

3.2 Faithful Compression on PG19

We evaluate physical KV compression on PG19 (Rae et al., 2019) using Qwen3-8B, Mistral-7B-v0.3 (Jiang et al., 2023), and Llama-3.1-8B (Grattafiori et al., 2024). Each model processes 20 deterministic windows with a 4K-token context and a held-out 256-token continuation. Selection uses only the context, and all methods physically gather KV at matched retention. We report the

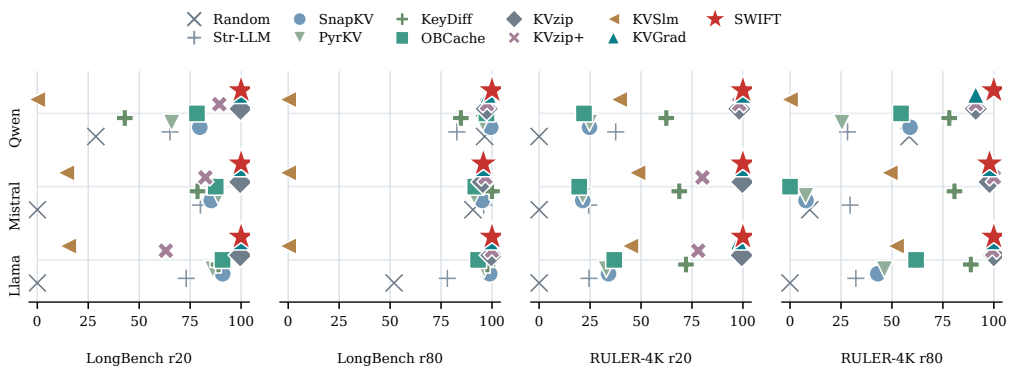

Figure 3: Normalized scores on LongBench English-16 and RULER-4K at 20% and 80% KV retention. Each setting is min–max normalized over compressed methods, excluding Full.

perplexity ratio to Full and token-averaged $D_{\mathrm{KL}}(p_{\mathrm{Full}} \| p_{\mathrm{Compressed}})$ over the continuation, where lower values indicate better fidelity.

Table 1 reports results at 20% and 80% retention. SWIFT ranks first in 9 of 12 model–retention–metric comparisons, including all six at 20% retention. Under this aggressive $5\times$ compression, it obtains suffix KL values of 0.0413, 0.0288, and 0.0285 across the three models, outperforming every competing method and consistently improving over the output-aware KVGrad, KVzip, and OBCache baselines. Its saliency accuracy therefore translates into faithful continuation after removing 80% of historical KV.

At 80% retention, perplexity differences narrow as most KV is preserved. SWIFT nevertheless achieves the lowest suffix KL on all three models at 0.0011, 0.0004, and 0.0010, while keeping perplexity ratios close to Full. SWIFT therefore preserves Full-cache behavior under both aggressive and near-dense retention.

Table 1: Physical KV compression on 20 PG19 windows. Each entry reports perplexity ratio ↓ / token-averaged suffix KL ↓ at realized retention r. Bold and underlined values mark the best and second-best compressed methods for each metric. Full is excluded from ranking. Str-LLM, PyrKV, and KVSlm denote StreamingLLM, PyramidKV, and KVSlimmer.

	Qwen3-8B		Mistral-7B-v0.3		Llama-3.1-8B	
Method	$r = 20\%$	$r = 80\%$	$r = 20\%$	$r = 80\%$	$r = 20\%$	$r = 80\%$
Full	1.0000 / 0.0000	1.0000 / 0.0000	1.0000 / 0.0000	1.0000 / 0.0000	1.0000 / 0.0000	1.0000 / 0.0000
Random	3.2498 / 1.3241	1.0594 / 0.0729	4.1015 / 1.3949	1.0836 / 0.0882	28.4515 / 3.3496	1.5609 / 0.4800
Str-LLM	1.0952 / 0.1110	1.0134 / 0.0141	1.0818 / 0.0719	1.0205 / 0.0141	1.1556 / 0.1665	1.0653 / 0.0848
SnapKV	1.0295 / 0.0558	0.9950 / 0.0021	1.0303 / 0.0336	1.0016 / 0.0024	1.0274 / 0.0380	1.0012 / 0.0017
PyrKV	1.0295 / 0.0558	0.9941 / 0.0022	1.0303 / 0.0336	1.0009 / 0.0016	1.0274 / 0.0380	1.0009 / 0.0016
KeyDiff	3.7708 / 1.4275	1.3504 / 0.3385	1.0457 / 0.0695	1.0025 / 0.0026	1.0845 / 0.0917	1.0022 / 0.0043
OBCache	1.0153 / 0.0523	0.9939 / 0.0019	1.0253 / 0.0316	1.0001 / 0.0019	1.0259 / 0.0361	**0.9995** / 0.0015
KVzip	1.1256 / 0.0977	0.9961 / 0.0013	1.0264 / 0.0336	**1.0000** / 0.0011	1.0484 / 0.0549	1.0008 / 0.0015
KVzip+	1.3801 / 0.3669	0.9942 / 0.0013	1.4573 / 0.3898	1.1604 / 0.1533	2.5351 / 0.9508	1.0012 / 0.0028
KVSlm	1.0584 / 0.0793	1.0011 / 0.0088	1.0271 / 0.0436	1.0007 / 0.0041	1.0433 / 0.0525	1.0031 / 0.0050
KVGrad	1.0544 / 0.0534	**0.9911** / 0.0101	1.0321 / 0.0482	1.0138 / 0.0163	1.0221 / 0.0301	1.0001 / 0.0013
SWIFT	**0.9890 / 0.0413**	0.9958 / **0.0011**	**1.0171 / 0.0288**	1.0002 / **0.0004**	**1.0172 / 0.0285**	1.0001 / **0.0010**

3.3 Long-context Generalization

We evaluate whether a cache selected without the downstream question generalizes across long-context tasks. We use LongBench English-16 (Bai et al., 2024) and RULER-4K (Hsieh et al., 2024) on Qwen3-8B, Mistral-7B-v0.3, and Llama-3.1-8B. We evaluate 160 LongBench examples across 16 tasks and 130 RULER examples across 13 tasks, reporting macro-average scores at 20% and 80% physical KV retention. Selection precedes the question and uses neither questions nor reference answers. Full is uncompressed and unranked.

Figure 3 compares the eleven compressed methods after within-setting min–max normalization. On LongBench at 20% retention, SWIFT ranks first for all three models. At 80%, it remains first on Qwen3-8B and Llama-3.1-8B, while ranking fifth on Mistral-7B-v0.3, where KVzip+ performs best.

Table 2: Aggregate ranks over 12 LongBench and RULER settings for eight leading methods among all 11 compressed methods. Counts include ties. Other denotes rank ≥ 3.

	SWIFT	KVGrad	KVzip	KVzip+	KeyDiff	SnapKV	OBCache	Str-LLM
1st	10	1	1	1	1	0	0	0
2nd	1	3	6	3	0	1	0	0
Other	1	8	5	8	11	11	12	12
Mean rank ↓	1.58	3.00	3.04	3.88	5.83	6.50	7.50	8.00

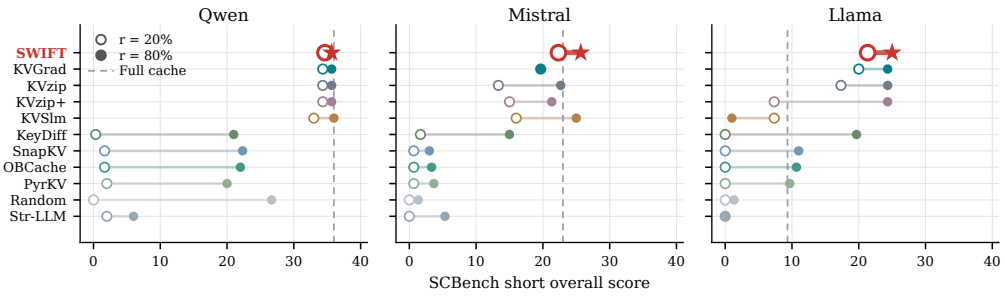

Figure 4: SCBench score under multi-query cache reuse, with each context compressed once. Hollow and filled markers denote 20% and 80% retention, and dashed lines mark Full.

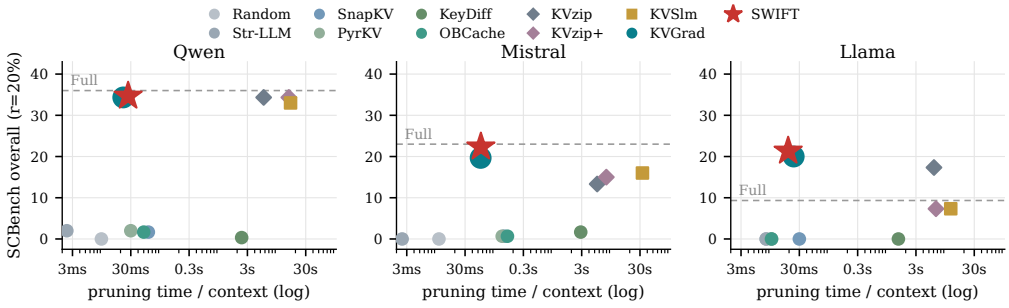

Figure 5: SCBench score versus per-context selection cost at 20% retention. Times are measured directly when available and otherwise estimated from matched prefill measurements. Dashed lines mark Full scores. Selection is a one-time cost and excludes generation.

Thus, SWIFT remains effective under aggressive compression even when the downstream question is unavailable during cache selection.

RULER-4K complements LongBench with retrieval, positional reasoning, and counting tasks. At 20% retention, SWIFT ranks first on Mistral-7B-v0.3 and Llama-3.1-8B, and ties for first with KVGrad on Qwen3-8B. At 80%, it ranks first on Qwen3-8B, ties for first with KVzip on Llama-3.1-8B, and remains competitive on Mistral-7B-v0.3. These results show transfer beyond natural language continuation to retrieval and structured reasoning.

Across the twelve model–benchmark–retention settings, SWIFT ranks first or joint-first in 10 settings, with an average fractional rank of 1.58, compared with 3.00 for KVGrad and 3.04 for KVzip. Table 2 summarizes the aggregate ranking. The results demonstrate that one query-agnostic cache can generalize across model families, task types, and compression budgets.

3.4 Multi-query Robustness

On SCBench short (Li et al., 2025), we evaluate 30 contexts and 200 queries. Each context is compressed once and reused across its queries, with queries and answers hidden during selection. We report overall score at 20% and 80% physical KV retention.

Figure 4 shows that SWIFT achieves the best compressed-cache score on all three models at 20% retention. At 80%, it remains best on Mistral-7B-v0.3 and Llama-3.1-8B and ties for best on Qwen3-8B. SWIFT therefore ranks first or joint-first in all six model–retention settings.

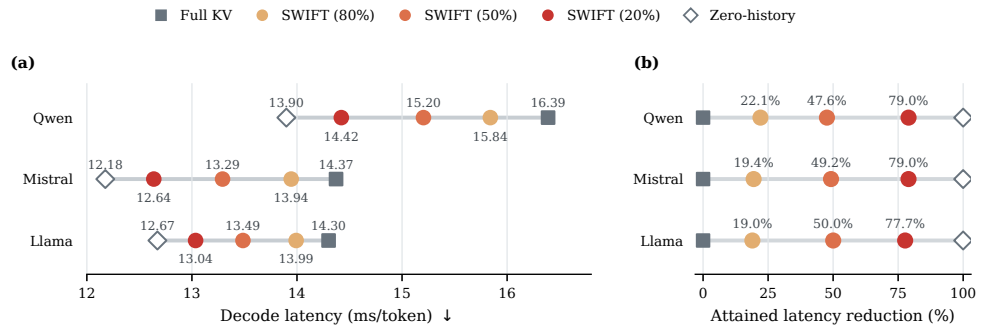

Figure 6: Physical KV deployment on Qwen3-8B (24.7K tokens), Mistral-7B-v0.3 (25.6K tokens), and Llama-3.1-8B (18.9K tokens). (a) Steady-state decode latency under Full, three SWIFT retention ratios, and zero-history. (b) Fraction of the Full-to-zero-history latency reduction attained by SWIFT. Zero-history is a latency reference, not a quality-valid method.

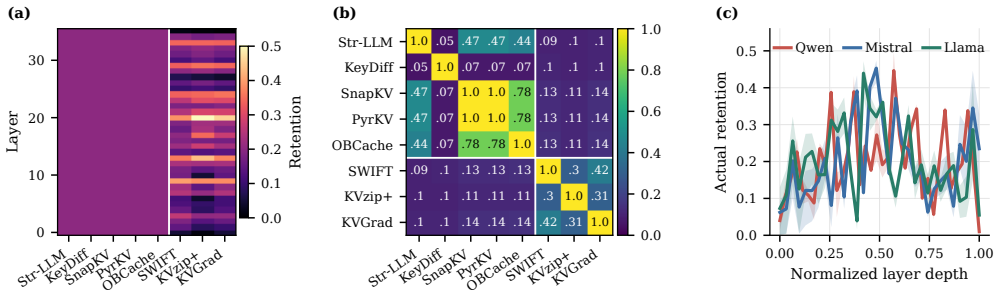

Figure 7: Realized KV allocation at $r = 20\%$ on PG19. (a) Layerwise retention on Qwen, separating layer-uniform and saliency-driven methods. (b) Mean token-mask Jaccard across three models and 20 documents. (c) Mean and range of the three saliency-driven profiles.

Full may retain redundant or distracting history that interferes with later queries. By removing low-importance KV while preserving output-important information, SWIFT can provide task-agnostic denoising. Consistent with this view, it surpasses Full on Mistral at 80% and on Llama at both retention levels, with large gains on Llama. This reflects task utility rather than fidelity to Full.

Figure 5 shows that SWIFT combines the highest score with low selection cost across all three models. Its context-level selection takes 19–142 ms, while the reconstruction-based KVzip variants require several seconds. Across models and baselines, SWIFT reduces selection cost by roughly one to three orders of magnitude. This one-time cost excludes end-to-end generation.

Overall, SWIFT selects one query-agnostic cache that remains effective across repeated queries while avoiding the high reconstruction cost of output-aware baselines.

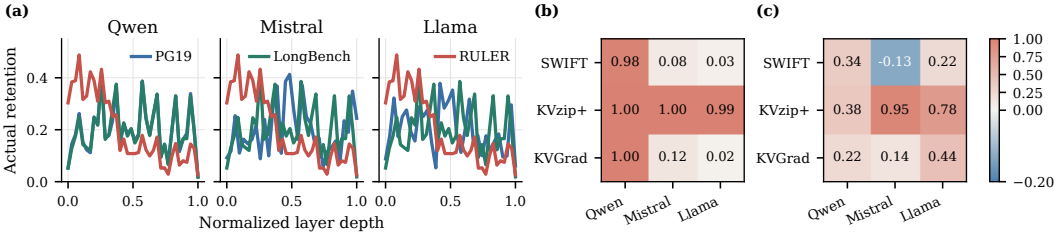

Figure 8: Task dependence of layer allocation at $r = 20\%$. (a) SWIFT profiles on PG19, LongBench, and RULER. (b, c) Spearman correlation of PG19 profiles with LongBench and RULER across methods and models.

3.5 Physical KV Deployment Benefits

We measure Full and SWIFT with the same layer-paged CUDA-Graph backend. For each request, we report median per-token latency over seven trials of 128-token greedy decoding after three warm-up runs. Timing begins after KV selection and gathering, query appending, and graph setup, thereby excluding one-time saliency computation.

Figure 6(a) shows that physical KV reduction consistently accelerates decoding. At 20% retention, SWIFT provides $5\times$ physical KV compression and a 9.7–14.1% steady-state decoding speedup. The paged KV allocation decreases from 3.42 to 0.71 GiB for Qwen3-8B, from 3.15 to 0.65 GiB for Mistral-7B-v0.3, and from 2.33 to 0.48 GiB for Llama-3.1-8B, confirming physical rather than logical cache reduction. Even at 8K, SWIFT delivers a 4.3–4.5% decoding speedup. The larger gains on longer contexts are consistent with a growing KV-access share.

To contextualize the remaining systems gap, we measure a zero-history reference that retains only query and generated-token KV. It serves solely as a latency reference and is not quality-valid. For latency T, we define $\eta = (T_{\mathrm{Full}} - T)/(T_{\mathrm{Full}} - T_{\mathrm{Zero}})$. As shown in Figure 6(b), SWIFT attains 77.7–79.0% of the empirical Full-to-zero-history latency gap at $r = 20\%$. The corresponding fractions are 47.6–50.0% at $r = 50\%$ and 19.0–22.1% at $r = 80\%$. Thus, SWIFT closes most of this gap at 20% retention while preserving the task-effective cache evaluated above. These results establish physical KV reduction and steady-state acceleration, but not first-request end-to-end speedup because one-time selection lies outside the timed region.

3.6 Allocation Structure and Task Adaptation

We analyze layer–token masks at $r = 20\%$ using 20 examples per model and benchmark. Figure 7(a) shows that fixed-budget methods retain KV uniformly across layers, whereas SWIFT, KVzip+, and KVGrad produce model-dependent ragged profiles. On PG19, their top three layers receive 16.0–20.8% of the budget, compared with 8.3–9.4% under uniform allocation. The ragged methods have pairwise Spearman profile correlations of 0.53–0.96, while SWIFT remains repeatable across documents within each model, with mean correlations of 0.90–0.95.

Related layer profiles do not imply identical token choices. Figure 7(b) shows that SWIFT's mean token-mask Jaccard is 0.42 with KVGrad, 0.30 with KVzip+, and only 0.09–0.13 with layer-uniform methods. Saliency-driven methods therefore agree on where layer capacity is needed but retain different tokens within those layers.

Figure 8 further shows that allocation depends on the model and task. SWIFT's PG19-to-LongBench correlations are $0.98/0.08/0.03$ across Qwen, Mistral, and Llama, and become $0.34/-0.13/0.22$ on RULER. On Qwen, RULER shifts the first-third budget from 0.31 to 0.61. Together with its competitive task scores, these shifts show that SWIFT adapts its budget across models and tasks while remaining repeatable within each setting.

4 Conclusion

SWIFT estimates KV-cache eviction damage under a task-free stationary self-fidelity objective. It combines exact softmax-renormalized attention perturbations with matrix-free second-order estimation and allocates a physical KV budget globally across the retained cache candidates.

Across three 7–8B models and four long-context benchmarks, SWIFT ranks first or joint-first in 22 of 24 primary settings. Its allocation is layer-ragged, repeatable within a setting, and adaptive across models and tasks. At 20% retention, SWIFT provides $5\times$ physical KV compression and delivers a 9.7–14.1% steady-state decoding speedup, reaching nearly 80% of the empirical Full-to-zero-history latency gap relative to a latency-only reference.

SWIFT requires one-time gradient-based scoring. Deployment results measure steady-state decoding after cache selection and setup, rather than first-request latency. We evaluate three 7–8B dense decoder-only models. Extending SWIFT to incremental selection, larger and mixture-of-experts models, and additional serving backends remains future work.

AI USE STATEMENT

We used generative AI tools to improve the clarity and language of the manuscript and to assist in producing selected statistical figures from our experimental results. The authors verified all underlying data, visualizations, technical claims, and AI-assisted text, and take responsibility for the final content of this work.

REPRODUCIBILITY STATEMENT

The main paper specifies the method, evaluation protocols, models, datasets, metrics, and retention settings. The supplementary material provides detailed derivations, implementation and environment configurations, exact results, and additional design studies. An anonymous code package containing the evaluation and plotting workflows accompanies the submission.

REFERENCES

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127, 2024.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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, 2024.

Adithya Bhaskar, Alexander Wettig, Tianyu Gao, Yihe Dong, and Danqi Chen. Cache me if you can: How many kvs do you need for effective long-context lms? *arXiv preprint arXiv:2506.17121*, 2025.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2 norm-based strategy for kv cache compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18476–18499, 2024.

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.

Raghavv Goel, Junyoung Park, Mukul Gagrani, Dalton Jones, Matthew Morse, Harper Langston, Mingu Lee, and Chris Lott. Caote: Kv cache selection for llms via attention output error-based token eviction. *arXiv preprint arXiv:2504.14051*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. In *Advances in Neural Information Processing Systems*, 2024.

Yuzhe Gu, Xiyu Liang, Jiaojiao Zhao, and Enmao Diao. Obcache: Optimal brain kv cache pruning for efficient long-context llm inference. *arXiv preprint arXiv:2510.07651*, 2025.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3991–4008, 2024.

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.

Simon Jegou and Maximilian Jeblick. Kvzap: Fast, adaptive, and faithful kv cache pruning. *arXiv preprint arXiv:2601.07891*, 2026.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023. URL https://arxiv.org/abs/2310.06825.

Hyun-rae Jo and Dongkun Shin. A2sf: Accumulative attention scoring with forgetting factor for token pruning in transformer decoder. *arXiv preprint arXiv:2407.20485*, 2024.

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, 2026.

Jihwan Kwak, Sunghwan Joo, Jung Yoon Hwang, Jaeseok Byun, and Taesup Moon. KVgrad: Query-agnostic KV cache eviction via gradient-based global importance scoring. In *AdaptFM: Resource-Adaptive Foundation Model Inference*, 2026. URL https://openreview.net/forum?id=cg1wTCJjjk.

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.

Lianjun Liu, Hongli An, Weiqi Yan, Xin Du, Shengchuan Zhang, Huazhong Liu, and Yunshan Zhong. Kvslimmer: Theoretical insights and practical optimizations for asymmetric kv merging. *arXiv preprint arXiv:2603.00907*, 2026.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

Junyoung Park, Dalton Jones, Matthew Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments. *Advances in Neural Information Processing Systems*, 38:5983–6019, 2026.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

Yiqun Shen, Song Yuan, Zhengze Zhang, Xiaoliang Wang, Daxin Jiang, and Nguyen Cam-Tu. Lava: Layer-wise kv cache eviction with dynamic budget allocation, 2025. URL https://arxiv.org/abs/2509.09754.

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

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

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*, 2025.

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