STRATIFIED SPARSE ATTENTION: REDUCING THE SPARSE-TO-DENSE GAP AT EXTREME SPARSITY

Anonymous authors
Paper under double-blind review

ABSTRACT

Large Language Model inference over long contexts is increasingly bottlenecked by the cost of attending to a growing Key-Value (KV) cache. Sparse attention reduces this cost by accessing only a fraction of the cache, but existing methods face a trade-off between deterministic retrieval, which captures high-scoring tokens but discards residual attention mass, and sampling, which improves coverage at the cost of higher variance. Hybrid methods combine top-k retrieval with uniform sampling, yet still exhibit a substantial gap from dense attention at aggressive sparsity. We introduce *Stratified Sparse Attention (SSA)*, a unified framework combining deterministic retrieval with score-guided sampling. SSA deterministically attends to high-scoring tokens while sampling the residual using proxy attention scores and correcting for their inclusion probabilities, thereby recovering attention mass beyond the retrieval set. SSA encompasses top-k, top-p, and hybrid top-k+sampling as special cases. Across RULER, LongBench, LOFT, and AIME, and five models ranging from 3B to 27B parameters, SSA consistently improves over the baselines at matched sparsity. On RULER-HARD, SSA improves accuracy over the strongest baseline from 72.5% to 83.7% at 32K and from 69.0% to 76.8% at 128K over the strongest baseline under $50\times$ sparsity, recovering 94% and 91% of dense-attention performance, respectively. With optimized CUDA kernels, SSA achieves up to $2.96\times$ decode speedup over FlashInfer at 1M context, demonstrating that its accuracy gains translate into practical long-context inference acceleration.

1 INTRODUCTION

Large language models (LLMs) are rapidly expanding into long-context workloads, spanning large document collections (Li et al., 2023), image analysis (Dosovitskiy et al., 2021; Peebles & Xie, 2023), software engineering (Hou et al., 2024; Chen et al., 2024), databases (Zhou et al., 2024b), video processing (Ruan & Jin, 2022), and long-horizon agents that continuously accumulate observations, tool outputs, and interaction histories (Cao et al., 2026; Chen et al., 2026). As context lengths approach millions of tokens, efficient inference becomes increasingly challenging. During autoregressive decoding, each newly generated token attends to an ever-growing key-value (KV) cache, requiring keys and values to be repeatedly transferred from GPU high-bandwidth memory (HBM) onto on-chip SRAM for attention computation (Shah et al., 2024; Kwon et al., 2023). At long contexts, this makes decoding increasingly memory-bandwidth bound: the attention cost per decoding step grows linearly with context length. Sparse attention addresses this bottleneck by attending to only a small fraction of the KV cache (Tang et al., 2024; Liu et al., 2025; Zhang et al., 2025), reducing both memory traffic and attention latency. However, realizing these gains in practice requires sparse attention to satisfy two competing objectives: *faithfully approximate dense attention at aggressive sparsity* while remaining *efficient enough to translate sparsity into end-to-end inference speedups* (Pagliardini et al., 2023; Lin et al., 2025).

Existing sparse-attention methods can be broadly categorized by how they allocate a limited attention budget. First, *approximate retrieval* methods use proxy attention scores to deterministically select high-scoring tokens, typically through top-k retrieval. Representative methods include PQ-Cache (Zhang et al., 2025), and Quest (Tang et al., 2024), with recent approaches such as Tactic (Zhu et al., 2026) extending this paradigm to top-p selection. While effective when attention is highly concentrated, deterministic retrieval can miss substantial attention mass when the distribution is diffuse

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.

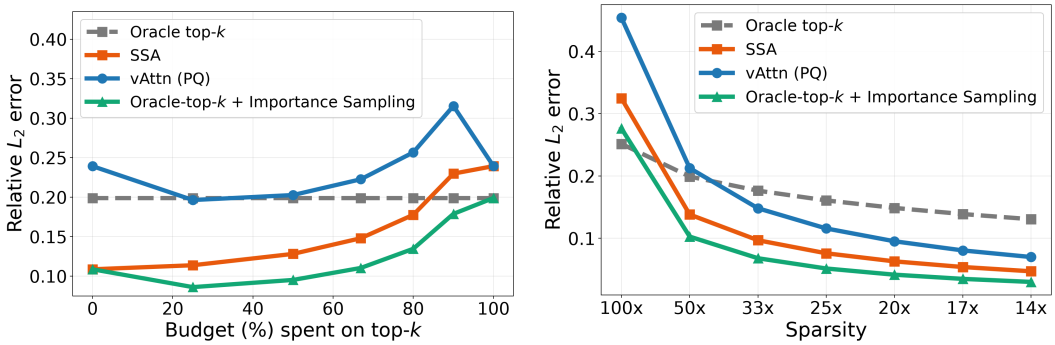


Figure 1: Relative L_2 error when varying the top-k:sampling budget allocation at $50\times$ sparsity (left) and varying sparsity with a fixed 3:2 top-k:sampling budget split (right).

or long-tailed (Chen et al., 2025; Desai et al., 2026). Second, *sampling-based* methods such as MagicPig (Chen et al., 2025) use importance sampling to approximate dense attention, providing broader coverage but introducing sampling variance that can lead to a larger gap from dense attention at limited budgets (see Table 4 in Appendix). Finally, *hybrid* methods such as vAttention (Desai et al., 2026) combine deterministic top-k retrieval with uniform sampling (Vitter, 1985), preserving high-confidence tokens while extending coverage beyond the top-k set. Although this combination improves the trade-off between exploitation and coverage, a substantial gap to dense attention remains at aggressive sparsity (see Table 4 in Appendix). See Section 2 for more details. These observations motivate a fundamental question: *Can we design a more effective sampling strategy that substantially narrows the sparse-to-dense accuracy gap?*

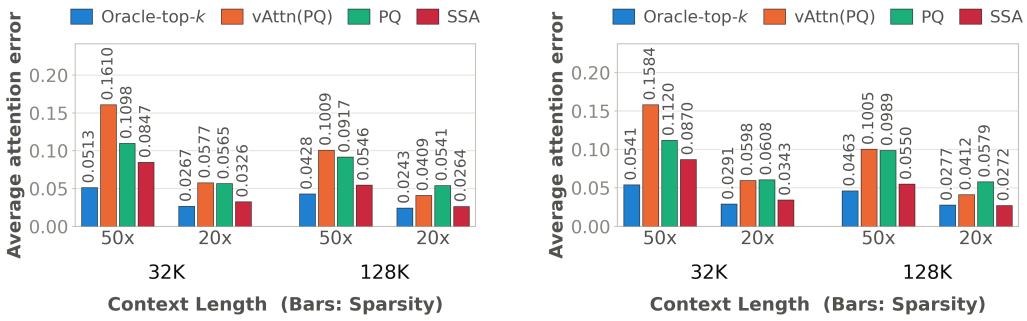
Our preliminary studies reveal a key limitation of uniform sampling: while it provides coverage beyond the deterministic top-k set, it ignores the highly non-uniform distribution of residual attention mass[1]. As shown in fig. 1, score-guided importance sampling (the principle that SSA uses) consistently achieves lower relative L_2 error than uniform sampling. Motivated by this observation, we introduce **Stratified Sparse Attention (SSA)**, a unified framework that combines deterministic retrieval of high-confidence heavy hitters with *score-guided stochastic exploration* of the remaining context. Rather than treating all residual tokens equally, SSA uses proxy attention scores to preferentially explore promising regions while preserving stochastic coverage of the long tail. This enables the sparse attention budget to better capture the attention mass that would otherwise be lost under aggressive sparsity. We evaluate SSA across diverse models, including Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3.5-4B, Qwen3.5-9B, Ministral-3-3B-Instruct-2517, and Ministral-3-8B-Instruct-2517 (Liu et al., 2026), and across RULER Hsieh et al. (2024), LongBench Bai et al. (2024), LOFT (Lee et al., 2024), and AIME Mathematical Association of America (2025). Across these evaluations, SSA consistently improves over deterministic, sampling-based, and hybrid sparse-attention baselines, substantially narrowing the gap to dense attention at extreme sparsity. Our contributions can be summarized as follows:

I. Unified Attention Paradigm: We introduce a unified sparse-attention framework that partitions tokens into deterministic and stochastic strata using a generic proxy score. By varying how tokens are retained or sampled, the framework recovers top-k, top-p, and hybrid top-k+sampling as special cases of importance-sampling-inspired sparse attention.

II. Theoretical Guarantees: We provide rigorous bounds for our proposed algorithm, characterizing how closely it approximates dense attention under a limited attention budget.

III. Efficient Kernel Support: We develop efficient CUDA/Triton kernels that realize the computational benefits of our approach in practice, delivering end-to-end decoding speedups over dense attention.

[1]Oracle-top-k + Importance Sampling uses exact attention scores for both top-k selection and residual importance sampling, providing an oracle reference for SSA.



(a) Qwen3.5-4B: Error

(b) Qwen3.5-9B: Error

Figure 2: Average attention error across different context lengths and sparsity levels for Qwen3.5-9B and Qwen3.5-4B on LOFT (Lee et al., 2024).

2 Related Work

In the context of SSA, the extensive prior work can be organized into three broad and comprehensive research directions:

Head-level compression and KV eviction: One line of work reduces KV-cache overhead by sharing or compressing KV heads. Multi-Query Attention (MQA) (Shazeer, 2019) shares a single KV head across all query heads, while Grouped-Query Attention (GQA) (Ainslie et al., 2023) and Multi-Head Latent Attention (MLA) (DeepSeek-AI, 2024) provide more flexible compression to balance efficiency and model quality. A complementary line of work dynamically evicts less important KV entries during inference. Methods such as SnapKV (Li et al., 2024), DynamicKV (Zhou et al., 2024a), KVZip (Kim et al., 2025), KeyDiff (Park et al., 2025), CurDKV (Sengupta et al., 2025), and CapKV (Yang et al., 2026) use different importance signals and eviction policies to reduce cache size and subsequent attention computation. These approaches are complementary to sparse attention: KV compression reduces the tokens or representations retained across decoding steps, whereas sparse attention dynamically selects which retained tokens to access for each query. The two can therefore be composed to further reduce memory and decoding costs.

Approximate top-k/top-p methods: A substantial body of sparse-attention work focuses on finding the tokens with the largest attention scores. Methods such as Quest Tang et al. (2024), PQ-Cache Zhang et al. (2025), SqueezedAttention (Hooper et al., 2025), Adamas (Yan et al., 2025), SOCKET Joshi et al. (2026), Sketch-and-Walk Le et al. (2026), RetrievalAttention Liu et al. (2025), Loki Singhania et al. (2024), HashAttention Desai et al. (2025), and Double Sparsity Yang et al. (2024) employ different proxy scores or retrieval mechanisms to approximate the top-k tokens under dense attention. These methods are effective at recovering high-attention "heavy hitters," but a fixed top-k selection discards all tokens outside the retrieval set. Consequently, when non-negligible attention mass is distributed across the residual tail, or when approximation error causes relevant tokens to fall just below the top-k boundary, this discarded mass can create a substantial gap from dense attention at aggressive sparsity. Tactic Zhu et al. (2026) extends this retrieval paradigm from top-k to top-p selection by modeling the decay of attention scores with a power-law distribution and estimating the number of highest-scoring tokens required to satisfy a target attention-mass threshold.

Sampling-based and Hybrid methods: MagicPig Chen et al. (2025) addresses the limitations of deterministic top-k attention by using LSH Indyk & Motwani (1998) to approximately sample tokens and importance correction to estimate dense attention. vAttention Desai et al. (2026) combines deterministic and stochastic selection through a verified hybrid estimator that retains top-k tokens while uniformly sampling from the residual KV cache. KNN-Attention (Haris, 2025) also explores top-k plus sampling via lazy Gumbel sampling, but primarily targets training rather than inference acceleration for existing models. Unlike uniform hybrid sampling, which treats all residual tokens equally, SSA uses approximate attention scores to guide stochastic selection toward tokens more likely to carry residual attention mass (see fig. 1).

3 BACKGROUND AND MOTIVATION

3.1 SPARSE ATTENTION

We begin by describing the dense attention computation. For clarity, we consider a single query $\mathbf{q} \in \mathbb{R}^d$ attending to a KV cache $\mathbf{K} := (\mathbf{k}_1, \ldots, \mathbf{k}_N)^\top \in \mathbb{R}^{N \times d}$ and $\mathbf{V} := (\mathbf{v}_1, \ldots, \mathbf{v}_N)^\top \in \mathbb{R}^{N \times d}$.

Full Scaled Dot-Product Attention (SDPA): Let $s_i = \mathbf{q}^\top \mathbf{k}_i / \sqrt{d}$ be the similarity score between $\mathbf{q}$ and $\mathbf{k}_i$, and $\mathbf{a} = (a_1, \ldots, a_N)$ with $a_i = w_i / \sum_{j=1}^N w_j$ be the corresponding attention weights where $w_i = \exp(s_i)$. We write SDPA output $\mathbf{o} \in \mathbb{R}^d$ as

$$\mathbf{o} = \frac{\sum_{i=1}^N w_i \mathbf{v}_i}{\sum_{j=1}^N w_j} = \sum_{i=1}^N a_i \mathbf{v}_i = \mathbf{V}^\mathsf{T} \mathbf{a}. \tag{1}$$

Sparse Attention with deterministic top-k Selection: Let heavy set $\mathcal{H} \subseteq [N]$ denote a deterministic top-k subset from the N cached tokens. Sparse attention over the set $\mathcal{H}$ is defined as $\mathrm{o}_\mathcal{H} = \frac{\sum_{i \in \mathcal{H}} w_i \mathbf{v}_i}{\sum_{j \in \mathcal{H}} w_j}$. While deterministic top-k selection concentrates computation on tokens deemed most relevant, tokens outside $\mathcal{H}$ do not contribute to the sparse output.

Sparse Attention with Importance Sampling: Let sampled set $\mathcal{S} \subseteq [N]$ be a non-empty random set of selected tokens. For each token i, let $Z_i = \mathbf{1}\{i \in \mathcal{S}\}$ indicate whether it is selected, and let $\pi_i = \Pr(i \in \mathcal{S}) > 0$ be its inclusion probability. Importance-based sparse attention estimates the dense output by weighting each selected token by the inverse of its inclusion probability:

$$\widehat{\mathbf{o}} = \frac{\sum_{i=1}^N (Z_i/\pi_i) w_i \mathbf{v}_i}{\sum_{i=1}^N (Z_i/\pi_i) w_i} = \frac{\sum_{i \in \mathcal{S}} (w_i/\pi_i) \mathbf{v}_i}{\sum_{i \in \mathcal{S}} w_i/\pi_i}. \tag{2}$$

The factors $1/\pi_i$ account for differences in how likely tokens are to be selected. The estimator in eq. 2 is an instance of inverse-probability weighting. We next recall the standard draw-based formulation of importance sampling.

3.2 IMPORTANCE SAMPLING

Importance sampling (Tokdar & Kass, 2010) is a Monte Carlo technique for estimating a quantity using samples drawn from a proposal distribution. Given a target distribution $p(x)$ and a function $f(x)$, consider the expectation $\mu = \mathbb{E}_{x \sim p}[f(x)] = \sum_x p(x) f(x)$. Instead of sampling directly from $p(x)$, importance sampling draws samples $x_1, \ldots, x_m \sim q(x)$ from a proposal distribution $q(x)$ and rewrites the expectation as $\mu = \mathbb{E}_{x \sim q}\left[\frac{p(x)}{q(x)} f(x)\right]$. This yields the Monte Carlo estimator

$$\hat{\mu} = \frac{1}{m} \sum_{j=1}^m \frac{p(x_j)}{q(x_j)} f(x_j).$$

The ratio $p(x)/q(x)$ is the *importance weight*, which corrects for sampling from q rather than p. When q places greater probability on samples with large contributions to the expectation, importance sampling can achieve lower variance than uniform sampling with the same sample budget.

4 INTRODUCING STRATIFIED SPARSE ATTENTION (SSA)

As already mentioned, deterministic top-k attention (Ge et al., 2024; Zhang et al., 2023) can omit meaningful attention mass outside the selected set. Uniform sampling on residual tokens provides broader coverage, but does not use information about which remaining tokens are more likely to matter. We introduce **Stratified Sparse Attention (SSA)**, a unified framework that retains tokens with high proxy scores and samples the residual tokens according to those scores. SSA divides a fixed attention budget between two well-known complementary objectives: *exploitation*, by deterministically retaining high-confidence heavy hitters, and *exploration*, by sampling the residual context according to proxy attention scores. This score-aware stratification preserves heavy hitters while providing meaningful stochastic coverage toward more promising residual tokens. We next formalize the SSA mechanism described Algorithm 1.

Algorithm 1 Stratified Sparse Attention (SSA)

1: **Input:** Proxy scores $\{\tilde{s}_i\}_{i=1}^N$, query $\mathbf{q}$, keys $\{\mathbf{k}_i\}_{i=1}^N$, values $\{\mathbf{v}_i\}_{i=1}^N$, total budget k
2: $q_i \leftarrow \frac{\exp(\tilde{s}_i)}{\sum_{j=1}^N \exp(\tilde{s}_j)}$,
3: Find $\lambda > 0$ s.t. $\sum_{i=1}^N \min(1, \lambda q_i) = k$
4: Define heavy set $\mathcal{H} = \{i : \lambda q_i \geq 1\}$, residual set $\mathcal{R} = [N] \setminus \mathcal{H}$, heavy size $h = |\mathcal{H}|$, sampling budget $m = k - h$
5: Normalize the proposal over $\mathcal{R}$: $p_i = \frac{q_i}{\sum_{j \in \mathcal{R}} q_j}$ for $i \in \mathcal{R}$
6: Draw with replacement: $J_1, J_2, \ldots J_m \sim \mathbf{p}$
7: $\mathcal{S} \leftarrow \text{unique}\{J_1, J_2, \ldots, J_m\}$
8: $\pi_i \leftarrow 1 - (1 - p_i)^m$ for $i \in \mathcal{S}$
9: $\mathcal{K} \leftarrow \mathcal{H} \cup \mathcal{S}$
10: $\pi_i \leftarrow 1$ for $i \in \mathcal{H}$
11: **for** $i \in \mathcal{K}$ **do**
12: $\quad \ell_i \leftarrow \mathbf{q}^\top \mathbf{k}_i / \sqrt{d}$
13: $\quad \hat{w}_i \leftarrow \exp(\ell_i)/\pi_i$
14: **end for**
15: **return** $\hat{\mathbf{o}} \leftarrow \frac{\sum_{i \in \mathcal{K}} \hat{w}_i \mathbf{v}_i}{\sum_{i \in \mathcal{K}} \hat{w}_i}$

Table 1: Average RULER-HARD performance at 50× attention sparsity across models and context lengths. Full per-task results are provided in the Appendix.

Model	Method	32K	64K	128K
Llama-3.1-8B	Dense	88.76	86.52	71.26
	PQ	72.51	67.53	39.73
	vAttn (PQ)	66.71	66.22	39.45
	Oracle-top-k	78.80	71.91	**56.10**
	SSA	**83.67**	**77.09**	47.53
Ministral-3-3B	Dense	86.69	82.19	81.49
	PQ	70.81	69.88	66.48
	vAttn (PQ)	69.26	65.77	63.06
	Oracle-top-k	72.81	75.09	**71.87**
	SSA	**82.05**	**75.70**	69.65
Ministral-3-8B	Dense	89.33	86.13	81.45
	PQ	81.83	78.02	62.99
	vAttn (PQ)	78.69	75.19	63.78
	Oracle-top-k	**86.61**	**83.41**	**75.83**
	SSA	84.17	79.91	69.25
Qwen3.5-4B	Dense	91.95	91.11	89.45
	PQ	91.22	90.72	88.72
	vAttn (PQ)	90.56	90.61	89.28
	Oracle-top-k	91.39	90.39	89.39
	SSA	**91.45**	**90.83**	**89.50**
Qwen3.5-9B	Dense	93.45	92.22	91.28
	PQ	92.00	91.08	90.95
	vAttn (PQ)	91.67	91.30	90.61
	Oracle-top-k	92.56	92.28	91.39
	SSA	**93.33**	**92.36**	**91.45**

4.1 Final Algorithm

SSA operates in three stages that divide the sparse-attention budget k between deterministic exploitation and stochastic exploration.

(i) **Proxy-based budget allocation (Steps 1-4):** SSA converts the proxy scores $\tilde{s}_i$ into probabilities q_i and finds λ such that $\sum_{i=1}^N \min\{1, \lambda q_i\} = k$. Tokens satisfying $\lambda q_i \geq 1$ form the deterministically retained heavy set $\mathcal{H}$. The remaining $m = k - |\mathcal{H}|$ is allocated to importance sampling from the residual set $\mathcal{R} = [N] \setminus \mathcal{H}$. Thus, the proxy distribution determines the split between deterministic allocation and residual sampling automatically; SSA does not require a separate choice of heavy-set size.

(ii) **Score-guided residual sampling (Steps 5-10):** SSA normalizes the proxy probabilities over $\mathcal{R}$ to obtain proposal probabilities $p_i = q_i / \sum_{j \in \mathcal{R}} q_j$, makes m multinomial draws with replacement, and retains the unique keys in $\mathcal{S}$. Let $\mathcal{K} = \mathcal{H} \cup \mathcal{S}$, and define $Z_i = \mathbf{1}\{i \in \mathcal{K}\}$ to indicate whether token i is selected. Its inclusion probability is $\pi_i = \Pr(Z_i = 1)$. For $i \in \mathcal{H}$, $\pi_i = 1$. For $i \in \mathcal{R}$, the token is selected if it appears in at least one of the m draws, giving $\pi_i = 1 - (1 - p_i)^m$.

(iii) **Exact scoring and correction (Steps 11-15):** For each token in $\mathcal{K}$, SSA computes its exact score $\ell_i = \mathbf{q}^\top \mathbf{k}_i / \sqrt{d}$ and corrected weight $\hat{w}_i = \exp(\ell_i)/\pi_i$. It returns the normalized weighted sum $\hat{\mathbf{o}}$. Since repeated draws are retained only once, $|\mathcal{K}| \leq k$.

4.2 Theoretical Insights

We now characterize how accurately SSA approximates dense attention. Our analysis of Algorithm 1 is designed for the budget parameter $k \in \{1, \ldots, N-1\}$. When $k = N$, every token is retained (see step 3), so SSA trivially returns dense attention exactly. The residual set is then empty and requires no sampling analysis. For $k < N$, the residual set is nonempty *i.e.*, $m \geq 1$. Let a_i be the dense attention weights from Section 3, and $\|\mathbf{V}\|_2$ denote the spectral norm of the value matrix. In

addition, we introduce the following quantities for our analysis,

$$A_{\mathcal{R}} = \sum_{i \in \mathcal{R}} a_i, \qquad \bar{a}_i = \frac{a_i}{A_{\mathcal{R}}}, \qquad \rho = \max_{i \in \mathcal{R}} \frac{\bar{a}_i}{p_i},$$

where $A_{\mathcal{R}}$ is the true attention mass left for sampling, while ρ measures how well the residual proposal covers that mass. For finite proxy scores, $q_i > 0$ and hence $p_i = q_i / \sum_{j \in \mathcal{R}} q_j > 0$ for every $i \in \mathcal{R}$. Since $\sum_{i \in \mathcal{R}} p_i(\bar{a}_i/p_i) = \sum_{i \in \mathcal{R}} \bar{a}_i = 1$, the largest ratio must satisfy $\rho \geq 1$. The equality holds precisely when every ratio is equal to one, *i.e.*, $p_i = \bar{a}_i$ for all $i \in \mathcal{R}$.

The budget-allocation rule also limits the effect of duplicate residual draws. Indeed, steps 3–4 of Algorithm 1 give $k = h + \lambda \sum_{i \in \mathcal{R}} q_i$ as each heavy token contributes 1 to the budget equation, and each residual token contributes $\lambda q_i < 1$. Since $m = k - h$, it follows that $m = \lambda \sum_{i \in \mathcal{R}} q_i$. Combining this with $p_i = q_i / \sum_{j \in \mathcal{R}} q_j$ gives $mp_i = \lambda q_i < 1$ for every $i \in \mathcal{R}$. Here mp_i is the expected number of times token i appears among the m draws. The largest expected multiplicity among the residual tokens is therefore defined as

$$c := \max_{i \in \mathcal{R}} mp_i < 1.$$

A small value of c corresponds to a diffuse proposal with few repetitions, whereas a value closer to one indicates a more concentrated proposal and a greater possibility of repeated draws.

Now, to use the proxy-quality parameter ρ in the error bound, we need to relate the proposal probability p_i to the inclusion probability π_i used by Algorithm 1. A residual token is retained if it appears in at least one of the m draws, so $\pi_i = 1 - (1 - p_i)^m$. To account for duplicate draws, we define

$$\kappa(c) := \frac{1 - e^{-c}}{c}.$$

The following lemma quantifies their effect on the inclusion probabilities.

Lemma 1 *Let $m \geq 1$, let $p_i > 0$ for $i \in \mathcal{R}$, and let c and $\kappa(c)$ be defined above. For every $i \in \mathcal{R}$, $\kappa(c) mp_i \leq \pi_i \leq mp_i$. Consequently, $\frac{a_i}{\pi_i} \leq \frac{A_{\mathcal{R}} \rho}{\kappa(c) m}$.*

Now, we are ready to use the lemma to bound SSA's approximation error.

Theorem 2 *For any $\delta \in (0, 1)$, let $\epsilon_D(\delta) := \frac{2\sqrt{2}\, A_{\mathcal{R}} \rho}{\kappa(c)} \sqrt{\frac{\log(4/\delta)}{m}}$. If $\epsilon_D(\delta) < 1$, then, with probability at least $1 - \delta$ over the residual sampling,*

$$\|\widehat{\mathbf{o}} - \mathbf{o}\|_2 \leq \frac{4\sqrt{2}\, A_{\mathcal{R}} \rho}{\kappa(c)\big(1 - \epsilon_D(\delta)\big)} \sqrt{\frac{\log(4/\delta)}{m}}\, \|\mathbf{V}\|_2. \tag{3}$$

Since $\kappa(c) > 1 - e^{-1}$, deduplication contributes only a universal constant factor to the approximation and sample-complexity bounds. The following corollary gives a direct (ε, δ) guarantee and a sufficient sample size $m = \mathcal{O}\left(A_{\mathcal{R}}^2 \rho^2 \varepsilon^{-2} \log \frac{1}{\delta}\right)$ that automatically ensures $\epsilon_D(\delta) < 1$.

Corollary 3 *Consider Algorithm 1 with an integer budget $1 \leq k < N$ and finite attention and proxy scores. Let $\varepsilon, \delta \in (0, 1)$. If $m \geq 21 A_{\mathcal{R}}^2 \rho^2 \frac{(2+\varepsilon)^2}{\varepsilon^2} \log \frac{4}{\delta}$, then, with probability at least $1 - \delta$,*

$$\|\widehat{\mathbf{o}} - \mathbf{o}\|_2 \leq \varepsilon \|\mathbf{V}\|_2. \tag{4}$$

Remark 4 *Note that the approximation error in Theorem 2 scales as $A_{\mathcal{R}} \rho / \sqrt{m}$. The residual mass $A_{\mathcal{R}}$ decreases when the retained heavy tokens capture more of the true attention mass, while ρ increases when the proxy underestimates an important residual token. Increasing the residual budget m reduces the stochastic error at the standard Monte Carlo rate $m^{-1/2}$. For a fixed total budget, SSA therefore balances the reduction in $A_{\mathcal{R}}$ obtained from retaining more heavy tokens against the larger sampling budget obtained from retaining fewer of them.*

Remark 5 *In both Theorem 2 and Corollary 3, we express the bounds using $\|\mathbf{V}\|_2$. The proofs only require a bound on the individual value-vector norms, so the same results hold with $\max_{i \in [N]} \|\mathbf{v}_i\|_2$ in its place, without changing the constants.*

4.3 KERNEL DESIGN

At long contexts, decode attention is memory-bandwidth bound because each step reads the entire KV cache from HBM (Shah et al., 2024; Kwon et al., 2023). SSA reduces this memory traffic by selecting a sparse subset of tokens before loading their KV states. While Algorithm 1 determines the division between the heavy and residual strata adaptively, our optimized GPU implementation uses a fixed heavy fraction. Developing an equally efficient implementation of the adaptive algorithm remains future work. As shown in Table 13, the practical fixed-budget variant closely matches the performance of the theoretical formulation. The residual proposal, inclusion-probability correction, and resulting attention estimator remain unchanged. Our SSA decode step consists of four stages, where only the first three scan the full context using a few bytes per token.

Proxy scoring: Each key is represented by a 16-byte PQ code (16 subspaces, 256 centroids) (Zhang et al., 2025), compared to 1 KiB for its fp16 key-value pair at head dimension 256. For each query, a lookup table computes the proxy score $\tilde{s}_i$ for every token. Scores are scaled and masked for sink and local tokens within the same kernel, avoiding an additional pass.

Heavy-set selection: We select $\mathcal{H}$ using radix top-k (Shanbhag et al., 2018), which returns an unsorted index set since the estimator in eq. 2 is invariant to the ordering of selected tokens.

Residual sampling: Direct sampling from the full residual distribution becomes expensive at long context lengths, as constructing and sampling from the distribution requires processing probabilities across the entire residual KV cache. To reduce this overhead, we use a hierarchical sampler that progressively narrows the search space. Each draw first selects a 1024-token chunk, then a 32-token sub-tile, and finally scans at most 32 tokens to select a key. We remove duplicate samples with a per-row hash table and reweight each unique sample by $1/\pi_i$. In practice, we clamp π_i at 10^{-4} to avoid excessively large weights, at the cost of introducing a small bias. Finally, we use a counter-based random number generator to obtain fresh samples across replays while retaining compatibility with CUDA graph capture.

Sparse decode: The final stage gathers the keys and values in $\mathcal{H} \cup \mathcal{S}$ from the paged KV cache and evaluates eq. 2 using an online softmax. To exploit more parallelism, we divide each query head across multiple thread blocks and combine their partial softmax results using a numerically stable log-sum-exp reduction. At 1M context and batch size 1, using 64 splits is 1.58-1.71$\times$ faster than using 32 splits across sparsity levels. Increasing the number of splits to 128 or 256, however, reduces performance because the additional parallelism no longer offsets the increased launch and merge overhead.

5 EXPERIMENTS

We conduct an extensive evaluation of SSA against Oracle-top-k, deterministic top-k, pure sampling, and hybrid baselines across multiple datasets, models, context lengths, and generation lengths. In addition, note that we slightly overload the notation h. In Algorithm 1 and the theoretical analysis, $h = |\mathcal{H}|$ denotes the number of heavy tokens, whereas in the implementation and experimental discussion, $h \in [0, 1]$ denotes the fraction of the total budget assigned to the heavy set. We describe the evaluation setup below.

5.1 SETUP

Baselines: We compare SSA against three representative sparse-attention baselines: vAttention (Desai et al., 2026), PQCache (Zhang et al., 2025), and Oracle-top-k. Oracle-top-k performs exact attention-score computation before selecting the top-k tokens and is therefore impractical for acceleration; we include it as an upper-bound reference for deterministic top-k selection. For all methods, we use 128 sink tokens and 128 local tokens, along with the hyperparameters recommended by the respective authors. Additional comparisons with MagicPig (Chen et al., 2025), HashAttention (Desai et al., 2025), Double Sparsity (Yang et al., 2024), and Quest (Tang et al., 2024) are provided in Table 4 in the Appendix. These results motivate our use of PQCache, the strongest approximate top-k baseline in this comparison, to construct SSA's proposal distribution throughout our experiments.

Sparsity	Qwen3.5-9B Oracle-top-k	PQ	vAttn(PQ)	SSA	Qwen3.5-4B Oracle-top-k	PQ	vAttn(PQ)	SSA
Dense	66.3				54.2			
5×	63.8	63.5	**65.4**	61.0	49.8	49.4	**54.4**	52.7
10×	59.4	53.8	62.7	**64.0**	47.9	41.5	**51.9**	51.0
20×	55.4	49.4	58.3	**63.5**	42.7	33.8	50.2	**53.1**
50×	48.3	36.7	35.8	**59.8**	36.5	25.4	27.5	**50.0**
100×	42.5	28.1	3.5	**51.5**	31.9	13.1	1.7	**42.5**

Method	50×	20×
Dense	41.56	41.56
PQ	40.60	40.98
vAttn (PQ)	40.39	41.47
SSA	40.92	**41.56**
Oracle-top-k	**41.25**	41.40

Table 2: AIME 2025 avg@16 accuracy (left). LongBench performance with Qwen3.5-27B (right).

Datasets and Models: We evaluate SSA on four benchmark suites: RULER (32K-128K) (Hsieh et al., 2024), LongBench (10K-16K) (Bai et al., 2024), LOFT (32K-128K) (Lee et al., 2024), and AIME (32K token generation) (Mathematical Association of America, 2025). Our evaluation spans five models: Llama-3.1-8B-Instruct (Grattafiori et al., 2024), Qwen3.5-4B and Qwen3.5-9B (Team, 2026), and Ministral-3-3B-Instruct-2517 and Ministral-3-8B-Instruct-2517 (Liu et al., 2026). Dataset subsets and additional evaluation details are provided in Table 5 in Appendix.

5.2 Effectiveness of SSA

RULER-HARD: Table 1 reports average RULER-HARD scores at 50× attention sparsity. SSA consistently outperforms both PQ and vAttention(PQ) across all models and context lengths. Since these methods use the same PQ scores and attention budget, the comparison highlights the benefit of allocating part of the budget to score-guided sampling rather than exclusively to top-k retrieval or uniform sampling. The gains are particularly pronounced when deterministic top-k struggles. At 32K, SSA improves Llama-3.1-8B from 72.51 (PQ) to 83.67, approaching the dense score of 88.76. Similarly, on Ministral-3-3B, SSA improves the score from 70.81 to 82.05, compared with 86.69 for dense attention. Oracle-top-k, which selects tokens using exact attention scores, remains stronger at 128K for Llama-3.1-8B and Ministral-3-3B, and across all context lengths for Ministral-3-8B. On Qwen3.5-4B and Qwen3.5-9B, all sparse methods remain within two points of dense attention.

AIME 2025: Table 2 reports avg@16 accuracy for long-form generation. SSA remains robust as sparsity increases, with particularly large gains in the aggressive-sparsity regime. On Qwen3.5-9B at 50× sparsity, SSA achieves 59.8, compared with 48.3 for oracle top-k, 36.7 for PQ, and 35.8 for vAttention(PQ). At 100× sparsity, SSA retains an accuracy of 51.5, while vAttention(PQ) drops to 3.5. At lower sparsity, vAttention(PQ) performs better at 5× and at 10× on Qwen3.5-4B. We further study the effect of the heavy fraction h in Table 12. Hybrid allocations with $h \in [0.2, 0.8]$ achieve 61.0-64.0, substantially outperforming both pure sampling ($h = 0$), which achieves 40.2-42.5, and pure top-k ($h = 1$), which achieves 49.4-53.8. These results demonstrate the benefit of combining deterministic retrieval with score-guided importance sampling, particularly under aggressive sparsity.

LOFT and LongBench: On LOFT with Qwen3.5-9B and Qwen3.5-4B (see fig. 2), SSA has the lowest attention error among the non-oracle methods at every context length and sparsity level, while subspan-EM varies by at most 0.04 across methods (see fig. 4). On LongBench with Qwen3.5-27B (Table 2), all sparse methods stay within one point of dense attention at both 50× and 20× sparsity.

5.3 Decode Efficiency

Sparse attention accelerates decoding only when its selection overhead is smaller than the memory traffic it saves. We evaluate this trade-off for SSA across context lengths and batch sizes, and separately measure the sampling overhead absent in top-k methods.

Efficiency analysis: SSA becomes increasingly advantageous as context length and batch size grow. At batch size 1, SSA is slower than FlashInfer at 128K, achieving approximately 0.60-0.74× its speed. As context length increases, this overhead is increasingly amortized, and SSA reaches 1.79×, 1.39×, and 1.07× speedups at 1M for 50×, 20×, and 10× sparsity, respectively (see fig. 3a). The gains further increase with batch size, reaching 2.96× at batch 16 for 50× sparsity (see fig. 3b); our kernel optimizations are particularly important at batch 1, improving SSA by 2.95-3.80× over the reference implementation (Table 3). Finally, although residual sampling introduces overhead absent from top-k methods, our two-level sampler reduces this cost by 1.21-3.16× over chunk-only routing

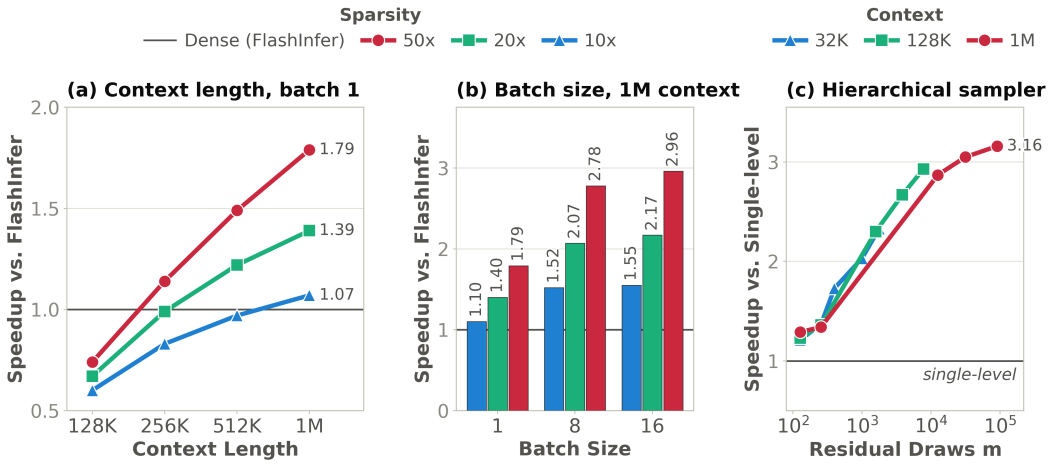


Figure 3: **SSA accelerates decode at long contexts, with larger gains at higher batch sizes.** Decode attention speedup on a single H200 (16 query heads, 4 KV heads, head dimension 256, fp16 KV). **(a)** At batch size 1, SSA overtakes FlashInfer beyond 256K context and reaches 1.79× speedup at 1M and 50× sparsity. **(b)** At 1M context, speedup increases with batch size, reaching 2.96× at batch 16 and 50× sparsity. **(c)** Overhead of two-level residual sampling relative to 1,024-token chunk-only sampling as the number of draws m increases.

Table 3: **Kernel optimizations make SSA faster than dense decoding even at batch size 1.** We report decode-time attention-layer latency at 1M context length on a single H200 GPU. *Reference* implements the same algorithm without the kernel optimizations described in Section 4.3. At batch size 1, it reaches only 0.37-0.47× the speed of FlashInfer and surpasses FlashInfer only at batch size 8. In contrast, optimized SSA is faster than FlashInfer across all measured batch sizes and sparsity levels. FlashInfer latencies across the two runs agree within 1.2%.

		Latency (ms)			SSA speedup	
Sparsity	**Batch**	**FlashInfer**	**Reference**	**SSA**	**vs. FlashInfer**	**vs. reference**
50×	1	1.41	2.99	0.79	1.79×	3.80×
	8	11.90	7.18	4.27	2.78×	1.68×
	16	24.47	12.57	8.26	2.96×	1.52×
20×	1	1.41	3.34	1.01	1.40×	3.32×
	8	11.91	8.74	5.76	2.07×	1.52×
	16	24.53	15.51	11.32	2.17×	1.37×
10×	1	1.42	3.83	1.30	1.10×	2.95×
	8	12.01	11.06	7.91	1.52×	1.40×
	16	24.55	19.82	15.81	1.55×	1.25×

as the number of draws m increases (see fig. 3c), keeping sampling latency at 0.30-1.18 ms at 1M context and $h = 0.4$ across the evaluated sparsity levels.

6 Conclusion

We introduce Stratified Sparse Attention (SSA), a unified framework encompassing top-k, pure sampling, and hybrid sparse attention while substantially outperforming existing methods and reducing the accuracy gap to dense attention. SSA is compatible with any approximate top-k method that provides proxy scores, allowing it to directly benefit from advances in approximate retrieval. An important direction for future work is to make SSA verifiable by incorporating user-controlled guarantees on approximation quality.

AI USE STATEMENT

In this work, we used generative AI tools to assist with polishing the writing and exploring design considerations for SSA's kernel implementations. Generative AI tools were not used to develop the core idea of SSA, formulate its methodology, conduct experiments, analyze results, or draw scientific conclusions. We carefully reviewed all AI-assisted content and take full responsibility for the final content of this work, including all text, claims, results, and artifacts produced.

ETHICS STATEMENT

Our work focuses on accelerating model inference through sparse attention. As such, it has the potential for broader impact by enabling practitioners to perform long-context inference more efficiently and at lower computational cost. Our experiments rely exclusively on publicly available datasets to evaluate the performance of our method, and we do not identify any additional ethical considerations arising specifically from this work.

REPRODUCIBILITY STATEMENT

We provide the source code and configurations for our key experiments on RULER, LongBench, and LOFT along with instructions for reproducing our results, in Section 5. All theoretical results and proofs are presented in the appendix, together with detailed explanations and the underlying assumptions. We have thoroughly validated our implementation and empirically verified that the proposed algorithm behaves as expected.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023.

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*, August 2024.

Weili Cao, Xunjian Yin, Bhuwan Dhingra, and Shuyan Zhou. Coding Agents are Effective Long-Context Processors. *arXiv preprint arXiv:2603.20432*, 2026.

Mingguang Chen, Licheng Wang, and Bo Qu. The Horizon Gap: Planning, Memory, Execution, Training, and Evaluation for Long-Horizon LLM Agents. *arXiv preprint arXiv:2608.06663*, 2026.

Xiangping Chen, Xing Hu, Yuan Huang, He Jiang, Weixing Ji, Yanjie Jiang, Yanyan Jiang, Bo Liu, Hui Liu, Xiaochen Li, Xiaoli Lian, Guozhu Meng, Xin Peng, Hailong Sun, Lin Shi, Bo Wang, Chong Wang, Jiayi Wang, Tiantian Wang, Jifeng Xuan, Xin Xia, Yibiao Yang, Yixin Yang, Li Zhang, Yuming Zhou, and Lu Zhang. Deep learning-based software engineering: Progress, challenges, and opportunities. *CoRR*, abs/2410.13110, 2024.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG: LSH Sampling for Efficient LLM Generation. In *International Conference on Learning Representations*, 2025.

DeepSeek-AI. DeepSeek-V2: A Strong, Economical, and Efficient Mixture-of-Experts Language Model. *arXiv preprint arXiv:2405.04434*, 2024.

Aditya Desai, Shuo Yang, Alejandro Cuadron, Ana Klimovic, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. HashAttention: Semantic Sparsity for Faster Inference. In *International Conference on Machine Learning*, 2025.

Aditya Desai, Kumar Krishna Agrawal, Shuo Yang, Alejandro Cuadron, Luis Gaspar Schroeder, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. vAttention: Verified Sparse Attention via Sampling. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=zzTDulLys0.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model Tells You What to Discard: Adaptive KV Cache Compression for LLMs. In *International Conference on Learning Representations*, 2024.

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. *arXiv preprint arXiv:2407.21783*, 2024.

Themistoklis Haris. kNN Attention Demystified: A Theoretical Exploration for Scalable Transformers. In *The Thirteenth International Conference on Learning Representations*, 2025.

Coleman Richard Charles Hooper, Sehoon Kim, Hiva Mohammadzadeh, Monishwaran Maheswaran, Sebastian Zhao, June Paik, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. Squeezed Attention: Accelerating Long Context Length LLM Inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, 2025.

Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, and Haoyu Wang. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Trans. Softw. Eng. Methodol.*, 33(8), 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? In *First Conference on Language Modeling (COLM)*, 2024.

Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613, 1998.

Sahil Joshi, Agniva Chowdhury, Wyatt Bellinger, Amar Kanakamedala, Ekam Singh, Hoang Anh Duy Le, Aditya Desai, and Anshumali Shrivastava. SOCKET: SOft Collision Kernel EsTimator for Sparse Attention. *arXiv preprint arXiv:2602.06283*, 2026.

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. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

Hoang Anh Duy Le, Sahil Joshi, Zeyu Yang, Zhaozhuo Xu, and Anshumali Shrivastava. Scout before you attend: Sketch-and-walk sparse attention for efficient LLM inference. In *Forty-third International Conference on Machine Learning*, 2026.

Jinhyuk Lee, Anthony Chen, Zhuyun Dai, Dheeru Dua, Devendra Singh Sachan, Michael Boratko, Yi Luan, Sébastien M. R. Arnold, Vincent Perot, Siddharth Dalmia, Hexiang Hu, Xudong Lin, Panupong Pasupat, Aida Amini, Jeremy R. Cole, Sebastian Riedel, Iftekhar Naim, Ming-Wei Chang, and Kelvin Guu. Can Long-Context Language Models Subsume Retrieval, RAG, SQL, and More? *arXiv preprint arXiv:2406.13121*, 2024.

Yinheng Li, Shaofei Wang, Han Ding, and Hang Chen. Large language models in finance: A survey. In *Proceedings of the 4th ACM International Conference on AI in Finance*, 2023. doi: 10.1145/3604237.3626869.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. SnapKV: LLM Knows What You are Looking for Before Generation. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Xihui Lin, Yunan Zhang, Suyu Ge, Liliang Ren, Barun Patra, Vishrav Chaudhary, Hao Peng, and Xia Song. S2-ATTENTION: HARDWARE-AWARE CONTEXT SHARDING AMONG ATTENTION HEADS. In *Sparsity in LLMs (SLLM): Deep Dive into Mixture of Experts, Quantization, Hardware, and Inference*, 2025. URL https://openreview.net/forum?id=AhlE339I7o.

Alexander H. Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, Alexandre Sablayrolles, Amélie Héliou, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. RetrievalAttention: Accelerating Long-Context LLM Inference via Vector Retrieval. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Mathematical Association of America. American Invitational Mathematics Examination (AIME) 2025. https://maa.org/maa-invitational-competitions/, 2025. Accessed: 2026-05-20.

Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Fast attention over long sequences with dynamic sparse flash attention. In *Advances in Neural Information Processing Systems*, volume 36, pp. –, 2023.

Junyoung Park, Dalton Jones, Matthew J Morse, Raghavv Goel, Mingu Lee, and Christopher Lott. KeyDiff: Key Similarity-Based KV Cache Eviction for Long-Context LLM Inference in Resource-Constrained Environments. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

William Peebles and Saining Xie. Scalable Diffusion Models with Transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Iosif Pinelis. Optimum bounds for the distributions of martingales in banach spaces. *The Annals of Probability*, 22(4):1679–1706, 1994. doi: 10.1214/aop/1176988477.

Ludan Ruan and Qin Jin. Survey: Transformer based Video-Language Pre-training. *AI Open*, 3:1–13, 2022. https://www.sciencedirect.com/science/article/pii/S2666651022000018.

Ayan Sengupta, Siddhant Chaudhary, and Tanmoy Chakraborty. Value-Guided KV Compression for LLMs via Approximated CUR Decomposition. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. FlashAttention-3: Fast and Accurate Attention with Asynchrony and Low-precision. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

Anil Shanbhag, Holger Pirk, and Samuel Madden. Efficient top-k query processing on massively parallel hardware. In *Proceedings of the 2018 International Conference on Management of Data*, SIGMOD '18. Association for Computing Machinery, 2018. ISBN 9781450347037. URL https://doi.org/10.1145/3183713.3183735.

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

Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank Keys for Efficient Sparse Attention. In *Advances in Neural Information Processing Systems*, 2024.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: Query-Aware Sparsity for Efficient Long-Context LLM Inference. In *International Conference on Machine Learning*, 2024.

Qwen Team. Qwen3.5: Accelerating productivity with native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Surya T. Tokdar and Robert E. Kass. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2, 2010.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software (TOMS)*, 11(1), 1985.

Siyuan Yan, Guo-Qing Jiang, Yuchen Zhang, Xiaoxing Ma, Ran Zhu, Chun Cao, and Jingwei Xu. Adamas: Hadamard Sparse Attention for Efficient Long-Context Inference. *arXiv preprint arXiv:2510.18413*, 2025.

Jiaming Yang, Chenwei Tang, Liangli Zhen, and Jiancheng Lv. Rethinking KV cache eviction via a unified information-theoretic objective. In *Forty-third International Conference on Machine Learning*, 2026.

Shuo Yang, Ying Sheng, Joseph E. Gonzalez, Ion Stoica, and Lianmin Zheng. Post-Training Sparse Attention with Double Sparsity. *arXiv preprint arXiv:2408.07092*, 2024.

Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. PQCache: Product Quantization-based KVCache for Long Context LLM Inference. *Proceedings of the ACM on Management of Data*, 3(3):1–30, 2025.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. In *Advances in Neural Information Processing Systems*, 2023.

Xiabin Zhou, Wenbin Wang, Minyan Zeng, Jiaxian Guo, Xuebo Liu, Li Shen, Min Zhang, and Liang Ding. DynamicKV: Task-Aware Adaptive KV Cache Compression for Long Context LLMs. *arXiv preprint arXiv:2412.14838*, 2024a.

Xuanhe Zhou, Zhaoyan Sun, and Guoliang Li. DB-GPT: Large Language Model Meets Database. *Data Science and Engineering*, 9:102 – 111, 2024b.

Kan Zhu, Tian Tang, Qinyu Xu, Zhan Jin, Yile Gu, Zhichen Zeng, Rohan Kadekodi, Liangyu Zhao, Ang Li, Arvind Krishnamurthy, and Baris Kasikci. Tactic: Adaptive Sparse Attention with Clustering and Distribution Fitting for Long-Context LLMs. In *The Fourteenth International Conference on Learning Representations*, 2026.

A Appendix

A.1 Proof of Lemma 1

Proof Token i is included if it appears in at least one of the m independent draws, so

$$\pi_i = 1 - (1 - p_i)^m.$$

By the union bound,

$$\pi_i = \Pr\left(\bigcup_{\ell=1}^{m} \{J_\ell = i\}\right) \le \sum_{\ell=1}^{m} \Pr(J_\ell = i) = mp_i.$$

For the lower bound, $1 - p_i \le e^{-p_i}$ gives

$$\pi_i = 1 - (1 - p_i)^m \ge 1 - e^{-mp_i}.$$

The function $g(x) = (1 - e^{-x})/x$ is decreasing for $x > 0$. Since $mp_i \le c$,

$$1 - e^{-mp_i} = mp_i\, g(mp_i) \ge mp_i\, g(c) = \kappa(c) mp_i.$$

This proves the two-sided bound on π_i.

Finally, the definition of ρ gives $a_i = A_{\mathcal{R}} \bar{a}_i \le A_{\mathcal{R}} \rho p_i$. Combining this with the lower bound on π_i yields

$$\frac{a_i}{\pi_i} \le \frac{A_{\mathcal{R}} \rho p_i}{\kappa(c) mp_i} = \frac{A_{\mathcal{R}} \rho}{\kappa(c) m}.$$

■

A.2 Proof of Theorem 2

We first establish the following dimension-free bounded-differences inequality used in the proof.

Lemma 6 (Vector-valued bounded differences) *Let $X_1, \dots, X_m$ be independent random variables, and let f be a measurable function taking values in $\mathbb{R}^r$. Suppose that changing only the ℓth argument changes f by at most b_ℓ in Euclidean norm* i.e.*,*

$$\|f(x_1, \dots, x_\ell, \dots, x_m) - f(x_1, \dots, x'_\ell, \dots, x_m)\|_2 \le b_\ell$$

for every $\ell \in [m]$ and every pair of input sequences differing only in coordinate ℓ, where $b_\ell \ge 0$ and $\sum_{\ell=1}^m b_\ell^2 > 0$. Then, for every $t > 0$,

$$\Pr\left(\|f(X_1, \dots, X_m) - \mathbb{E}[f(X_1, \dots, X_m)]\|_2 \ge t\right) \le 2\exp\left(-\frac{t^2}{2\sum_{\ell=1}^m b_\ell^2}\right). \tag{5}$$

Proof Reveal the variables one at a time and define

$$M_\ell = \mathbb{E}[f(X_1, \dots, X_m) \mid X_1, \dots, X_\ell], \qquad \ell = 0, \dots, m.$$

Here

$$M_0 = \mathbb{E}[f(X_1, \dots, X_m)], \qquad M_m = f(X_1, \dots, X_m).$$

The sequence $(M_\ell)_{\ell=0}^m$ is a vector-valued martingale.

We first bound its increments. Fix the first $\ell - 1$ variables and let

$$g_\ell(x) = \mathbb{E}_{X_{\ell+1}, \dots, X_m}[f(X_1, \dots, X_{\ell-1}, x, X_{\ell+1}, \dots, X_m)].$$

Averaging over the same future variables preserves the bounded-differences condition:

$$\|g_\ell(x) - g_\ell(x')\|_2 \le b_\ell.$$

Let X'_ℓ be an independent copy of X_ℓ, independent of all the original variables. Since $M_\ell = g_\ell(X_\ell)$ and $M_{\ell-1}$ averages g_ℓ over the ℓth variable, Jensen's inequality gives

$$\begin{aligned}\|M_\ell - M_{\ell-1}\|_2 &= \|\mathbb{E}[g_\ell(X_\ell) - g_\ell(X'_\ell) \mid X_1, \dots, X_\ell]\|_2 \\ &\le \mathbb{E}[\|g_\ell(X_\ell) - g_\ell(X'_\ell)\|_2 \mid X_1, \dots, X_\ell] \\ &\le b_\ell.\end{aligned}$$

The Euclidean specialization of the martingale inequality in (Pinelis, 1994, Theorem 3.5), applied to $(M_\ell - M_0)_{\ell=0}^m$, therefore yields

$$\Pr\left(\|M_m - M_0\|_2 \ge t\right) \le 2\exp\left(-\frac{t^2}{2\sum_{\ell=1}^m b_\ell^2}\right).$$

Substituting M_m and M_0 proves the claim. ■

Now, we are ready to prove Theorem 2.

Proof [Proof of Theorem 2] We consider the sparse regime with an integer budget $1 \le k < N$ and finite attention and proxy scores. The case $k = N$ can be handled by evaluating dense attention exactly. Recall that $\mathcal{H}$ and $\mathcal{R} = [N] \setminus \mathcal{H}$ be the heavy and residual sets produced by SSA, and let $h = |\mathcal{H}|$ and $m = k - h$. As shown in Section 4.2, $\mathcal{R} \ne \emptyset$ and $m \ge 1$. Since the proxy scores are finite, $p_i > 0$ for every $i \in \mathcal{R}$.

Let $J_1, \dots, J_m$ denote the independent residual draws:

$$\Pr(J_\ell = i) = p_i, \qquad i \in \mathcal{R}, \qquad \ell \in [m].$$

For each residual token, define its occupancy indicator

$$Z_i = \mathbf{1}\{J_\ell = i \text{ for some } \ell \in [m]\}, \qquad i \in \mathcal{R}.$$

We know that its inclusion probability is given by $\pi_i = \Pr(Z_i = 1) = 1 - (1 - p_i)^m$, so $\mathbb{E}[Z_i] = \pi_i$. Heavy tokens are retained deterministically and have inclusion probability one.

Note that although the residual indicators $\{Z_i\}_{i\in\mathcal{R}}$ are dependent, they are deterministic functions of the independent draws $J_1, \ldots, J_m$. We therefore apply Lemma 6 to these draws.

Write

$$D := \sum_{i=1}^{N} w_i, \qquad a_i = \frac{w_i}{D}, \qquad \mathbf{o} = \sum_{i=1}^{N} a_i \mathbf{v}_i.$$

Define the estimated denominator and numerator by

$$\widehat{D} := \sum_{i\in\mathcal{H}} w_i + \sum_{i\in\mathcal{R}} \frac{Z_i w_i}{\pi_i},$$
$$\widehat{\mathbf{N}} := \sum_{i\in\mathcal{H}} w_i \mathbf{v}_i + \sum_{i\in\mathcal{R}} \frac{Z_i w_i}{\pi_i} \mathbf{v}_i.$$

The SSA output is therefore

$$\widehat{\mathbf{o}} = \frac{\widehat{\mathbf{N}}}{\widehat{D}}.$$

Since $m \geq 1$ and all weights are positive, $\widehat{D} > 0$ for every realization of the residual draws.

If $\|\mathbf{V}\|_2 = 0$, all value vectors vanish, and $\widehat{\mathbf{o}} = \mathbf{o} = \mathbf{0}$, so the theorem is immediate. Therefore, we assume $\|\mathbf{V}\|_2 > 0$.

By Lemma 1,

$$\frac{a_i}{\pi_i} \leq \frac{A_{\mathcal{R}}\rho}{\kappa(c)m}, \qquad i \in \mathcal{R}.$$

For brevity, define

$$\beta := \frac{A_{\mathcal{R}}\rho}{\kappa(c)m}. \tag{6}$$

Thus $a_i/\pi_i \leq \beta$ for every $i \in \mathcal{R}$.

Next, we define

$$\Delta_D := \frac{\widehat{D} - D}{D}. \tag{7}$$

The contributions from $\mathcal{H}$ are evaluated exactly and cancel in $\widehat{D} - D$. Consequently,

$$\Delta_D = \sum_{i\in\mathcal{R}} a_i \left(\frac{Z_i}{\pi_i} - 1\right). \tag{8}$$

Since $\mathbb{E}[Z_i] = \pi_i$,

$$\mathbb{E}[\Delta_D] = 0.$$

Consider two draw sequences $\mathbf{J} = (J_1, \ldots, J_m)$ and $\mathbf{J}'$ differing only in coordinate ℓ, where the draw changes from r to s. Only Z_r and Z_s can change, and each changes by at most one. Therefore,

$$\begin{aligned} |\Delta_D(\mathbf{J}) - \Delta_D(\mathbf{J}')| &\leq \frac{a_r}{\pi_r} + \frac{a_s}{\pi_s} \\ &\leq 2\beta. \end{aligned} \tag{9}$$

Applying Lemma 6 with $r = 1$ and $b_\ell = 2\beta$ gives, for every $t > 0$,

$$\Pr\left(|\Delta_D| \geq t\right) \leq 2\exp\left(-\frac{t^2}{8m\beta^2}\right).$$

Taking $t = 2\sqrt{2m}\,\beta\sqrt{u}$, we obtain, for every $u > 0$,

$$\Pr\left(|\Delta_D| \geq 2\sqrt{2m}\,\beta\sqrt{u}\right) \leq 2e^{-u}. \tag{10}$$

Substituting eq. 6 yields

$$\Pr\left(|\Delta_D| \geq \frac{2\sqrt{2}\,A_{\mathcal{R}}\rho}{\kappa(c)}\sqrt{\frac{u}{m}}\right) \leq 2e^{-u}. \tag{11}$$

Now, we define

$$\mathbf{E} := \frac{\widehat{\mathbf{N}} - \mathbf{o}\widehat{D}}{D}. \tag{12}$$

Because $\sum_{i=1}^{N} a_i = 1$ and $\mathbf{o} = \sum_{i=1}^{N} a_i \mathbf{v}_i$, we have

$$\sum_{i=1}^{N} a_i(\mathbf{v}_i - \mathbf{o}) = \mathbf{0}.$$

It follows that

$$\begin{aligned}\mathbf{E} &= \sum_{i\in\mathcal{H}} a_i(\mathbf{v}_i - \mathbf{o}) + \sum_{i\in\mathcal{R}} \frac{Z_i a_i}{\pi_i}(\mathbf{v}_i - \mathbf{o}) \\ &= \sum_{i\in\mathcal{R}} a_i \left(\frac{Z_i}{\pi_i} - 1\right)(\mathbf{v}_i - \mathbf{o}).\end{aligned} \tag{13}$$

Again using $\mathbb{E}[Z_i] = \pi_i$, we obtain

$$\mathbb{E}[\mathbf{E}] = \mathbf{0}.$$

Let $\mathbf{e}_i$ denote the ith standard basis vector of $\mathbb{R}^N$. Since the value vectors are the rows of $\mathbf{V}$, for every $i \in [N]$,

$$\|\mathbf{v}_i\|_2 = \|\mathbf{V}^\top \mathbf{e}_i\|_2 \le \|\mathbf{V}^\top\|_2 \|\mathbf{e}_i\|_2 = \|\mathbf{V}\|_2.$$

Moreover, $\mathbf{o}$ is a convex combination of the value vectors, so

$$\|\mathbf{o}\|_2 \le \sum_{i=1}^{N} a_i \|\mathbf{v}_i\|_2 \le \|\mathbf{V}\|_2.$$

Therefore,

$$\|\mathbf{v}_i - \mathbf{o}\|_2 \le \|\mathbf{v}_i\|_2 + \|\mathbf{o}\|_2 \le 2\|\mathbf{V}\|_2. \tag{14}$$

Now consider draw sequences $\mathbf{J}$ and $\mathbf{J}'$ that differ only in one draw, changed from r to s. As before, only Z_r and Z_s can change. Hence,

$$\begin{aligned}\|\mathbf{E}(\mathbf{J}) - \mathbf{E}(\mathbf{J}')\|_2 &\le \frac{a_r}{\pi_r}\|\mathbf{v}_r - \mathbf{o}\|_2 + \frac{a_s}{\pi_s}\|\mathbf{v}_s - \mathbf{o}\|_2 \\ &\le 4\beta\|\mathbf{V}\|_2.\end{aligned} \tag{15}$$

Applying Lemma 6 with $r = d$ and $b_\ell = 4\beta\|\mathbf{V}\|_2$ gives, for every $t > 0$,

$$\Pr\left(\|\mathbf{E}\|_2 \ge t\right) \le 2\exp\left(-\frac{t^2}{32m\beta^2\|\mathbf{V}\|_2^2}\right).$$

Taking $t = 4\sqrt{2m}\,\beta\sqrt{u}\,\|\mathbf{V}\|_2$ gives, for every $u > 0$,

$$\Pr\left(\|\mathbf{E}\|_2 \ge 4\sqrt{2m}\,\beta\sqrt{u}\,\|\mathbf{V}\|_2\right) \le 2e^{-u}. \tag{16}$$

Equivalently,

$$\Pr\left(\|\mathbf{E}\|_2 \ge \frac{4\sqrt{2}\,A_{\mathcal{R}}\rho}{\kappa(c)}\sqrt{\frac{u}{m}}\,\|\mathbf{V}\|_2\right) \le 2e^{-u}. \tag{17}$$

Now, fix the failure probability $\delta \in (0,1)$ and set

$$u = \log\frac{4}{\delta}.$$

Then $2e^{-u} = \delta/2$. By eq. 11, with probability at least $1 - \delta/2$,

$$|\Delta_D| \le \frac{2\sqrt{2}\,A_{\mathcal{R}}\rho}{\kappa(c)}\sqrt{\frac{\log(4/\delta)}{m}} = \epsilon_D(\delta).$$

Similarly, by eq. 17, with probability at least $1 - \delta/2$,

$$\|\mathbf{E}\|_2 \leq \frac{4\sqrt{2}\, A_{\mathcal{R}}\rho}{\kappa(c)}\sqrt{\frac{\log(4/\delta)}{m}}\,\|\mathbf{V}\|_2.$$

A union bound shows that both inequalities hold simultaneously with probability at least $1 - \delta$.

By the definition of Δ_D,

$$\frac{\widehat{D}}{D} = 1 + \Delta_D.$$

Consequently,

$$\widehat{\mathbf{o}} - \mathbf{o} = \frac{\widehat{\mathbf{N}}}{\widehat{D}} - \mathbf{o} = \frac{\widehat{\mathbf{N}} - \mathbf{o}\widehat{D}}{\widehat{D}} = \frac{\mathbf{E}}{1 + \Delta_D}. \tag{18}$$

On the simultaneous concentration event, the assumption $\epsilon_D(\delta) < 1$ implies

$$1 + \Delta_D \geq 1 - |\Delta_D| \geq 1 - \epsilon_D(\delta) > 0.$$

Therefore,

$$\begin{aligned}\|\widehat{\mathbf{o}} - \mathbf{o}\|_2 = \frac{\|\mathbf{E}\|_2}{1 + \Delta_D} &\leq \frac{\|\mathbf{E}\|_2}{1 - \epsilon_D(\delta)} \\ &\leq \frac{4\sqrt{2}\, A_{\mathcal{R}}\rho}{\kappa(c)\big(1 - \epsilon_D(\delta)\big)}\sqrt{\frac{\log(4/\delta)}{m}}\,\|\mathbf{V}\|_2.\end{aligned}$$

This proves eq. 3 with probability at least $1 - \delta$. ■

A.3 Proof of Corollary 3

Proof Let

$$x := \epsilon_D(\delta) = \frac{2\sqrt{2}\, A_{\mathcal{R}}\rho}{\kappa(c)}\sqrt{\frac{\log(4/\delta)}{m}}.$$

The assumed lower bound on m gives

$$x \leq \frac{2\sqrt{2}}{\sqrt{21}\,\kappa(c)}\,\frac{\varepsilon}{2 + \varepsilon}.$$

Under the Algorithm 1 allocation rule, $c < 1$, so $\kappa(c) > 1 - e^{-1}$. Since

$$\frac{8}{(1 - e^{-1})^2} < 21,$$

we have

$$x \leq \frac{\varepsilon}{2 + \varepsilon} < 1.$$

Thus the denominator-stability condition in Theorem 2 holds.

The numerator of the bound in Theorem 2 equals $2x\|\mathbf{V}\|_2$. Therefore, with probability at least $1 - \delta$,

$$\|\widehat{\mathbf{o}} - \mathbf{o}\|_2 \leq \frac{2x}{1 - x}\|\mathbf{V}\|_2 \leq \frac{2\varepsilon/(2 + \varepsilon)}{1 - \varepsilon/(2 + \varepsilon)}\|\mathbf{V}\|_2 = \varepsilon\|\mathbf{V}\|_2.$$

■

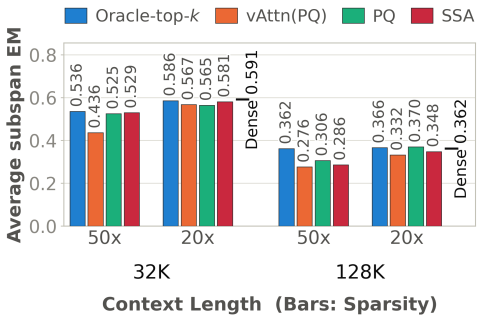
A.4 Additional Experiments

Table 4: Average performance on RULER-HARD-32K across different models at 50× sparsity.

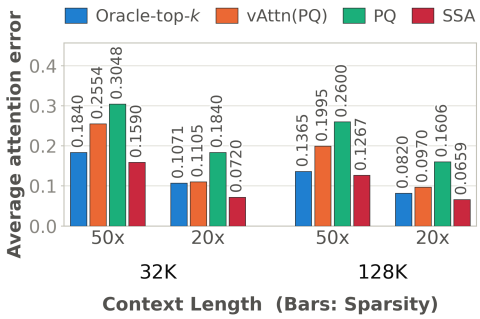
Model	Dense	PQCache	vAttn(PQ)	Quest	HashAttn	Double Sparsity	MagicPig
Qwen3-4B-Instruct-2507	88.67	**85.80**	85.67	42.03	–	39.29	18.72
Llama-3.2-3B-Instruct	66.25	39.30	**39.59**	34.11	–	31.21	17.22
Llama-3.1-8B-Instruct	87.89	**68.95**	64.42	64.17	49.48	44.62	16.28

Table 5: Evaluation datasets and subsets used in our experiments.

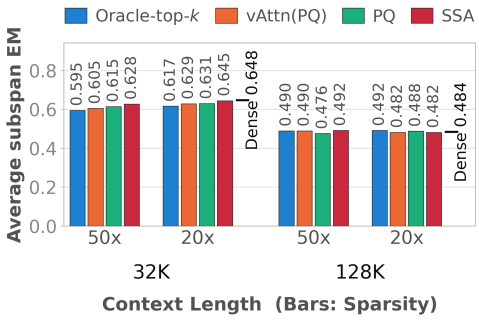
Dataset	# Samples	Subsets
RULER-HARD	100	fwe, qa1, qa2, niah_multikey_2, niah_multikey_3, vt
LongBench	200	NrtvQA, Qasper, MF-en, HotpotQA, 2WikiMQA, Musique, DuReader, GovRep, QMSum, MultiNews, VCSum, TREC, TriviaQA, SAMSum, PCount, PR-en, LCC, RBP
LOFT	100	HotpotQA, Musique, NQ, Qampari, Quest
AIME	16	–



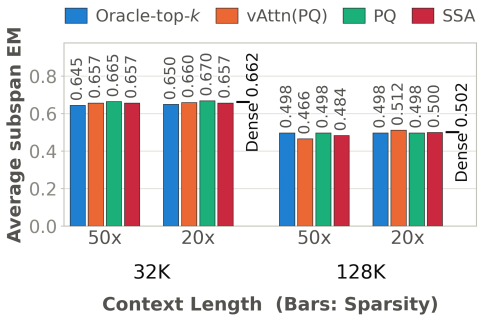
(a) Ministral-3-8B: Subspan-EM



(b) Ministral-3-8B: Error



(c) Qwen3.5-4B: Subspan-EM



(d) Qwen3.5-9B: Subspan-EM

Figure 4: Average Subspan-EM and error across different context lengths and sparsity levels on LOFT.

Table 6: Performance on RULER-HARD with Llama-3.1-8B.

Sparsity	Context	Method	FWE	VT	NM2	NM3	QA1	QA2	Avg.
50×	32K	Dense	93.33	99.2	100	100	88	52	88.76
		PQ	77.67	95.4	67	71	76	48	72.51
		vAttn (PQ)	83.67	94.6	52	58	66	46	66.71
		Oracle-top-k	81.00	97.8	87	76	79	52	78.80
		SSA	91.00	98.0	83	92	85	53	**83.67**
	64K	Dense	86.33	98.8	98	99	84	53	86.52
		PQ	80.00	92.2	65	50	71	47	67.53
		vAttn (PQ)	83.33	94.0	40	68	67	45	66.22
		Oracle-top-k	79.67	95.8	78	56	77	45	71.91
		SSA	85.33	96.2	67	84	77	53	**77.09**
	128K	Dense	73.33	66.2	91	67	83	47	71.26
		PQ	66.00	39.4	27	15	57	34	39.73
		vAttn (PQ)	71.67	55.0	21	20	41	28	39.45
		Oracle-top-k	61.00	66.6	71	32	66	40	**56.10**
		SSA	73.00	54.2	30	30	59	39	47.53
20×	32K	PQ	87.33	98.6	97	93	84	52	85.32
		vAttn (PQ)	93.33	99.4	97	99	84	52	87.46
		Oracle-top-k	90.33	98.8	100	99	84	52	87.36
		SSA	93.67	99.0	97	100	87	54	**88.45**
	64K	PQ	86.33	96.2	91	91	77	55	82.76
		vAttn (PQ)	86.00	98.4	87	98	79	57	84.23
		Oracle-top-k	85.33	98.0	94	95	81	52	84.22
		SSA	85.33	98.6	94	99	81	54	**85.32**
	128K	PQ	69.00	65.8	66	46	68	34	58.13
		vAttn (PQ)	73.33	63.6	61	51	69	39	59.49
		Oracle-top-k	67.00	63.2	85	51	72	42	63.37
		SSA	75.00	69.0	70	63	76	44	**66.17**

Table 7: Performance on LongBench with Qwen3.5-27B (10K-16K context).

Sparsity	Method	NrtvQA	Qasper	MF-en	MF-zh	HotpotQA	2WikiMQA	Musique	DuReader	GovRep	QMSum
50×	Dense	20.49	29.64	38.94	42.97	34.83	34.64	26.69	25.92	31.91	21.96
	PQ	20.13	28.69	38.60	42.01	33.27	31.92	20.61	25.54	31.92	21.64
	vAttn(PQ)	21.97	25.33	39.38	43.53	32.00	30.44	21.45	26.91	30.68	22.42
	SSA	20.42	27.51	38.87	42.13	34.64	32.79	24.88	24.82	31.96	22.72
	Oracle-top-k	20.51	28.97	38.22	42.12	34.79	33.77	22.50	25.76	32.86	22.52
20×	PQ	20.61	29.55	38.41	41.76	34.51	33.38	23.90	25.74	31.63	22.31
	vAttn(PQ)	21.55	27.17	40.14	43.67	33.39	34.18	25.65	26.70	32.20	22.27
	SSA	20.90	28.80	39.73	42.91	34.41	33.97	25.24	25.93	31.36	22.58
	Oracle-top-k	20.74	29.28	38.68	42.87	34.90	34.57	23.46	25.82	32.77	21.97

Sparsity	Method	MultiNews	VCSUM	TREC	TriviaQA	SAMSum	PCount	PRe-en	PRe-zh	LCC	RB-P	Avg.
50×	Dense	23.33	14.50	72.50	89.65	45.63	5.16	88.12	57.33	67.96	59.12	41.56
	PQ	24.37	14.50	71.25	89.68	45.92	6.37	87.40	56.60	66.65	54.97	40.60
	vAttn(PQ)	23.17	14.62	71.25	89.04	43.76	6.85	85.25	59.21	66.46	53.98	40.39
	SSA	23.08	14.35	72.50	89.77	44.32	5.16	86.25	59.69	64.55	58.01	40.92
	Oracle-top-k	24.62	14.66	72.50	89.66	46.91	5.28	87.81	53.90	68.56	59.06	**41.25**
20×	PQ	24.20	14.79	70.00	89.46	45.83	5.10	89.06	54.21	67.87	57.24	40.98
	vAttn(PQ)	23.41	15.05	75.00	90.52	45.12	4.68	86.88	57.23	66.02	58.56	41.47
	SSA	24.01	14.67	72.50	90.18	46.93	4.76	88.12	56.81	66.85	60.47	**41.56**
	Oracle-top-k	24.75	14.29	71.25	89.66	46.25	5.26	87.71	55.04	68.84	59.90	41.40

Table 8: Performance on RULER-HARD with Ministral-3-3B.

Sparsity	Context	Method	FWE	VT	NM2	NM3	QA1	QA2	Avg.
50×	32K	Dense	96.33	91.8	98	99	79	56	86.69
		PQ	92.67	60.2	91	56	73	52	70.81
		vAttn (PQ)	90.33	72.2	67	77	65	44	69.26
		Oracle-top-k	93.67	62.2	99	55	75	52	72.81
		SSA	95.67	88.6	92	90	72	54	**82.05**
	64K	Dense	74.33	99.8	93	96	75	55	82.19
		PQ	72.67	94.6	77	51	69	55	69.88
		vAttn (PQ)	76.00	95.6	54	75	52	42	65.77
		Oracle-top-k	75.33	98.2	89	68	68	52	75.09
		SSA	75.00	99.2	76	87	67	50	**75.70**
	128K	Dense	92.33	98.6	95	87	66	50	81.49
		PQ	77.67	99.2	66	43	66	47	66.48
		vAttn (PQ)	89.33	98.0	54	42	54	41	63.06
		Oracle-top-k	88.00	98.2	79	51	66	49	**71.87**
		SSA	91.67	98.2	60	67	60	41	69.65
20×	32K	PQ	93.33	67.6	98	86	79	54	79.66
		vAttn (PQ)	95.33	90.8	97	98	76	52	84.86
		Oracle-top-k	96.00	73.0	99	93	75	52	81.33
		SSA	95.67	94.2	99	97	79	56	**86.81**
	64K	PQCache	77.00	98.4	87	84	70	56	78.73
		vAttn (PQ)	73.00	99.8	86	96	73	53	80.13
		Oracle	77.00	99.2	93	91	67	52	79.87
		SSA	75.33	99.8	86	96	73	56	**81.02**
	128K	PQ	88.33	99.2	71	65	70	50	73.92
		vAttn (PQ)	89.33	98.4	59	77	65	44	72.12
		Oracle-top-k	91.00	98.4	85	75	68	50	**77.90**
		SSA	90.33	98.6	75	81	67	46	76.32

Table 9: Performance on RULER-HARD with Qwen3.5-4B at 50× sparsity.

Context	Method	FWE	VT	NM2	NM3	QA1	QA2	Avg.
32K	Dense	99.67	100	100	100	89	63	91.95
	PQ	97.33	100	100	100	88	62	91.22
	vAttn (PQ)	96.33	100	100	100	89	58	90.56
	Oracle-top-k	99.33	100	100	100	88	61	91.39
	SSA	99.67	100	100	100	89	60	**91.45**
64K	Dense	99.67	100	100	100	88	59	91.11
	PQ	98.33	100	100	100	88	58	90.72
	vAttn (PQ)	99.67	100	100	100	87	57	90.61
	Oracle-top-k	99.33	100	100	100	86	57	90.39
	SSA	99.00	100	100	100	88	58	**90.83**
128K	Dense	96.67	100	100	100	84	56	89.45
	PQ	95.33	100	100	100	82	55	88.72
	vAttn (PQ)	96.67	100	100	100	82	57	89.28
	Oracle-top-k	97.33	100	100	100	82	57	89.39
	SSA	97.00	100	100	100	82	58	**89.50**

Table 10: Performance on RULER-HARD with Qwen3.5-9B at 50× sparsity.

Context	Method	FWE	VT	NM2	NM3	QA1	QA2	Avg.
32K	Dense	99.67	100	100	100	93	68	93.45
	PQ	94.00	100	100	100	91	67	92.00
	vAttn (PQ)	98.00	100	100	100	89	63	91.67
	Oracle-top-k	98.33	100	99	100	91	67	92.56
	SSA	100.00	100	100	100	93	67	**93.33**
64K	Dense	99.33	100	100	100	88	66	92.22
	PQ	95.67	99.8	100	100	86	65	91.08
	vAttn (PQ)	98.00	99.8	100	99	88	63	91.30
	Oracle-top-k	98.67	100	100	100	89	66	92.28
	SSA	99.33	99.8	100	100	89	66	**92.36**
128K	Dense	99.67	100	100	100	87	61	91.28
	PQ	97.67	100	98	100	88	62	90.95
	vAttn (PQ)	98.67	100	98	100	84	63	90.61
	Oracle-top-k	99.33	100	99	100	88	62	91.39
	SSA	99.67	100	100	100	88	61	**91.45**

Table 11: Performance on RULER-HARD with Ministral-3-8B.

Context	Sparsity	Method	FWE	VT	NM2	NM3	QA1	QA2	Avg.
32K	50×	Dense	94.00	100.00	99.00	100.00	80.00	63.00	89.33
		PQ	88.00	100.00	97.00	67.00	80.00	59.00	81.83
		vAttn (PQ)	89.33	98.80	85.00	87.00	64.00	48.00	78.69
		SSA	**94.00**	**100.00**	88.00	**91.00**	74.00	58.00	84.17
		Oracle-top-k	90.67	**100.00**	**99.00**	90.00	**81.00**	**59.00**	**86.61**
64K	50×	Dense	81.00	99.80	100.00	100.00	75.00	61.00	86.13
		PQ	84.33	99.80	89.00	71.00	68.00	56.00	78.02
		vAttn (PQ)	81.33	99.80	71.00	87.00	63.00	49.00	75.19
		SSA	81.67	99.80	80.00	**95.00**	69.00	54.00	79.91
		Oracle-top-k	**84.67**	99.80	**100.00**	82.00	**75.00**	**59.00**	**83.41**
128K	50×	Dense	92.67	100.00	100.00	90.00	62.00	44.00	81.45
		PQ	79.33	99.60	63.00	32.00	63.00	41.00	62.99
		vAttn (PQ)	89.67	**100.00**	36.00	61.00	59.00	37.00	63.78
		SSA	**90.67**	99.80	50.00	**71.00**	62.00	42.00	69.25
		Oracle-top-k	85.00	**100.00**	**93.00**	67.00	**64.00**	**46.00**	**75.83**
32K	20×	PQ	92.33	100.00	**99.00**	97.00	**83.00**	60.00	**88.56**
		vAttn (PQ)	**95.00**	100.00	96.00	**100.00**	78.00	61.00	88.33
		SSA	93.33	100.00	97.00	99.00	77.00	**63.00**	88.22
		Oracle-top-k	92.00	100.00	**99.00**	98.00	78.00	**63.00**	88.33
64K	20×	PQ	**83.67**	99.80	99.00	97.00	**74.00**	**60.00**	**85.58**
		vAttn (PQ)	83.33	99.80	94.00	99.00	72.00	57.00	84.19
		SSA	81.67	99.80	95.00	**100.00**	73.00	58.00	84.58
		Oracle-top-k	83.00	99.80	**100.00**	95.00	71.00	**60.00**	84.80
128K	20×	PQ	85.33	**100.00**	88.00	80.00	**66.00**	44.00	77.22
		vAttn (PQ)	91.67	99.80	77.00	**89.00**	62.00	43.00	77.08
		SSA	**93.33**	99.80	81.00	**89.00**	62.00	42.00	77.86
		Oracle-top-k	90.67	**100.00**	**99.00**	84.00	64.00	**44.00**	**80.28**

Table 13: Comparison of theoretical and practical SSA variants on average RULER-HARD-32K performance.

Model	**Sparsity**	**SSA-Theory**	**SSA-Practical**
Llama-3.1-8B	50×	81	83
Llama-3.1-8B	20×	88	88
Ministral-3-8B	50×	82	84
Ministral-3-8B	20×	89	88

Table 12: Ablation on the heavy-hitter fraction h for SSA on AIME 2025. A fraction h of the sparse attention budget is allocated to deterministic heavy hitters, while the remaining $1 - h$ is allocated to importance sampling. Thus, $h = 0$ corresponds to pure importance sampling and $h = 1$ corresponds to PQCache (pure top-k retrieval).

Heavy Fraction h	**20**×	**10**×
0.0	40.2	42.5
0.2	61.0	63.8
0.4	63.5	**64.0**
0.6	62.7	62.7
0.8	**63.8**	63.1
1.0 (PQ)	49.4	53.8