CALITOPP: NATIVELY TRAINABLE TOP-p SPARSE ATTENTION VIA SET-MASS CALIBRATION

Anonymous authors
Paper under double-blind review

ABSTRACT

As context windows grow, the quadratic cost of full attention becomes the main bottleneck of long-context LLM inference, motivating sparse attention. Learning lightweight indexers for token selection is emerging as a mainstream approach because it maintains consistency between training and inference. However, these indexers almost universally use fixed top-k budgets, overlooking the vast differences in token requirements across tasks and models. In contrast, top-p selection retains a target fraction of attention mass, providing a more reliable accuracy guarantee and an input-adaptive budget. Top-k selection depends only on token rankings, whereas top-p additionally depends on cumulative probability mass. We show theoretically and empirically that KL-distilled top-k indexers can accurately rank tokens yet underestimate the cumulative mass of important tokens, leading to severe over-selection when reused for top-p selection. Based on this finding, we propose CaliTopP, a trainable token-level top-p sparse-attention framework. Its parameter-free budget-calibration loss $\mathcal{L}_{\mathrm{bud}}$ aligns the indexer's selected budget with the model's true requirement without adding parameters or reducing the recall of important tokens. With only two lightweight training stages, CaliTopP transfers across diverse architectures and achieves up to a $5.36\times$ speedup over dense attention while preserving generation quality.

1 INTRODUCTION

In recent years, large language models (LLMs) have achieved transformative breakthroughs in natural language processing (NLP) and, more broadly, across artificial intelligence (OpenAI, 2023). As application scenarios grow increasingly complex — repository-level code generation, long-document analysis, multi-turn dialogue — the context window of LLMs has expanded from the early 2K to 1M and beyond (DeepSeek-AI, 2026; Team, 2026; Qiu et al., 2026). Since the computational complexity of self-attention grows quadratically with sequence length, ever-longer contexts cause inference latency to climb sharply, which has become the core obstacle to extending context length further (Zhang et al., 2025; 2023).

Existing studies have shown that attention computation in large models is sparse: only a small number of key tokens play a decisive role in the computational results, while the vast majority of tokens have negligible impact (Miao et al., 2026; Ge et al., 2024). Sparse attention mechanisms exploit precisely this property by computing attention only for the selected key tokens, thereby reducing computational overhead. Existing selection mechanisms follow two paths: *training-free* and *trainable* (DeepSeek-AI, 2025; Zhou et al., 2026b; Team, 2026; Cai et al., 2024). Training-free methods rely on block-level selection, proxy scores, or quantization. Block-level selection methods such as Quest (Tang et al., 2024) and SnapKV (Li et al., 2024) treat an entire block as a single unit of selection. However, because important tokens are sparsely distributed in LLMs, such methods inevitably recall many irrelevant neighboring tokens. Precise per-token recall instead requires token-level proxy scores, but training-free proxies typically accelerate only a single phase: PQCache (Zhang et al., 2025) and RaBitQCache (Li et al., 2026) must construct their quantization codebooks during prefill and therefore accelerate decode only. More fundamentally, all these training-free methods are applied to models that have already been trained with full attention, which inevitably introduces train–inference inconsistency and thus degrades model performance. Trainable methods instead learn a lightweight indexer that performs the selection: DSA (DeepSeek-AI, 2025) reaches per-token granularity, sparsifies both prefill and decode, and — because the back-

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.

bone is trained together with the sparse pattern — lets the model learn the sparse distribution rather than merely tolerate it. Trainable sparse attention is consequently being adopted by frontier model providers (Lai et al., 2026; GLM, 2026).

Trainable sparse attention today almost universally adopts a fixed top-k retrieval scheme (DeepSeek-AI, 2025; GLM, 2026; Zhou et al., 2026a). However, the retrieval budget actually needed varies enormously with the query, the downstream task, and the model itself, so a fixed k is redundant for some tasks yet insufficient for others (Appendix D). This has motivated query-adaptive top-p recall, which keeps the minimal token set whose estimated attention mass reaches a threshold p, providing a more reliable accuracy guarantee and an input-adaptive budget. RaBitQCache (Li et al., 2026) realizes dynamic top-p recall through an unbiased proxy-score estimator, but it must construct its codebooks during the prefill stage and can therefore only accelerate the decode stage. RTPurbo (Zhou et al., 2026b) takes a first step toward a trainable top-p mechanism: it inherits DSA's KL-distilled indexer unchanged and directly replaces the top-k operator with top-p during the decode stage. This research does not focus on top-p indexer, so it does not conduct in-depth exploration of it, nor does it report the degree of budget over-selection in the paper. We show both theoretically and experimentally that KL distillation alone can yield severe top-p budget inflation under limited indexer capacity (§3). A top-k operator depends only on token rankings, whereas a top-p cut relies on absolute cumulative mass. A small underestimate of this mass can require many additional tokens when the remaining probability mass is spread over a long, diffuse tail. Although lowering p can mitigate over-selection, its severity varies across models and tasks, so a uniform reduction can hurt accuracy (Figure 5). This requires manual threshold tuning for each setting, undermining the original motivation for top-p: adapting the retrieval budget without such tuning. How to train a per-token top-p indexer therefore remains an open problem.

Our goal is a trainable, per-token, top-p sparse attention that accelerates both prefill and decode. Three difficulties stand in the way. ❶ A fixed-budget selector only needs to predict the relative ordering among tokens and does not care about the actual magnitudes of the scores, whereas top-p recall requires the proxy to faithfully recover the true values, and achieving this with a lightweight proxy is not easy. ❷ The method must keep additional training overhead acceptable and ideally avoid extra parameters, facilitating integration with existing optimization techniqueas (Bai et al., 2026). ❸ The method must apply to different model architecture such as GQA and MLA (Bi et al., 2024; Ainslie et al., 2023; DeepSeek-AI, 2024), so that it can be used on various models.

In this paper, we propose CaliTopP, a trainable top-p sparse-attention framework. We first derive through theoretical and experimental analysis why the KL-divergence training scheme used by current top-k sparse attention leads to over-selection when the selection mechanism is changed to top-p. Based on this analysis, we propose $\mathcal{L}_{\mathrm{bud}}$, a parameter-free budget-calibration loss that aligns the token budget selected by the indexer with the budget prescribed by applying top-p to the teacher's attention distribution. We jointly optimize KL and $\mathcal{L}_{\mathrm{bud}}$ on the same indexer weights: KL encourages fidelity to the teacher's attention distribution, while $\mathcal{L}_{\mathrm{bud}}$ directly supervises the selection budget. The resulting indexer recalls key tokens with a computational budget close to the true requirement without new parameters. CaliTopP consists of two short and lightweight training stages: we first freeze the backbone to warm up the indexer, and then perform joint training so that the model adapts to the sparse distribution. Building on this lightweight index for task-adaptive top-p retrieval, CaliTopP provides an efficient, accurate, and general acceleration solution for long-context inference. Extensive experiments show that CaliTopP can be applied to different model architectures and significantly outperforms SOTA baselines.

Our contributions are as follows:

- We analyze why applying top-p to KL-trained top-k indexers causes over-selection and propose $\mathcal{L}_{\mathrm{bud}}$, a parameter-free loss that calibrates the selected-token budget to the teacher's top-p budget.
- We build CaliTopP, a trainable per-token top-p sparse attention framework. It keeps selection identical between training and inference, requires only two short training stages, and accelerates both prefill and decode across different attention architectures.
- We conduct extensive and comprehensive evaluations, and the results show that our method adapts to a variety of architectures, delivering up to a $5.36\times$ speedup with almost no drop in model performance.

2 Related Work

2.1 Sparse Attention

The standard attention mechanism computes the output o at step t from the query q_t, keys K, and values V as $o = \mathrm{softmax}(q_t K^\top/\sqrt{d})\, V$ (Vaswani et al., 2017). Sparse attention approximates this by computing the softmax over only a subset of indices $\mathcal{I}$ and dropping the rest. Because the value vectors are distributed smoothly, the error of this approximation is governed by the attention mass left outside $\mathcal{I}$ (Lin et al., 2025; Zhao et al., 2024a), so the objective is to select as few tokens as possible while keeping the dropped mass low, approaching full attention at a significantly reduced cost. Most existing methods adopt the top-k rule, which keeps the k tokens with the highest estimated attention weights and fixes the size of the selected set in advance. A few recent works instead explore top-p selection, which places a stricter requirement on the proxy score, since it needs not only the relative ordering of the tokens but also the actual magnitudes of their scores. The proxy scores are normalized with softmax into a distribution $\tilde{a}$, and the smallest set whose cumulative mass reaches p is selected,

$$\mathcal{I}_p = \arg\min_{\mathcal{I}} |\mathcal{I}| \quad \text{s.t.} \quad \sum_{i\in\mathcal{I}} \tilde{a}_i \geq p. \tag{1}$$

Depending on whether the selection is learned, existing methods follow two paths. Training-free methods impose the selection on a model that has already been trained with full attention, while trainable methods train an index to perform the selection.

Training-free methods select tokens for a frozen model with approximate importance estimates (Wu et al., 2025). Quest (Tang et al., 2024) partitions the KV cache into blocks and selects whole blocks by per-block score estimates. Because important tokens are sparsely distributed, many irrelevant tokens inside the selected blocks are recalled together. PQCache (Zhang et al., 2025) and RaBitQCache (Li et al., 2026) estimate per-token importance with quantization codebooks, and the latter further enables dynamic top-p recall through an unbiased estimator. However, the codebooks must be built during prefill, so only decode is accelerated.

Trainable methods train an index to perform the selection. DSA (DeepSeek-AI, 2025) reaches per-token granularity in both prefill and decode with a distilled indexer, whose recipe we detail in §2.2, but its budget is a fixed top-k. RTPurbo (Zhou et al., 2026b) conducts a preliminary exploration of top-p sparse attention. It replaces the top-k operator with top-p, yet its indexer is still trained with KL alone. We prove that this method leads to severe over-selection under top-p (§3); moreover, its index is used only at the decode stage. There is currently a blank in the research of per-token trainable top-p sparse attention.

2.2 DeepSeek Sparse Attention

DeepSeek-V3.2 introduces DSA (DeepSeek-AI, 2025), which trains a lightweight indexer that assigns each history token a score s_i and distills it from the model's own attention. Let a denote the dense attention distribution of the current query, which serves as the teacher of this distillation, and let $\tilde{a} = \mathrm{softmax}(s)$ be the normalized proxy distribution of §2.1. The indexer is trained by minimizing $\mathrm{KL}(a \,\|\, \tilde{a})$ and, at inference, the top-k tokens under s are selected, with k=2048 as the official default. This recipe is sufficient for top-k selection, because the operator consumes only the ordering of the scores.

3 Why KL-Distilled Indexers Over-Select under Top-p

In this section we analyze the defects of the existing DSA recipe (§2.2) and why its training scheme cannot be reused for top-p selection unchanged. We first show on DeepSeek-V3.2 that a fixed budget does not fit the varying context lengths of realistic workloads. We then explain theoretically what goes wrong once the same indexer is consumed by a top-p rule and verify the prediction experimentally, and the combination of the two leads to the insight behind our method.

Limitations of Fixed Budgets. In deployment, context lengths vary widely across tasks, and one fixed budget cannot serve them all. To quantify the mismatch, we ran DeepSeek-V3.2 on RULER dataset (Hsieh et al., 2024) from 4K to 128K and recorded the dense attention distribution of the

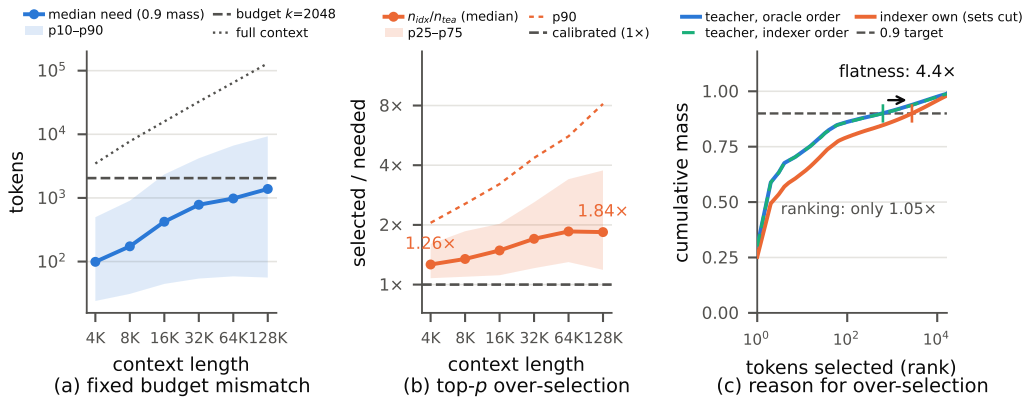


Figure 1: Over-selection of DeepSeek-V3.2's DSA indexer under top-p

query at the first decoded token; for each such query we then count the minimal number of history tokens whose attention probabilities sum to 0.9. Figure 1a reports these counts against the official default k=2048. The required budget differs drastically across context lengths and keeps growing with length. At 4K the default k=2048 is 21× the median need, while at 128K it is exceeded by 39% of queries, and the attention mass lost to this shortfall directly degrades model performance. Nor is length the only axis of the mismatch; our experiments in Appendix D further show that the required budget also differs sharply across the layers of the same model and shifts dynamically with the query task. The budget should therefore vary with the query. To achieve this within trainable sparse attention, an intuitive route is to only replace the top-k operator with the top-p rule of Eq. 1, which introduces no new parameters and by construction keeps the estimated recalled mass at p.

Why KL Alone Can Lead to Over-Selection. Fix a query, writing a for its teacher distribution and $\tilde{a} = \mathrm{softmax}(s)$ for the indexer's proxy as in §2.2. Minimizing $\mathrm{KL}(a \,\|\, \tilde{a})$ encourages the indexer to match the teacher's attention probabilities, but does not directly supervise the number of tokens selected by top-p. A lightweight indexer may identify the important tokens correctly while failing to reproduce their probability mass accurately. In particular, it may assign slightly too little total mass to an important prefix and spread the remaining mass over many background tokens. The proxy then appears flatter around the selection boundary: the same high-ranked tokens accumulate less probability than they do under the teacher. This mismatch can remain small in KL while having a large effect on the selected budget, and can persist even at a KL optimum under limited representational flexibility. For readability, we keep this qualitative explanation in the main text and defer a formal construction and proof of this failure mode to Appendix A (Proposition 1). DSA's top-k operator does not notice this calibration bias when the token ordering is preserved, because it consumes only the importance ordering of the scores. The top-p rule of Eq. 1 instead cuts by cumulative probability mass, keeping the shortest prefix of the proxy's sorted distribution that reaches p. If the proxy underestimates the mass of the important prefix, the cut must extend further into the background to reach the same threshold. Since each background token contributes little probability, recovering even a small mass deficit can require many additional tokens. A KL-distilled indexer used with top-p can therefore substantially over-select despite accurate token rankings, motivating explicit budget supervision alongside KL distillation.

Empirical Evidence of Over-Selection. We verify the above finding experimentally. We replaced the selection strategy of DeepSeek-V3.2 indexer with top-p=0.9 and repeated the RULER sweep from 4K to 128K, comparing for each query the teacher budget n_{tea}, the number of tokens the dense attention a itself needs to reach mass p, with the indexer budget n_{idx}, the number kept by applying the top-p rule to $\tilde{a}$. As shown in Figure 1b, the indexer systematically over-selects, and the excess widens with context length, from 1.26× at 4K to 1.84× at 128K in the median and beyond 8× at the 90th percentile. We also investigate the cause of the over-selection. We accumulate the teacher's attention mass along two orderings, the teacher's own and the one induced by the indexer's scores, and record the number of tokens each ordering needs to reach mass 0.9, together with the position of the indexer's actual top-p cut. As shown in Figure 1c, mis-ranking costs a factor of only 1.05, while flatness costs 4.4. These measurements indicate that ranking errors account for only a small part of the excess budget in this setting. Most of the excess comes from the proxy underestimating cumulative attention mass, consistent with the tail-amplification mechanism analyzed above.

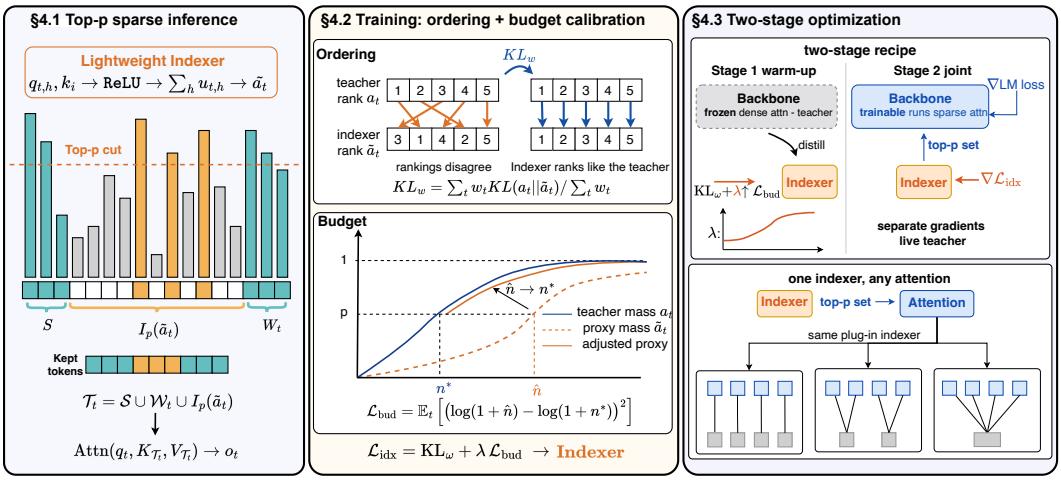


Figure 2: Overview of CaliTopP. Left: top-p token selection with sink and window tokens. Middle: indexer training with KL distillation and budget calibration via $\mathcal{L}_{\text{bud}}$. Right: indexer warm-up with a frozen backbone, followed by joint training.

Calibrating the Top-p Budget. KL alone can inflate top-p budgets, motivating explicit budget supervision. Fixed temperatures can be absorbed into existing weights, but global or per-layer scales cannot independently correct each query's mass errors. A separate learned temperature predictor adds parameters. Appendix A.3 discusses these limitations. **Our goal is to avoid over-selection without disturbing the ranking that KL already trains and without introducing new parameters that would raise the training cost.** Our insight is that during training the teacher provides n_t^* at negligible cost: the shortest prefix length in the indexer's ordering that covers teacher mass p, matching the oracle budget under correct ranking and expanding as needed to compensate for ranking errors.

$$\mathcal{L}_{\text{bud}} = \mathbb{E}_t\Big[\big(\log(1+\hat{n}_t) - \log(1+n_t^*)\big)^2\Big], \tag{2}$$

where the logarithm absorbs the order-of-magnitude spread of budgets across queries and layers, and $\hat{n}_t$ is the number of tokens that the cut rule of Eq. 1 keeps on the indexer's distribution. Concretely, sorting the indexer's distribution of query t in descending order and writing $c_{t,i}$ for the cumulative mass ranked strictly before token i, the cut keeps exactly the tokens with $c_{t,i} < p$, and we count them as

$$\hat{n}_t = \sum_i \sigma\big((p - c_{t,i})/\tau\big), \tag{3}$$

where σ is the sigmoid function and τ a small smoothing width parameter, so that the count varies smoothly with the scores instead of jumping in integer steps that pass no gradient. Empirically, this relaxation supports stable optimization. On DeepSeek-V2-Lite, the logged indexer loss decreases from 8.72×10^{-3} to 7.74×10^{-3} after the calibration-weight warm-up, and stabilizes toward the end of training. Meanwhile, the budget is effectively reduced (Figure 6 and Figure 7, Appendix C). The ablation of $\mathcal{L}_{\text{bud}}$ (§5.3, Figure 4) further shows that training with this surrogate reduces the actual token budget under discrete top-p selection while maintaining comparable generation quality, demonstrating its practical effectiveness for training top-p indexers. $\mathcal{L}_{\text{bud}}$ introduces no additional indexer parameters and is jointly optimized with KL on the existing weights. KL encourages fidelity to the teacher's attention distribution, while $\mathcal{L}_{\text{bud}}$ directly penalizes deviations from the teacher-derived budget target.

4 CaliTopP

Figure 2 summarizes CaliTopP, a trainable top-p sparse attention framework. During inference, the indexer estimates an attention distribution over past tokens for each query. We apply top-p selection to this distribution and compute attention over the selected tokens, together with the initial and most recent tokens (§4.1). Then we train the indexer with KL distillation and $\mathcal{L}_{\text{bud}}$ to approximate the teacher's attention distribution and calibrate the number of selected tokens (§4.2). Training consists of two short stages: we first train the indexer while freezing the backbone, then train them jointly to adapt the backbone to sparse attention (§4.3).

4.1 Inference with Top-p Selection

We inherit the lightweight scorer of DSA (§2.2). For query t the indexer projects the query into a small number of heads $q_{t,h}$ and every history token i into a shared key k_i, and scores $s_{t,i} = \sum_h u_{t,h} \operatorname{ReLU}\big(q_{t,h}^\top k_i\big)$ with learned per-head weights $u_{t,h}$; the proxy distribution is $\tilde{a}_t = \operatorname{softmax}(s_t)$ over the causal history. Prior work observes that the tokens at the beginning of the sequence and those immediately preceding the query are important for the vast majority of queries, so we keep them directly (Xiao et al., 2024). Writing $\mathcal{S}$ for the first n_s tokens (the sink), $\mathcal{W}_t$ for the w tokens immediately preceding query t (the window), and $\mathcal{M}_t$ for the remaining history (the middle), attention at query t is computed only over

$$\mathcal{T}_t \;=\; \mathcal{S} \cup \mathcal{W}_t \cup \mathcal{I}_p(\tilde{a}_t), \tag{4}$$

where $\mathcal{I}_p$ is the top-p rule of Eq. 1 applied to $\tilde{a}_t$; the attention output is then computed on the kept tokens alone, $o_t = \operatorname{Attn}\big(q_t, K_{\mathcal{T}_t}, V_{\mathcal{T}_t}\big)$, where $K_{\mathcal{T}_t}$ and $V_{\mathcal{T}_t}$ are the keys and values of the tokens in $\mathcal{T}_t$. Implementation details are deferred to Appendix B.

4.2 Training Objective

The indexer learns from the original dense model; the target of query t is the model's dense attention distribution a_t, as in §2.2. The objective has two terms. The first trains the ordering. To prevent the queries that attend over long ranges from being drowned out by the far more numerous short-context ones, we weight each row by its effective context length when averaging the KL,

$$\mathrm{KL}_\omega = \frac{\sum_t \omega_t \,\mathrm{KL}(a_t \,\|\, \tilde{a}_t)}{\sum_t \omega_t}, \qquad \omega_t = \operatorname{clip}\big(\ell_t/\ell_0,\, 1,\, \omega_{\max}\big), \tag{5}$$

where ℓ_t is the number of tokens visible to query t, ℓ_0 is a reference length, and clip bounds the weight to $[1, \omega_{\max}]$.

The second term instantiates $\mathcal{L}_{\mathrm{bud}}$ of Eq. 2 on the operator. Because the sink and window are kept unconditionally, the quantity the loss must control is the number of middle tokens admitted by the cut; $\hat{n}_t$ of Eq. 3 is therefore selected over $i \in \mathcal{M}_t$ only, while the cumulative mass $c_{t,i}$ is still computed on the full distribution, exactly as the operator computes it. Sorting the history of query t by the indexer's own scores in descending order, let r_t be the length of the shortest prefix whose cumulative *teacher* mass reaches p, i.e., the number of tokens the current ordering needs in order to cover the mass the operator is asked to keep. The target is the number of middle tokens inside that prefix,

$$n_t^* = \big|\{i \in \mathcal{M}_t : \operatorname{rank}_t(i) \le r_t\}\big|, \tag{6}$$

where $\operatorname{rank}_t(i)$ is the position of token i in that ordering. When the ordering is perfect, n_t^* coincides with the teacher's own middle-token budget; where the ordering is still poor, it grows by exactly the amount needed to keep covering mass p, and it shrinks back toward the oracle count as KL improves the ordering, so the calibration never asks the cut to drop mass the current ranking still needs.

The total indexer loss is

$$\mathcal{L}_{\mathrm{idx}} = \mathrm{KL}_\omega + \lambda\, \mathcal{L}_{\mathrm{bud}}, \tag{7}$$

trained jointly on the same indexer weights with no additional parameters; both terms reach the indexer only through its scores s. The coefficient λ controls the strength of the budget constraint.

4.3 Two-Stage Training

The indexer starts from random initialization and its early selections are unreliable, so we do not expose the backbone to them. Stage 1 starts from a pretrained dense checkpoint and trains the indexer alone with the backbone frozen; the attention it learns from is that of the intact dense model and stays stationary throughout the stage. The dial λ ramps up slowly from zero, so early training is dominated by KL and adjusts the ordering first, with the budget calibration engaging as the ordering stabilizes. Stage 2 unfreezes the backbone and trains the two jointly, letting the backbone adapt to computing through the sparse attention it now actually runs. At every step the indexer learns from the current model's own attention and tracks it as it drifts — and the two gradient streams stay separate. The backbone learns only from the language-modeling loss computed through the sparse output, the indexer only from $\mathcal{L}_{\mathrm{idx}}$, and no gradient flows between the two through the selection.

Table 1: Main results on LongBench v1 and RULER. The results are formatted as Generation Score (Attention Recall). The recall rate is defined as the accumulated attention score of the selected tokens divided by the total score of Full Attention.

		DeepSeek-V2-Lite			Llama-3.1-8B		
Method	Budget	LongBench	RULER	Tokens	LongBench	RULER	Tokens
Full attention	full	40.99 (100%)	76.83 (100%)	–	50.40 (100%)	79.93 (100%)	–
Quest	1024	34.72 (54%)	60.43 (56%)	1020/1021	43.84 (55%)	78.37 (56%)	1020/1021
	2048	36.71 (68%)	66.67 (68%)	2020/2045	46.59 (71%)	79.23 (69%)	2002/2044
	4096	38.63 (82%)	71.93 (84%)	3755/3959	46.70 (84%)	79.30 (85%)	3671/3969
DS	1024	34.54 (58%)	43.93 (60%)	1023/1024	48.05 (71%)	79.43 (72%)	1022/1024
	2048	37.19 (74%)	59.73 (75%)	2023/2048	47.80 (83%)	79.27 (83%)	2005/2048
	4096	39.26 (87%)	70.03 (89%)	3758/3962	47.87 (92%)	79.47 (92%)	3674/3972
SparQ	ratio $= 25\%$	40.28 (89%)	77.30 (90%)	2713/2278	47.73 (89%)	79.73 (90%)	2481/2277
MagicPIG	LSH	37.94 (71%)	66.10 (75%)	1840/1496	46.35 (73%)	70.37 (76%)	1697/1565
RaBitQCache	$p = 0.95$	40.33 (98%)	75.90 (98%)	3628/3376	47.93 (91%)	80.83 (90%)	1923/1849
DSA	$k = 1024$	44.51 (93%)	68.48 (92%)	1032/1048	40.05 (84%)	76.58 (83%)	1017/1049
	$k = 2048$	46.07 (97%)	74.38 (96%)	2138/2176	45.08 (91%)	81.75 (90%)	2113/2176
	$k = 4096$	**46.79 (99%)**	**79.30 (99%)**	3845/4053	49.29 (97%)	**87.28 (98%)**	3755/4064
CaliTopP	$p = 0.95$	**45.90 (98%)**	**78.70 (95%)**	2442/1673	**52.19 (96%)**	**86.13 (97%)**	3110/3271

5 EXPERIMENTS

Implementation and Setup. We build an inference stack on top of vLLM v0.26 (Kwon et al., 2023), and our training stack on top of Megatron-LM (Shoeybi et al., 2019) and ms-swift (Zhao et al., 2024b). The combined implementation exceeds 6,000 lines of code, with Python driving the orchestration and OpenAI Triton together with custom CUDA kernels providing the performance-critical operators. All experiments are run on servers equipped with up to 64 NVIDIA Hopper-architecture GPUs, and the concrete hardware and parallelism configuration used for each study is stated in the corresponding section.

5.1 ACCURACY EVALUATION

Benchmarks and Models. To evaluate generation quality and retrieval accuracy, we use four complementary benchmarks: LongBench v1 (Bai et al., 2024b), LongBench v2 (Bai et al., 2025), RULER (Hsieh et al., 2024), and GSM8K (Cobbe et al., 2021). LongBench v1 and LongBench v2 assess long-context generation and understanding, and RULER stresses retrieval over synthetic long contexts. The first three mainly evaluate the model's capabilities under long-input scenarios, while GSM8K measures mathematical reasoning ability with short context and long output. The model suite spans several architectures and scales: DeepSeek-V2-Lite (DeepSeek-AI, 2024), an MLA–MoE model; Llama-3.1-8B (Team, 2024), a dense GQA model; the gated-attention Qwen3.8-27B (Qwen Team, 2026); and the larger model DeepSeek-V3.2 (DeepSeek-AI, 2025). This selection lets us test whether the proposed top-p selection and calibration objective transfer beyond a single attention or backbone design. The architecture-specific adaptations, including the layers to which the indexer is attached, model-specific training settings, and evaluation details, are provided in Appendices B and C.

Baselines. We compare CaliTopP with the trainable sparse-attention method DSA (DeepSeek-AI, 2025) and with the training-free methods RaBitQCache (Li et al., 2026), Quest (Tang et al., 2024), Double Sparse (Yang et al., 2024), MagicPIG (Chen et al., 2025), and SparQ (Ribar et al., 2024). Unless stated otherwise, each baseline follows the recommended configuration in its official repository, with the concrete per-method settings listed in Appendix C.

Results on LongBench v1 and RULER. To test the model performance within the training length, we select 13 subsets from LongBench v1 , covering all task types, and on RULER we test synthetic long-context retrieval tasks. To provide a comprehensive evaluation, each baseline is tested under multiple token budgets. The aggregated average results are reported in Table 1, and the detailed per-subset LongBench results are given in Tables 4 and 5 (Appendix E). The accuracy degradation of CaliTopP is negligible, and it even surpasses the full-attention baseline in many settings. On DeepSeek-V2-Lite our model retains only 27% of the context on average while preserving 98%

of the full-attention mass on LongBench, which demonstrates the recall efficiency of our indexer. By comparison, the other advanced methods show clear performance gaps under different token budgets. DSA under a fixed budget also drops sharply on several subsets, which shows that a fixed top-k budget cannot fit all scenarios and further supports the superiority of our top-p selection. Although some gains arise from improved backbone capabilities after Stage 2, the ablation in §5.3 shows that CaliTopP already outperforms nearly all baselines on LongBench v1 after Stage 1 and, after Stage 2, exceeds the dense model trained with the same recipe on several subsets. These results support both the adaptability of our top-p indexer and the effectiveness of Stage-2 joint training in learning sparse representations.

Results on LongBench v2. We evaluate Qwen3.8-27B on LongBench v2 (Table 2), whose 8K–2M-token inputs exceed the range of the previous two models and test generalization beyond our indexer's training lengths of less than 32K. Of 503 questions, the original dense model and CaliTopP (S1) fail to produce parsable answers in 173 and 172 cases, respectively. Their scores therefore understate reasoning ability and serve only as references; the fair comparison is between Stage 2 and the dense model trained on the same data. Full results appear in Table 6 (Appendix E). Degradation against the corresponding dense reference is small at both stages: within trained lengths, our model improves by 1.7; beyond training, the drop is at most 5.1. This indicates reasonable length generalization, with the best results within the trained range.

Table 2: LongBench v2 accuracy on Qwen3.8-27B.

Method	Overall	Short	Medium	Long
Original dense	29.0	25.0	28.8	36.1
CaliTopP (S1)	26.4	24.4	24.7	33.3
CaliTopP (S2)	54.1	61.7	49.3	50.9
Dense twin	56.7	60.0	54.4	55.6

Results on GSM8K. We also investigate the capability of our method in short-input and long-output scenarios on GSM8K. Since the prompt itself is short, we use $K{=}256$ for the fixed-budget baselines. Table 7 (Appendix E) reports the complete results. With about a third of the tokens on DeepSeek-V2-Lite, CaliTopP suffers almost no performance degradation with the corresponding dense model on this dataset, and on Llama-3.1-8B its score is even higher than full attention. We also find that DeepSeek-V2-Lite and Llama-3.1-8B require very different token counts on the same task, which further confirms the superiority of the top-p recall rule, since a single parameter setting adapts itself to different tasks and models.

5.2 Efficiency Experiments

Setup. We measure end-to-end latency and per-layer attention costs on DeepSeek V3.2 using vLLM v0.26 with FlashAttention 3. We use DeepSeek V3.2 as the top-k baseline, and its dense variant with the indexer removed as the Full baseline. We test 4K–128K contexts with tensor parallelism over 8 Hopper GPUs and report the mean of four runs after warm-up.

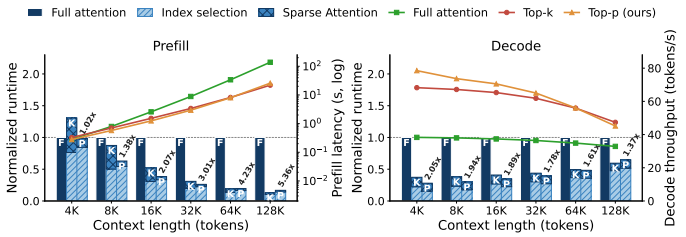


Figure 3: End-to-end efficiency and per-layer breakdown on DeepSeek V3.2.

Overall Speedup. Unlike training-free methods, which can accelerate only one stage, our method achieves substantial speedups in both the prefill and the decode stage, reaching up to $5.36\times$ and $2.05\times$(Figure 3). On the prefill side, at short context our method is faster than native top-k — 0.278 s against 0.328 s at 4K context — because the adaptive budget retains far fewer tokens; the two cross near 64K context, and at 128K, top-p becomes 17% slower than top-k, a decline incurred at that length in order to preserve model capability, which we consider worthwhile. The decode stage exhibits a similar trend.

Performance Breakdown. Figure 3 separates index selection from sparse attention. At 4K, per-layer costs are 0.98 of full attention for CaliTopP and 1.31 for top-k: although top-p selection is more expensive, it reduces the attention stage from 54% to 14%, more than offsetting that overhead. At longer contexts, sparse-attention costs flatten as the selected count changes little, while indexer projections and selection costs grow with length. With fewer indexer heads and lower precision,

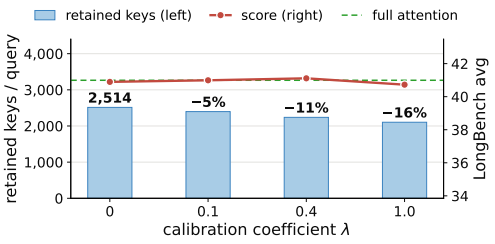


Figure 4: Ablation of $\mathcal{L}_{\text{bud}}$.

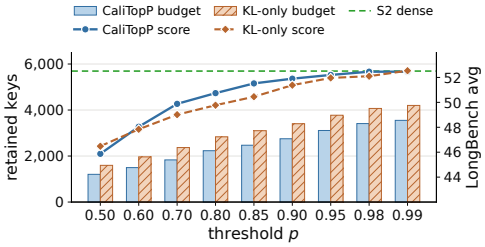


Figure 5: CaliTopP and KL-only across p.

both sparse arms remain much lower than full attention at long context; their per-layer costs are 28–65% of full attention across sequence lengths.

5.3 Ablation Experiments

Ablation of $\mathcal{L}_{\text{bud}}$. We ablate $\mathcal{L}_{\text{bud}}$ on Stage-1 DeepSeek-V2-Lite with a frozen backbone, using $\lambda \in \{0, 0.1, 0.4, 1.0\}$ and fixed p=0.95. From $\lambda = 0$ (KL only) to $\lambda = 1$, the average budget falls by 16.3% with similar LongBench v1 scores (40.90 vs. 40.73; Figure 4); Table 8 gives full results. Per-task reductions range from 3.1% on Qasper to 29.4% on TriviaQA (Figure 10, Appendix E). This variation highlights the need for explicit budget control: over-selection varies across tasks, so we should not simply reduce the budget by lowering p, but instead align the fraction of tokens selected by top-p with the fraction actually required by the teacher.

Two-Stage Training. We evaluate the effects of the two training stages by comparing the sparse models with their dense controls on DeepSeek-V2-Lite and Llama-3.1-8B (Table 9, Appendix E). In Stage 1, training only the indexer with the backbone frozen already yields LongBench performance comparable to the dense controls. Stage 2 joint training further improves both models on LongBench and RULER, while keeping their LongBench scores close to those of the corresponding dense controls. These results suggest that Stage 1 establishes an effective sparse selector, while Stage 2 further improves the model's performance under sparse attention.

Selection Threshold p. We further study the selection threshold p, adjustable at inference time without retraining, and whether lowering it can reduce KL-only budgets. We sweep p from 0.50 to 0.99 on Llama-3.1-8B over LongBench v1, comparing CaliTopP with KL-only approach under the same training budget (Figure 5). For CaliTopP, the budget grows with p, while scores saturate at 52.19–52.46 for $p \geq 0.95$. At similar budgets (3110 vs. 3100 keys/query), CaliTopP at $p = 0.95$ scores 52.19, versus 50.45 for KL-only at $p = 0.85$. Since KL-only over-selection varies across tasks, uniformly lowering p is not a reliable remedy. For CaliTopP, a broad threshold range offers similarly favorable accuracy–compute trade-offs, so we fix one operating point per model in the main tables.

Model Scalability. To test our method at a larger scale, we apply it to the 671B DeepSeek-V3.2, whose released model was trained to convergence under a fixed top-k=2048 operator, so we freeze all backbone parameters and continue training only the indexer with the same KL anchor and $\mathcal{L}_{\text{bud}}$ at p=0.9 for 150 steps, then compare it against the released top-k selector (Table 10 and Table 11 , Appendix E). The capability of the model survives the substitution: LongBench is indistinguishable between the two arms, and on RULER, our method achieves comparable performance to the top-k under short-context scenarios and outperforms the top-k method for long contexts. Crucially, the calibrated selector achieves the above effect with much lower overhead for short contexts and can dynamically adjust the budget for long contexts. This shows that our top-p calibration scales to a 671B model, preserving quality while adapting the budget to each input.

6 Conclusion

We showed that token requirements vary across context length, layer, and task, motivating adaptive top-p selection in place of a fixed top-k budget. However, applying top-p to a KL-distilled indexer can lead to substantial budget inflation, as demonstrated by our theoretical construction and observed empirically in the DSA indexer. We therefore proposed CaliTopP, which adds a zero-parameter calibration term $\mathcal{L}_{\text{bud}}$ to explicitly align the selected-token budget with a teacher-derived target, yielding a native top-p indexer with identical training and inference operators and a single quality–compute dial. Across four benchmarks and four backbones, CaliTopP maintains competitive generation quality while retaining only a fraction of the context.

AI USE STATEMENT

In this work, we used generative AI tools (large language models) to assist in writing proofs, implementing our research methods in code, helping formulate mathematical claims, helping provide ingredients for proving mathematical claims, and cleaning and reformatting datasets. We did not use generative AI tools to develop theoretical models or conceptual frameworks, propose or refine hypotheses, design or provide feedback on research methodology or experiments, assist with translation, support qualitative or thematic data analysis, or interpret results. The use of generative AI for synthetic dataset generation is not applicable to this work. Additionally, we used generative AI tools to edit the manuscript to improve its readability.

We have reviewed all AI-assisted work: we checked the proofs, tested the implementation code, inspected the results of dataset cleaning and reformatting, and reviewed the manuscript edits. We take responsibility for the final content of this work, including text, mathematical claims, code, and data artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To facilitate reproducibility, we have submitted our training and inference code as supplementary material. The theoretical derivations and proofs are provided in Appendix A. The datasets used for training are described in Appendix C.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: training generalized multi-query transformer models from multi-head checkpoints. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 4895–4901. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.298. URL https://doi.org/10.18653/v1/2023.emnlp-main.298.

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. Longalign: A recipe for long context alignment of large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, volume EMNLP 2024 of *Findings of ACL*, pp. 1376–1395. Association for Computational Linguistics, 2024a. doi: 10.18653/V1/2024.FINDINGS-EMNLP.74. URL https://doi.org/10.18653/v1/2024.findings-emnlp.74.

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 *ACL (1)*, pp. 3119–3137. Association for Computational Linguistics, 2024b.

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *ACL (1)*, pp. 3639–3664. Association for Computational Linguistics, 2025.

Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. Indexcache: Accelerating sparse attention via cross-layer index reuse. *CoRR*, abs/2603.12201, 2026. doi: 10.48550/ARXIV.2603.12201. URL https://doi.org/10.48550/arXiv.2603.12201.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic KV cache compression based on pyramidal information funneling. *CoRR*, abs/2406.02069, 2024.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Léon Bottou, Zhihao Jia, and Beidi Chen. Magicpig: LSH sampling for efficient LLM generation. In *ICLR*. OpenReview.net, 2025.

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

DeepSeek-AI. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model, 2024.

DeepSeek-AI. Deepseek-v3.2: Pushing the frontier of open large language models. *CoRR*, abs/2512.02556, 2025.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026.

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 *ICLR*. OpenReview.net, 2024.

GLM. GLM-5: from vibe coding to agentic engineering. *CoRR*, abs/2602.15763, 2026.

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? *CoRR*, abs/2404.06654, 2024.

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 ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Xunhao Lai, Weiqi Xu, Yufeng Yang, Qiaorui Chen, Yang Xu, Lunbin Zeng, Xiaolong Li, Haohai Sun, Haichao Zhu, Vito Zhang, Jinkai Hu, Jiayao Li, Rui Gao, Zekun Li, Songquan Zhu, Jingkai Zhou, and Pengyu Zhao. Minimax sparse attention. *CoRR*, abs/2606.13392, 2026. doi: 10.48550/ARXIV.2606.13392. URL https://doi.org/10.48550/arXiv.2606.13392.

Wenhao Li, Jinhao Dong, Hailin Zhang, Wenhang Shi, Wei Lu, and Xiaoyong Du. Rabitqcache: Rotated binary quantization for kvcache in long context LLM inference. *CoRR*, abs/2606.31519, 2026.

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 *NeurIPS*, 2024.

Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. In *NeurIPS*, 2025.

Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa: Surpassing gpt-4 on conversational qa and rag. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 15416–15459. Curran Associates, Inc., 2024. doi: 10.52202/079017-0493. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/1c0d54ebd0a6e58c4eca7d591e374b9d-Paper-Conference.pdf.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Hongyi Jin, Tianqi Chen, and Zhihao Jia. Towards efficient generative large language model serving: A survey from algorithms to systems. *ACM Comput. Surv.*, 58(1):15:1–15:37, 2026.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023.

Zihan Qiu, Zekun Wang, Xiao Li, Yanpeng Li, Yang Xu, Yixuan Wang, Huaqing Zhang, Rui Men, Bochao Mao, Chengruidong Zhang, Fan Zhou, Hao Luo, Haofeng Huang, Haoran Lian, Haoyan Huang, Hongqing Chen, Jianwei Zhang, Jing Xu, Junjie Wang, Langshi Chen, Liangyu Wang, Linlang Jiang, Man Yuan, Minmin Sun, Peng Jin, Siqi Zhang, Siyu Wang, Xingzhang Ren, Yakai Wang, Yi Zhang, Yiming Dong, Yizhong Cao, Yubo Ma, Yunfei Mao, Bo Zheng, and Dayiheng Liu. On the design of qwen3.8-next architecture: Evaluation, efficiency, and training stability. *CoRR*, abs/2608.30320, 2026.

Qwen Team. Qwen3.8-Max: A new bar for coding and cowork, August 2026. URL https://qwen.ai/blog?id=qwen3.8.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient LLM inference. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 42558–42583. PMLR / OpenReview.net, 2024.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. QUEST: query-aware sparsity for efficient long-context LLM inference. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 47901–47911. PMLR / OpenReview.net, 2024.

Kimi Team. Kimi K3: open frontier intelligence. *CoRR*, abs/2607.24653, 2026.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pp. 5998–6008, 2017.

Boxin Wang, Chankyu Lee, Nayeon Lee, Sheng-Chieh Lin, Wenliang Dai, Yang Chen, Yangyi Chen, Zhuolin Yang, Zihan Liu, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nemotron-cascade: Scaling cascaded reinforcement learning for general-purpose reasoning models. *CoRR*, abs/2512.13607, 2025. doi: 10.48550/ARXIV.2512.13607. URL https://doi.org/10.48550/arXiv.2512.13607.

Huaijin Wu, Lianqiang Li, Hantao Huang, Tu Yi, Jihang Zhang, Minghui Yu, and Junchi Yan. Hshare: Fast LLM decoding by hierarchical key-value sharing. In *ICLR*. OpenReview.net, 2025.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=NG7sS51zVF.

Shuo Yang, Ying Sheng, Joseph E. Gonzalez, Ion Stoica, and Lianmin Zheng. Post-training sparse attention with double sparsity. *CoRR*, abs/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. *Proc. ACM Manag. Data*, 3(3):201:1–201:30, 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.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate LLM serving. In *MLSys*. mlsys.org, 2024a.

Yuze Zhao, Jintao Huang, Jinghan Hu, Xingjun Wang, Yunlin Mao, Daoze Zhang, Zeyinzi Jiang, Zhikai Wu, Baole Ai, Ang Wang, Wenmeng Zhou, and Yingda Chen. Swift:a scalable lightweight infrastructure for fine-tuning, 2024b. URL https://arxiv.org/abs/2408.05517.

Ruijie Zhou, Fanxu Meng, Yufei Xu, Tongxuan Liu, Guangming Lu, Muhan Zhang, and Wenjie Pei. MISA: mixture of indexer sparse attention for long-context LLM inference. *CoRR*, abs/2605.07363, 2026a.

Yanke Zhou, Yiduo Li, Hanlin Tang, Maohua Li, Kan Liu, Tao Lan, Lin Qu, Yuan Yao, and Xiaoxing Ma. Full attention strikes back: Transferring full attention into sparse within hundred training steps. *arXiv preprint arXiv:2605.16928*, 2026b.

A PROOFS

We formalize a failure mode of forward-KL distillation under limited representational flexibility. Specifically, we construct a one-parameter softmax indexer whose unique global KL optimum preserves the teacher's token ordering and achieves arbitrarily small distillation error, yet incurs arbitrarily large top-p budget inflation. The construction isolates probability calibration from ranking errors and optimization failures.

We fix a single query, a threshold $p \in (0,1)$, and the candidate set $[n] = \{1, \dots, n\}$. All logarithms are natural. For a distribution v, write $v(H) = \sum_{i \in H} v_i$ and define its top-p budget by

$$N_p(v) = \min \left\{ k \in \{1, \dots, n\} : \sum_{j=1}^{k} v_{(j)} \geq p \right\},$$

where $v_{(1)} \geq \cdots \geq v_{(n)}$ is the descending ordering, with ties broken by token index. This is the minimum cardinality in Eq. 1.

A.1 AMPLIFICATION BY A DIFFUSE TAIL

Lemma 1 (Tail amplification)**.** *Let a and q be distributions on $[n]$, and let H contain the r highest-probability tokens under q, where $1 \leq r < n$. Suppose that, for some $\delta > 0$ and $b > 0$,*

$$a(H) \geq p, \qquad q(H) = p - \delta, \qquad \max_{i \notin H} q_i \leq b.$$

Then

$$N_p(a) \leq r, \qquad N_p(q) \geq r + \left\lceil \frac{\delta}{b} \right\rceil. \tag{8}$$

If q is uniform on the $n - r$ tail tokens, one can take $b = (1 - p + \delta)/(n - r)$, giving

$$N_p(q) = r + \left\lceil \frac{\delta(n - r)}{1 - p + \delta} \right\rceil. \tag{9}$$

Proof. The r largest teacher probabilities have total mass at least $a(H) \geq p$, so $N_p(a) \leq r$. Under the proxy ordering, the first r tokens are precisely H and leave a mass deficit of δ. Each additional token supplies at most b, so at least $\lceil \delta/b \rceil$ more tokens are necessary. When the tail is uniform, each supplies exactly $(1 - p + \delta)/(n - r)$, which gives equality. □

The lemma identifies the amplification mechanism: a small underestimate of head mass can require many additional tokens when the missing mass must be recovered from a long, diffuse tail.

A.2 A ONE-PARAMETER KL-OPTIMAL CONSTRUCTION

Fix integers $3 \leq r < n$ and $0 < \delta < \min\{p/2, 1 - p\}$. Let $H = \{1, \dots, r\}$ and define three group masses

$$h = \frac{p}{2}, \qquad m = \frac{p}{2} - \delta, \qquad t = 1 - p + \delta.$$

The reference proxy has one high peak, $r - 1$ lower peaks, and $n - r$ background tokens:

$$v_i = \begin{cases} h, & i = 1, \\ m/(r-1), & 2 \leq i \leq r, \\ t/(n-r), & r < i \leq n. \end{cases} \tag{10}$$

The indices only label the groups; their positions in the original sequence can be arbitrary. Choose the parameters so that $h > m/(r-1) > t/(n-r)$, and define the positive log gaps and a mass transfer by

$$g_1 = \log \frac{h}{m/(r-1)}, \qquad g_2 = \log \frac{m/(r-1)}{t/(n-r)}, \qquad u = \frac{2\delta g_2}{g_1}.$$

Our indexer has a single trainable scale:

$$q_\theta(i) = \frac{\exp(\theta s_i)}{Z(\theta)}, \qquad s_i = \log v_i, \qquad Z(\theta) = \sum_{j=1}^{n} \exp(\theta s_j), \qquad \theta > 0. \tag{11}$$

Since $\sum_i v_i = 1$, $q_1 = v$. Changing θ preserves the ordering and multiplies both log gaps by the same factor; it cannot adjust the two contrasts independently. There is no upper bound on θ.

Define the teacher by

$$a_i = \begin{cases} h - u, & i = 1, \\ (m + u + 2\delta)/(r-1), & 2 \le i \le r, \\ (t - 2\delta)/(n-r), & r < i \le n. \end{cases} \tag{12}$$

We additionally require

$$h - u > \frac{m + u + 2\delta}{r-1} > \frac{t - 2\delta}{n-r} > 0. \tag{13}$$

These conditions ensure full support and identical ordering of the three groups for a and v. Both distributions sum to one, and

$$a(H) = p + \delta, \qquad v(H) = p - \delta.$$

Relative to v, the teacher lowers the top-to-middle probability ratio while increasing the middle-to-background ratio, so no common scale can reproduce both. The following proposition also establishes that the required parameter conditions are feasible.

Proposition 1 (Small KL with unbounded budget inflation)**.** *For the construction above, the following hold.*

(i) The unique global minimizer of $\mathrm{KL}(a \,\|\, q_\theta)$ *over* $\theta > 0$ *is* $\theta^\star = 1$.

(ii) The teacher and the KL-optimal proxy have identical token orderings, including ties within each group, but

$$N_p(a) \le r, \qquad N_p(q_{\theta^\star}) = r + \left\lceil \frac{\delta(n-r)}{1 - p + \delta} \right\rceil,$$

and hence

$$\frac{N_p(q_{\theta^\star})}{N_p(a)} \ge 1 + \frac{\delta(n-r)}{r(1-p+\delta)}. \tag{14}$$

(iii) For every fixed $p \in (0,1)$*, setting*

$$r = \lceil \sqrt{n} \rceil, \qquad \delta = n^{-1/4}$$

makes the construction valid for all sufficiently large n*, with*

$$\mathrm{KL}(a \,\|\, q_{\theta^\star}) = O(n^{-1/2}) \longrightarrow 0, \qquad \frac{N_p(q_{\theta^\star})}{N_p(a)} = \Omega(n^{1/4}) \longrightarrow \infty. \tag{15}$$

The asymptotic constants may depend on p.

Proof. **(i) Global KL optimality.** Let $F(\theta) = \mathrm{KL}(a \,\|\, q_\theta)$. Expanding and differentiating the finite sum gives

$$F(\theta) = \sum_i a_i \log a_i - \theta\, \mathbb{E}_a[s] + \log Z(\theta),$$
$$F'(\theta) = \mathbb{E}_{q_\theta}[s] - \mathbb{E}_a[s], \qquad F''(\theta) = \mathrm{Var}_{q_\theta}(s) > 0.$$

The variance is strictly positive because q_θ has full support and the three score levels are distinct. Thus F is strictly convex. By construction,

$$\begin{aligned}\mathbb{E}_a[s] - \mathbb{E}_v[s] &= -u \log h + (u + 2\delta) \log \frac{m}{r-1} - 2\delta \log \frac{t}{n-r} \\ &= -u g_1 + 2\delta g_2 = 0.\end{aligned}$$

Since $q_1 = v$, we have $F'(1) = 0$. Strict convexity makes $\theta^\star = 1$ the unique global minimizer on $(0, \infty)$. The head-mass underestimate therefore persists at the global KL optimum.

(ii) Budget inflation without ranking errors. The strict inequalities defining v and Eq. 13 give identical group orderings; both distributions are uniform within each group, so their ties also agree. The set H is the first r tokens in both orderings, with $a(H) = p + \delta$ and $v(H) = p - \delta$. Its complement is uniform under v, with per-token mass $t/(n-r)$. Lemma 1 gives the stated budgets. The ratio bound follows from $N_p(a) \le r$ and $\lceil x \rceil \ge x$.

(iii) Vanishing KL and diverging inflation. Uniformity within each group reduces the full KL exactly to the KL of the three group masses:

$$\begin{aligned}\mathrm{KL}(a \,\|\, v) = (h-u) \log \frac{h-u}{h} \\ + (m + u + 2\delta) \log \frac{m+u+2\delta}{m} + (t - 2\delta) \log \frac{t-2\delta}{t}.\end{aligned}$$

Using $\log x \le x - 1$ and normalization yields the chi-square bound

$$\mathrm{KL}(a \,\|\, v) \le \sum_i \frac{(a_i - v_i)^2}{v_i} = \frac{u^2}{h} + \frac{(u+2\delta)^2}{m} + \frac{4\delta^2}{t}. \tag{16}$$

Now fix p and take $r = \lceil \sqrt{n} \rceil$, $\delta = n^{-1/4}$. Then

$$\begin{aligned}g_1 &= \log(r-1) + \log(h/m) = \tfrac{1}{2} \log n + O(1), \\ g_2 &= \log \frac{n-r}{r-1} + \log(m/t) = \tfrac{1}{2} \log n + O(1).\end{aligned}$$

In particular, both gaps are positive for large n, $g_2/g_1 \to 1$, and $u = 2\delta(1 + o(1))$. The teacher's per-token levels satisfy

$$h - u \longrightarrow \frac{p}{2}, \qquad \frac{m+u+2\delta}{r-1} \sim \frac{p}{2} n^{-1/2}, \qquad \frac{t-2\delta}{n-r} \sim (1-p) n^{-1}.$$

All leading constants are positive, so Eq. 13 holds for sufficiently large n; the bounds on δ hold as well. Also, $h, m \to p/2$ and $t \to 1 - p$, so the denominators in Eq. 16 stay bounded away from zero. It follows that $\mathrm{KL}(a \,\|\, v) = O(\delta^2) = O(n^{-1/2})$. Finally, $(n-r)/r \sim \sqrt{n}$, so Eq. 14 gives $\Omega(n^{1/4})$ inflation. □

Consequently, for every $\varepsilon > 0$ and every finite $R > 0$, this construction contains a globally KL-optimal proxy with correct ordering, $\mathrm{KL}(a \,\|\, q_{\theta^\star}) \le \varepsilon$, and $N_p(q_{\theta^\star})/N_p(a) > R$. An arbitrarily small bound on forward KL therefore provides no finite bound on budget inflation that is uniform in context length for this class of indexers.

A.3 Peak-to-background contrast and existence of a suitable selector

Lower peak-to-background contrast. For a lower peak $2 \le i \le r$ and a background token $j > r$,

$$\frac{a_i / a_j}{v_i / v_j} = \frac{1 + (u + 2\delta)/m}{1 - 2\delta/t} > 1.$$

For the highest peak,

$$\frac{a_1 / a_j}{v_1 / v_j} = \frac{1 - u/h}{1 - 2\delta/t} > 1 \quad \Longleftrightarrow \quad \frac{g_2}{g_1} < \frac{h}{t}.$$

In the asymptotic construction, $g_2/g_1 \to 1$ while $h/t \to p/[2(1-p)]$. Thus, for $p > 2/3$, including $p = 0.9$ and $p = 0.95$, the KL-optimal proxy underestimates the peak-to-background contrast of every peak group for all sufficiently large n. This is a statement about relative contrast: the highest peak actually has $a_1 = h - u < v_1$.

Existence of a suitable selector. We now show that the same indexer family contains a selector with a smaller budget and sufficient teacher-mass coverage. Every head score exceeds the common tail score $\log(t/(n-r))$. Differentiating the head mass gives

$$\frac{d}{d\theta} q_\theta(H) = q_\theta(H)\big(1 - q_\theta(H)\big)\left(\mathbb{E}_{q_\theta}[s \mid H] - \log \frac{t}{n-r}\right) > 0.$$

Moreover, $q_1(H) = p - \delta < p$ and $\lim_{\theta\to\infty} q_\theta(H) = 1$. Continuity and strict monotonicity therefore give a unique $\theta_p > 1$ such that $q_{\theta_p}(H) = p$. At this value all r head tokens are necessary: every shorter prefix omits positive head mass and hence falls below p. Thus

$$N_p(q_{\theta_p}) = r, \qquad \mathcal{I}_p(q_{\theta_p}) = H, \qquad a\big(\mathcal{I}_p(q_{\theta_p})\big) = p + \delta > p.$$

The scale θ_p witnesses the existence of a smaller-budget selector with sufficient teacher-mass coverage in this single-query family. It neither establishes a common temperature for different queries nor characterizes the minimizer of the smooth loss in Eq. 2. Our trained indexer has no explicit temperature parameter; $\mathcal{L}_{\text{bud}}$ supervises the teacher-derived budget target n_t^* through its existing weights.

Fixed temperatures and query-dependent calibration. For the scorer in §4.1, let W_u denote the existing projection producing the head weights $u_{t,h}$. Replacing W_u by αW_u, for a fixed $\alpha > 0$, scales every score to $\alpha s_{t,i}$, exactly implementing inverse-temperature scaling. An explicit fixed temperature is therefore a redundant parameterization and can be folded into this matrix at inference. However, a global scale, or one scale per layer, provides only a shared correction across queries. Although the resulting top-p budgets still vary with the input, the scale cannot independently adjust each query's calibration error. A separate learned predictor could output query-dependent temperatures, but would introduce additional parameters and a calibration module. We instead apply budget supervision to the existing indexer, allowing its input-dependent scores to learn these corrections without adding such a module.

A.4 A FINITE-LENGTH THEORETICAL EXAMPLE

Take $n = 32768$, $r = 512$, $p = 0.9$, and $\delta = 0.005$. These parameters satisfy the ordering and positivity conditions above, as well as $g_2/g_1 < h/t$. Evaluating the construction and solving $q_{\theta_p}(H) = p$ gives the following values. This is a theoretical example, not experimental data.

Distribution or indexer q	$\mathrm{KL}(a \,\|\, q)$	$N_{0.9}(q)$
Teacher $q = a$	0	507
KL-optimal indexer, $\theta = 1$	0.000979	2048
Existence witness, $\theta_p \approx 1.006205$	0.001284	512

The KL values and θ_p are rounded for display; the witness budget is evaluated at the exact solution of $q_{\theta_p}(H) = p$. The KL optimum uses $2048/507 \approx 4.04$ times the teacher's budget despite correct ordering and a small distillation loss. The witness within the same family uses 512 tokens and covers teacher mass $a(H) = 0.905$. This comparison isolates the objective mismatch for one query; it is not an evaluation of a shared temperature-calibration method.

Algorithm 1 Top-p selection at inference (one layer, one query t)

Require: query state of token t; cached keys $\{k_i\}_{i \le t}$; threshold $p \in (0, 1]$; sink size n_s; window size w

1: $s_{t,i} \leftarrow \sum_h u_{t,h} \operatorname{ReLU}\big(q_{t,h}^\top k_i\big)$ for all $i \le t$ ▷ indexer scores
2: $\tilde{a}_t \leftarrow \operatorname{softmax}(s_t)$ over all $i \le t$ ▷ normalize over the full causal history
3: $\pi \leftarrow \operatorname{argsort}(\tilde{a}_t, \text{descending})$ ▷ break ties by token index
4: $c_j \leftarrow \sum_{j' < j} \tilde{a}_{t,\pi_{j'}}$ ▷ mass strictly before rank j
5: $r \leftarrow |\{j \in \{1, \dots, t\} : c_j < p\}|$ ▷ full-history prefix length
6: $\mathcal{P} \leftarrow \{\pi_1, \dots, \pi_r\}$ ▷ includes the token that reaches p
7: $\mathcal{T}_t \leftarrow \mathcal{S} \cup \mathcal{W}_t \cup \mathcal{P}$ ▷ sink and window are unconditional
8: **return** $o_t \leftarrow \operatorname{Attn}\big(q_t, K_{\mathcal{T}_t}, V_{\mathcal{T}_t}\big)$

Table 3: Indexer configuration per backbone.

Backbone	Host attention	Host (heads × dim)	Indexer (heads × dim)
DeepSeek-V2-Lite	MLA (MoE)	16×192	8×128
Llama-3.1-8B	dense GQA	32×128	16×64
Qwen3.8-27B	gated, hybrid	24×256	12×64
DeepSeek-V3.2	MLA (MoE)	128×192	64×128 (released)

B Implementation Details

Algorithm 1 spells out the selection operator of Eq. 4 as it runs at inference time, for one layer and one query; prefill applies it to every query position and decoding to the single new query, and training uses the identical procedure. The softmax and cumulative mass in lines 2–4 are computed over the *whole* causal history. The cut in line 5 gives the length r of the shortest global prefix reaching p, including the token that first reaches or exceeds the threshold. This prefix length is never capped by $|\mathcal{M}_t|$; the number of selected middle tokens is $|\mathcal{P} \cap \mathcal{M}_t|$, which may be zero. Sink and window tokens are added by union in line 7, giving exactly $\mathcal{T}_t = \mathcal{S} \cup \mathcal{W}_t \cup \mathcal{I}_p(\tilde{a}_t)$ and ensuring $\sum_{i \in \mathcal{T}_t} \tilde{a}_{t,i} \ge p$. The ReLU in the score makes exact ties common, so ties are broken by token index in line 3 and membership is decided by rank in line 6; this avoids admitting every token tied at the cut.

What changes across backbones is not this procedure but how the indexer is attached to the host attention, in four places. **(i) Its query source.** On MLA the indexer reads the compressed query latent the host already produces; on dense and gated attention there is no such latent, so it reads the layer's hidden states directly. **(ii) Its positional encoding.** The indexer carries its own rotary embedding over $\min(d_{\text{freqs}}, d_{\text{head}}^{\text{idx}})$ dimensions: on MLA this reproduces the released convention of rotating only the leading dimensions, while on a dense backbone it rotates the indexer's full head dimension with the host's own frequencies, including any length-scaling the host applies, so that the indexer and the model it imitates share one positional geometry. **(iii) Head grouping.** Under grouped-query attention the host packs keys and values per group, so they are expanded to per-head layout before scoring; the indexer's own key stays shared across its heads in every case. **(iv) Which layers get an indexer.** On hybrid backbones that interleave full attention with linear-attention layers, the indexer is attached only to the full-attention layers, the linear ones having no quadratic term to sparsify.

The indexer itself is sized relative to its host rather than copied between models. We refer to DeepSeek V3.2 to provide configuration data for each model. Table 3 lists the resulting configurations.

C Evaluation Protocol Details

Baseline configurations. Unless stated otherwise, each baseline uses the configuration recommended in its official repository: RaBitQCache (top-p=0.95), Quest (chunk size = 16), SparQ (r=16), DS (r=32, quant bit = 2, Query channel), and MagicPIG (K=10, L=150, W=64). For

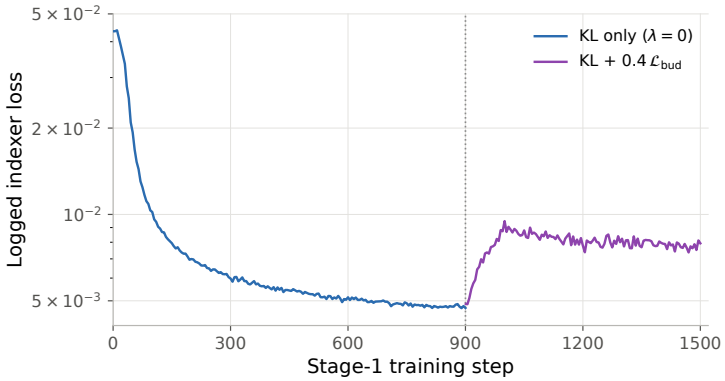


Figure 6: Stage-1 indexer training on DeepSeek-V2-Lite, with a logarithmic vertical axis. The KL-only warm-up (blue) runs through step 900 (dotted line), followed by 600 additional steps with $\mathcal{L}_{\text{bud}}$ ($\lambda = 0.4$, purple). Curves show the logged total indexer loss, including the training loss coefficient. The initial rise follows the ramp-up of the budget-loss weight. The averages over total training steps 1001–1050 and 1451–1500 are 8.72×10^{-3} and 7.74×10^{-3}, respectively.

DSA we set its Sink to 64 and Window to 64 to match our method. The reason we don't select RTPurbo as the baseline is that it keeps DSA's KL-distilled indexer and only swaps in a top-p cut, which coincides with our λ=0 setting; its remaining optimizations are orthogonal to our method and transfer directly, so we report it as an ablation of the calibration coefficient in §5.3 rather than as a separate system.

Our configuration. Stage 1 trains the indexer on 17,331 long documents drawn from Nemotron-SFT-Instruction-Following-Chat-v3 and Nemotron-SFT-SWE-v3 (Wang et al., 2025), the latter supplying the genuinely long contexts that long-range selectivity has to be learned from. Stage 2 uses 6,142 single-turn examples built from LongAlign-10k (Bai et al., 2024a), the ChatQA training sources (Liu et al., 2024) and the same Nemotron corpora, covering long-document question answering, multi-passage retrieval, summarization, in-context learning and code continuation. We confirm that the adopted training data does not contain any test data. Both stages run at a sequence length of 16K with sequence packing and left truncation, a global batch of 8 and a micro-batch of 1. Stage 1 keeps the backbone frozen and trains the indexer for 900 steps under the distillation term alone at a learning rate of 2×10^{-4}, then for 600 further steps with the calibration term engaged at λ=0.4 and a learning rate of 1×10^{-4}, both decaying to a tenth of their initial value. Stage 2 unfreezes the backbone for 500 joint steps at 2×10^{-6} and supervises only the final response. Every run fits on a single node of eight Hopper GPUs, and the two stages together cost about 130 GPU-hours

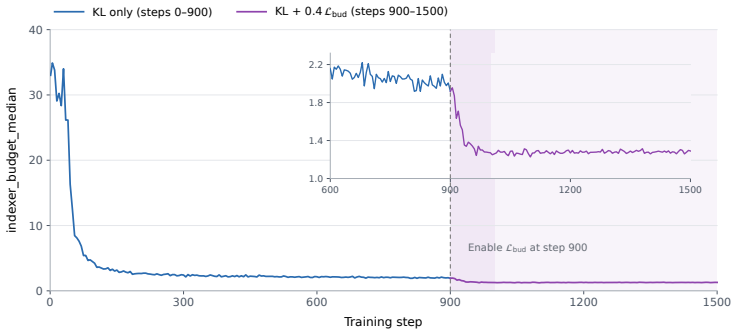


Figure 7: Budget calibration during Stage-1 indexer training on DeepSeek-V2-Lite. The vertical axis shows the median indexer-to-teacher ratio of selected middle-token counts, averaged over layers. KL-only training (blue) runs through step 900; the dashed line marks the start of 600 additional steps with $\mathcal{L}_{\text{bud}}$ ($\lambda = 0.4$, purple).

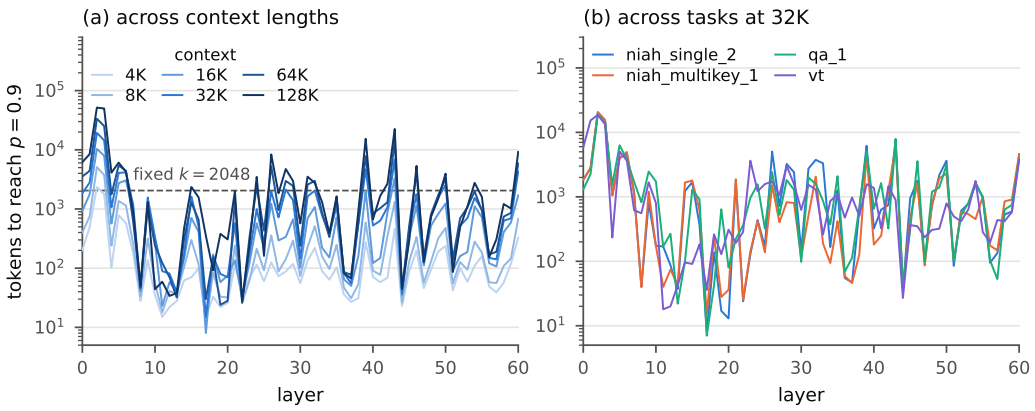


Figure 8: Per-layer top-p=0.9 requirement of DeepSeek-V3.2 — the minimal number of history tokens whose dense attention probabilities sum to 0.9, as a median over prompts. (a) the same 61 layers at six context lengths, against the fixed k=2048 of the released selector. (b) the same layers at 32K under the four RULER task families.

on DeepSeek-V2-Lite, 220 on Llama-3.1-8B and 180 on Qwen3.8-27B. The smooth count uses τ=0.02 *to* 0.005 in mass units during training, the row weights of Eq. 5 use ℓ_0=2048 and $\omega_{\max}$=8. At inference the operator uses p=0.95 on DeepSeek-V2-Lite, Qwen3.8-27B and Llama-3.1-8B and p=0.9 on DeepSeek-V3.2, with a sink of 64 tokens, a window of 64 tokens; the cut is taken on the distribution over the whole causal history, and decoding is greedy under each benchmark's official prompting protocol. For all models, the p used in training is the same as that used in final inference. The DSA training configuration is the same as ours. Figure 6 plots the logged total Stage-1 indexer loss for the KL-only warm-up and the $\lambda = 0.4$ calibration run starting from its step-900 checkpoint, documenting the calibration-weight warm-up and subsequent loss decrease and stabilization discussed in §3. Figure 7 shows the corresponding decrease in the indexer-to-teacher budget ratio after enabling $\mathcal{L}_{\text{bud}}$.

D Per-Layer and Per-Task Budget Variability

§3 shows that the token budget required to reach a given attention mass grows with context length. Here we report the two remaining axes of the same mismatch, measured on DeepSeek-V3.2 with the same protocol: for every query we take the dense attention of the first decoded token and count, per layer, the minimal number of history tokens whose attention probabilities sum to p=0.9. We sweep RULER prompts from 4K to 128K over four task families (`niah_single_2`, `niah_multikey_1`, `qa_1`, `vt`).

The requirement varies by orders of magnitude across layers. Within a single model and at one fixed context length, the per-layer requirement is anything but uniform (Figure 8a). At 4K the layer medians run from 15 to 2344 tokens, a 156× spread, and the spread widens with length: 70–4081 between the 10th and 90th percentile layer at 32K around a median layer of 693, and 28 to 51666 tokens at 128K, a 1845× spread. A single k shared by all layers therefore cannot be right for more than a few of them at once: the budget that serves the heavy layers is two orders of magnitude larger than the light layers need, and the budget that suits the light layers starves the heavy ones.

The requirement also shifts with the task. Fixing the length at 32K and comparing the four task families (Figure 8b), the median layer requirement moves from 514 tokens on `niah_multikey_1` to 868 on `qa_1`, and the per-layer picture moves considerably more: the same layer's requirement swings by a median of 5.0× across the four tasks, by 17.5× at the 90th percentile and by up to 37.7×, and 13 of the 61 layers swing beyond 10×. The identity of the budget-heavy layers is not fixed either. Layers 2, 3 and 43 are the heaviest on the three retrieval and QA tasks, whereas on variable tracking (`vt`) the demand concentrates entirely in the first handful of layers. A budget tuned on one task family is thus mis-allocated on another, which is what an input-adaptive rule such as top-p removes by construction.

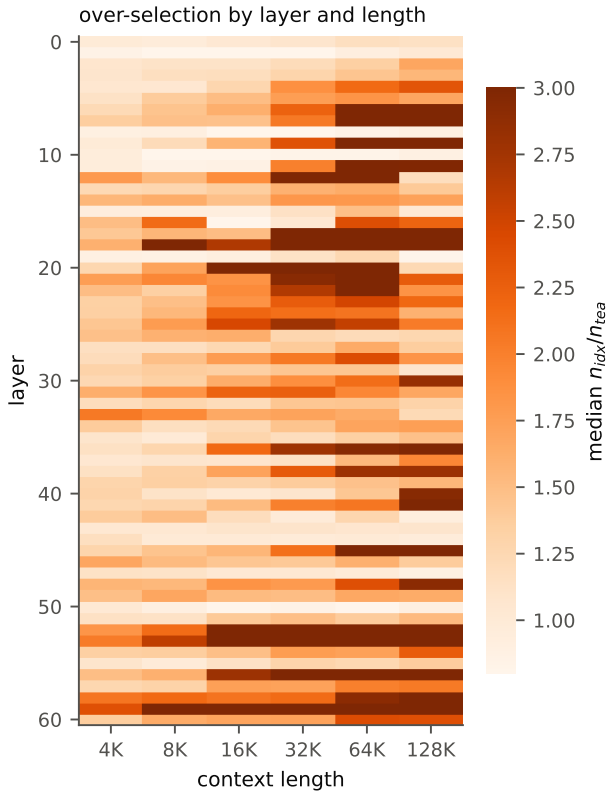


Figure 9: Median over-selection $n_{\text{idx}}/n_{\text{tea}}$ of the released DeepSeek-V3.2 indexer under top-p=0.9, per layer and context length.

Over-selection is likewise layer-dependent. The same per-layer view makes the over-selection of §3 more precise. Figure 9 maps the median ratio $n_{\text{idx}}/n_{\text{tea}}$ between the tokens the released indexer keeps under top-p=0.9 and the tokens the teacher distribution actually needs. The ratio is mild and nearly uniform at short context — median 1.26 at 4K with only 4 of 61 layers past $2\times$ — and both grows and disperses with length, reaching a median of 1.83 at 64K with 28 of 61 layers past $2\times$ and individual layers above $20\times$. Over-selection is therefore not a uniform offset that a global constant could absorb; it is concentrated in particular layers and lengths, which is why $\mathcal{L}_{\text{bud}}$ calibrates the cut point per query and per layer rather than rescaling the scores globally.

E Full Result Tables

For the accuracy evaluation in §5.1, Tables 4 and 5 provide per-subset LongBench v1 generation scores and retained-token budgets for DeepSeek-V2-Lite and Llama-3.1-8B, respectively. Table 6 reports the LongBench v2 results by difficulty and context length, including counts of unparsable answers, while Table 7 gives the complete GSM8K accuracy, attention recall, and retained-token budgets.

Table 8 details the $\mathcal{L}_{\text{bud}}$ ablation in §5.3, reporting per-subset scores and mean retained keys for $\lambda \in \{0, 0.1, 0.4, 1.0\}$. Figure 10 compares the percentage budget reductions from $\lambda = 0$ to $\lambda = 1$ at fixed p=0.95, showing how the budget changes vary across tasks.

Table 9 compares both training stages with their corresponding dense references on LongBench v1 and RULER, as summarized in §5.3. Table 10 reports aggregate scores and retained-key budgets for the frozen-backbone DeepSeek-V3.2 comparison in §5.3. Table 11 provides its LongBench v1 per-subset breakdown, pairing generation scores with the calibrated selector's retained-key budgets.

Table 4: LongBench v1 per-subset F1/ROUGE for DeepSeek-V2-Lite, broken out for every baseline and token budget of Table 1. In the score rows the AVG column is the unweighted mean over the 13 subsets; in the budget rows it is an average over queries and is therefore weighted by subset size.

Method	nrqa	qspr	mfqa	htpt	2wik	musq	govr	qmsm	mnws	trvq	psrt	lcc	rpbp	AVG
Full attention	21.5	33.2	47.4	25.5	20.2	11.9	33.2	23.8	25.7	84.2	78.0	63.4	64.9	40.99
Training-free selectors (baseline harness)														
Quest (K=1024)	16.7	27.5	44.0	19.5	19.3	11.5	27.9	21.2	26.2	70.7	61.6	50.7	54.5	34.72
Quest (K=2048)	18.9	32.1	45.2	21.7	18.2	11.1	29.8	22.2	26.2	75.5	62.5	55.4	58.5	36.71
Quest (K=4096)	20.0	33.6	47.4	22.8	19.4	11.3	30.8	23.1	26.3	82.1	68.5	57.3	59.6	38.63
DS (K=1024)	15.0	24.7	39.8	18.3	18.8	8.9	29.8	22.5	25.5	83.7	44.5	57.2	60.3	34.54
DS (K=2048)	19.4	28.4	43.8	19.3	20.1	8.8	30.8	23.1	26.2	83.1	60.5	57.8	62.0	37.19
DS (K=4096)	21.9	33.1	46.4	21.3	19.3	9.6	31.8	23.9	26.2	83.4	72.0	58.3	63.1	39.26
SparQ (ratio 25%)	22.7	33.0	46.2	24.6	19.3	11.3	33.2	23.9	26.0	84.6	75.5	59.2	64.2	40.28
MagicPIG (LSH)	22.0	28.2	42.9	20.9	20.9	11.4	30.6	23.9	25.0	83.6	64.5	58.0	61.3	37.94
RaBitQCache (p=0.95)	22.6	32.9	47.5	23.9	18.8	11.5	32.8	24.1	25.8	84.2	78.0	59.3	62.9	40.33
Trained selectors														
DSA (k=1024)	21.4	37.9	42.8	47.2	35.4	25.0	30.0	24.4	26.6	87.2	65.0	69.0	66.8	44.51
DSA (k=2048)	24.7	39.6	47.0	49.4	38.5	27.3	30.7	24.7	26.8	88.6	65.0	69.1	67.6	46.07
DSA (k=4096)	23.6	40.6	48.2	50.7	39.2	30.2	31.6	24.6	26.8	87.8	68.0	69.1	67.8	46.79
CaliTopP	23.7	36.6	47.4	49.9	36.4	29.0	31.7	25.3	26.5	88.8	65.5	68.5	67.4	45.90
Average retained tokens per query (budget)														
Quest (K=1024)	1022	1021	1021	1022	1022	1022	1022	1022	999	1022	1021	1022	1022	1020
Quest (K=2048)	2046	2044	2027	2045	2027	2045	2045	2045	1804	2043	2045	1998	2045	2020
Quest (K=4096)	4093	3783	3786	4066	3944	4093	4031	4086	2546	3961	4092	2957	4086	3755
DS (K=1024)	1024	1024	1024	1024	1024	1024	1024	1024	1001	1024	1024	1024	1024	1023
DS (K=2048)	2048	2048	2030	2048	2030	2048	2048	2048	1806	2046	2048	2001	2048	2023
DS (K=4096)	4096	3786	3789	4069	3947	4096	4034	4090	2547	3964	4096	2958	4090	3758
SparQ (ratio 25%)	6020	1324	1838	3462	1936	4195	2695	3669	758	3174	3277	978	3361	2713
MagicPIG (LSH)	4085	892	1243	2274	1297	2738	1791	2522	561	2080	2117	722	2357	1840
RaBitQCache (p=0.95)	7113	2123	2812	4763	2798	5640	3953	4993	1290	3848	4157	1572	4172	3628
DSA (k=1024)	1104	1004	1026	1093	1040	1108	1065	1093	811	1071	1101	918	1080	1032
DSA (k=2048)	2176	2176	2151	2175	2154	2176	2176	2176	1820	2172	2176	2098	2176	2138
DSA (k=4096)	4224	3867	3885	4194	4062	4224	4138	4217	2457	4077	4224	2973	4214	3845
CaliTopP	3874	1471	2101	3463	2165	3934	2787	3377	901	2706	3322	1028	2530	2442

F TIME AND SPACE COMPLEXITY

Write L for the context length, $h \times d$ for the host attention geometry and $h_i \times d_i$ for the indexer's (Table 3), and $\bar{n}$ for the mean size of the kept set $\mathcal{T}_t$. Per layer a forward pass splits into indexer scoring, the top-p cut, and attention on the kept tokens.

Time. Scoring is a second attention-shaped product over the indexer's smaller geometry: $O(L^2 h_i d_i)$ at prefill and $O(L h_i d_i)$ per decode step, a fraction $h_i d_i/(hd)$ of the host's $QK^\top$ that ranges from $1/8$ to $1/3$ over the four backbones of Table 3. Attention itself costs $O(L\bar{n}hd)$, which is nearly linear since $\bar{n}$ varies only mildly with the context.

Space. All indexer heads share one key vector per token, so the indexer adds d_i floats per token per layer to the KV cache. Nothing of size L^2 is materialized — scores are consumed in head and query-block chunks, so selection peaks at $O(Lch_i)$ for a query block of c rows, and the kept sets are carried as a padded index tensor of $O(L \max_t n_t)$.

Table 5: LongBench v1 per-subset F1/ROUGE for Llama-3.1-8B, broken out for every baseline and token budget of Table 1. Same conventions as Table 4.

Method	nrqa	qspr	mfqa	htpt	2wik	musq	govr	qmsm	mnws	trvq	psrt	lcc	rpbp	AVG
Full attention	23.8	45.3	55.1	54.1	43.8	31.8	34.9	25.1	27.4	91.5	99.5	65.0	57.9	50.40
Training-free selectors (baseline harness)														
Quest (K=1024)	24.7	40.0	49.7	45.5	38.1	20.3	33.6	23.7	27.6	80.5	99.5	39.1	47.6	43.84
Quest (K=2048)	27.7	45.2	51.7	52.6	38.6	27.0	34.2	24.7	27.4	87.2	99.5	41.1	48.9	46.59
Quest (K=4096)	28.8	46.1	50.9	52.1	36.5	24.1	35.0	25.2	27.4	89.8	99.5	42.3	49.5	46.70
DS (K=1024)	28.2	45.5	52.3	55.2	41.8	25.5	34.4	24.6	27.2	91.5	99.5	45.5	53.5	48.05
DS (K=2048)	28.4	45.6	53.1	54.5	40.8	26.3	34.3	25.3	27.2	91.7	99.5	43.3	51.4	47.80
DS (K=4096)	30.0	45.4	53.2	55.4	41.9	25.5	34.2	25.0	27.3	91.0	99.5	42.9	51.1	47.87
SparQ (ratio 25%)	30.1	44.3	53.6	56.2	38.1	25.0	34.3	25.0	27.2	91.2	99.5	45.4	50.6	47.73
MagicPIG (LSH)	28.9	42.0	50.2	55.0	39.0	26.2	33.3	24.6	26.8	89.3	99.0	41.4	46.7	46.35
RaBitQCache (p=0.95)	30.1	46.1	53.0	55.5	40.8	28.5	35.0	25.2	27.3	91.0	99.5	41.8	49.2	47.93
Trained selectors														
DSA (k=1024)	19.8	35.1	33.7	46.9	44.9	23.0	27.1	22.5	26.1	87.4	18.0	71.2	64.9	40.05
DSA (k=2048)	22.5	41.3	43.8	50.6	48.1	25.2	30.5	24.1	27.2	88.2	46.0	72.0	66.3	45.08
DSA (k=4096)	23.1	45.1	52.1	55.6	48.6	28.1	33.7	24.4	27.3	89.7	72.5	72.4	68.1	49.29
CaliTopP	25.0	39.7	53.9	57.9	46.6	31.4	36.2	25.4	28.2	91.8	99.5	72.7	70.3	52.19
Average retained tokens per query (budget)														
Quest (K=1024)	1022	1021	1021	1021	1021	1021	1022	1022	995	1021	1021	1021	1022	1020
Quest (K=2048)	2045	2044	2019	2043	2021	2045	2045	2045	1775	2040	2044	1903	2045	2002
Quest (K=4096)	4092	3741	3726	4060	3918	4092	4019	4085	2454	3935	4091	2585	4030	3671
DS (K=1024)	1024	1024	1024	1024	1024	1024	1024	1024	998	1024	1024	1024	1024	1022
DS (K=2048)	2048	2048	2022	2047	2025	2048	2048	2048	1777	2044	2048	1906	2048	2005
DS (K=4096)	4096	3744	3729	4064	3923	4096	4023	4089	2455	3939	4096	2586	4033	3674
SparQ (ratio 25%)	5959	1276	1738	3215	1794	3906	2572	3494	714	2947	3132	803	2701	2481
MagicPIG (LSH)	3924	896	1197	2132	1227	2572	1756	2392	549	1943	2085	619	1923	1697
RaBitQCache (p=0.95)	3763	1357	1639	2326	1563	2686	2278	2818	934	2163	1944	919	1971	1923
DSA (k=1024)	1103	999	1018	1090	1031	1108	1061	1091	790	1066	1099	865	1063	1017
DSA (k=2048)	2176	2175	2144	2174	2149	2176	2176	2176	1771	2169	2176	1966	2176	2113
DSA (k=4096)	4224	3830	3830	4190	4038	4224	4123	4216	2329	4050	4224	2584	4144	3755
CaliTopP	5172	1887	2559	4368	2522	5224	3432	4188	983	3530	4094	1189	3476	3110

Table 6: LongBench v2 full split-level accuracy (%) on Qwen3.8-27B (503 questions; four-way multiple choice). *Null* counts the questions for which the model produced no parsable option; the original-dense and S1 rows are depressed by these format-following failures. *Dense twin* is the same Stage-2 recipe with full attention.

Method	Overall	Easy	Hard	Short	Medium	Long	Null
Original dense	29.0	31.3	27.7	25.0	28.8	36.1	173
CaliTopP (S1)	26.4	27.1	26.1	24.4	24.7	33.3	172
CaliTopP (S2)	54.1	57.3	52.1	61.7	49.3	50.9	1
Dense twin	56.7	60.4	54.3	60.0	54.4	55.6	1

Table 7: Detailed GSM8K results. Scores are formatted as accuracy (attention recall); *Tokens* is the average number of key tokens retained per decode step.

		DeepSeek-V2-Lite		Llama-3.1-8B	
Method	Budget	Acc	Tokens	Acc	Tokens
Full attention	full	69.7 (100%)	848	77.1 (100%)	771
Quest	256	36.0 (60%)	254	60.0 (61%)	253
DS	256	57.8 (50%)	256	73.3 (72%)	256
SparQ	256	68.2 (78%)	256	77.6 (86%)	256
MagicPIG	LSH	58.0 (70%)	193	66.3 (73%)	186
RaBitQCache	$p = 0.95$	70.8 (93%)	383	76.1 (89%)	302
DSA	$k = 256$	69.5 (92%)	384	71.6 (90%)	384
CaliTopP (S1)	$p = 0.95$	**69.7 (90%)**	301	**78.0 (94%)**	436
CaliTopP (S2)	$p = 0.95$	62.5 (89%)	292	76.4 (93%)	402
Dense twin	full	61.8 (100%)	843	76.5 (100%)	766

Table 8: Ablation of $\mathcal{L}_{\text{bud}}$ on DeepSeek-V2-Lite (Stage 1).

	$\lambda = 0$		$\lambda = 0.1$		$\lambda = 0.4$		$\lambda = 1.0$	
Dataset	Score	Keys/query	Score	Keys/query	Score	Keys/query	Score	Keys/query
NarrativeQA	21.93	3916	22.36	3830	21.75	3626	21.93	3439
Qasper	31.93	1418	32.25	1433	34.83	1409	32.86	1375
MultiFieldQA-en	47.50	2051	46.84	2013	47.19	1929	46.23	1845
HotpotQA	25.11	3244	25.04	3020	26.76	2763	24.80	2558
2WikiMQA	18.61	2020	20.92	1901	18.98	1763	19.57	1649
MuSiQue	12.48	3688	12.46	3390	12.06	3068	11.85	2820
GovReport	32.81	2668	32.81	2665	32.68	2570	32.85	2468
QMSum	23.77	3337	23.69	3325	23.60	3175	23.43	3021
MultiNews	25.56	863	25.53	854	25.83	830	25.56	805
TriviaQA	85.09	2589	85.01	2300	85.17	2028	84.85	1827
Passage Retrieval-en	78.50	3120	77.50	2796	76.50	2480	77.00	2247
LCC	62.67	1068	63.44	1050	63.82	1019	63.37	985
RepoBench-P	65.68	2702	65.04	2597	65.27	2444	65.14	2305
AVG	40.90	2514	40.99	2398	41.11	2239	40.73	2103

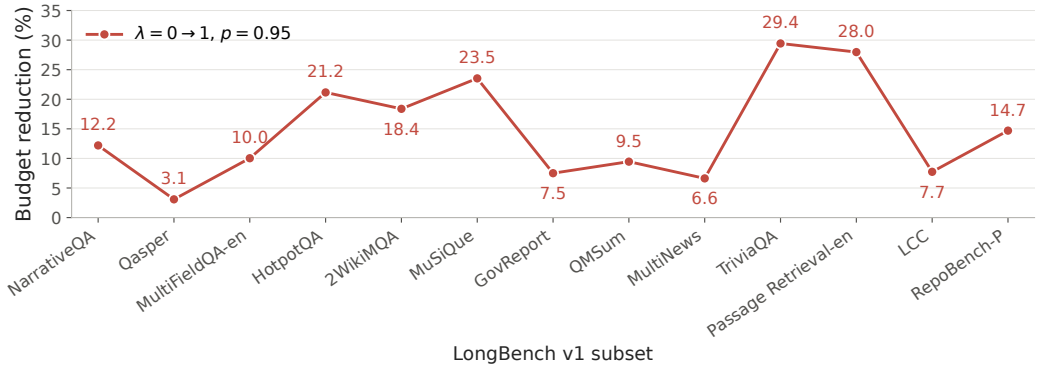


Figure 10: Per-task budget reduction from $\lambda = 0$ (KL only) to $\lambda = 1$ on DeepSeek-V2-Lite (Stage 1), with p=0.95 fixed. Each point reports $100(1 - \bar{n}_{\lambda=1}/\bar{n}_{\lambda=0})$, where $\bar{n}_\lambda$ is the mean number of retained keys per prefill query on that subset (Table 8). Reductions range from 3.1% to 29.4% across the 13 LongBench v1 subsets.

Table 9: Two-stage training ablation. LongBench and RULER use the same evaluation subsets and score aggregation as Table 1. The sparse rows use the default p=0.95.

Model	Configuration	LongBench	RULER
DeepSeek-V2-Lite	S1 (sparse)	41.11	74.80
	S1 dense	40.99	76.83
	S2 (sparse)	45.90	78.70
	S2 dense	45.69	80.03
Llama-3.1-8B	S1 (sparse)	50.17	79.63
	S1 dense	50.40	79.93
	S2 (sparse)	52.19	86.13
	S2 dense	52.51	87.40

Table 10: Frozen-backbone calibration on DeepSeek-V3.2: released top-k=2048 vs. CaliTopP (p=0.9). Each cell is score / mean retained keys per query.

	Original (k=2048)	CaliTopP (p=0.9)
RULER 4K	**96.40** / 2048	**96.40** / 287
RULER 8K	95.33 / 2048	**96.53** / 514
RULER 16K	**96.80** / 2048	96.40 / 1267
RULER 32K	94.80 / 2048	**95.37** / 1953
RULER 64K	93.77 / 2048	**94.27** / 2639
RULER 128K	89.37 / 2048	**92.20** / 3712
LongBench	56.24 / 2048	**56.26** / 1180

Table 11: LongBench v1 per-subset scores for DeepSeek-V3.2 with the backbone frozen, with the retained-key budget of the calibrated selector in the last block. Column abbreviations follow Table 4; AVG is the 13-subset average reported in §5.3. The released selector retains k=2048 keys per query on every subset by construction.

Method	nrqa	qspr	mfqa	htpt	2wik	musq	govr	qmsm	mnws	trvq	psrt	lcc	rpbp	AVG
DeepSeek-V3.2 (top-k=2048)	34.6	50.0	53.6	68.4	70.1	55.6	28.0	22.4	22.6	92.5	89.5	73.5	70.2	56.24
CaliTopP (p=0.9, frozen)	34.4	50.3	53.8	69.2	67.5	56.0	28.4	22.7	22.5	91.0	92.2	74.8	68.6	56.26
Retained keys per query														
CaliTopP (p=0.9, frozen)	1421	611	1142	1518	803	1633	1561	1895	332	989	1038	656	1737	1180