SpotAttention: Plug-In Block-Sparse Routing for Pretrained Long-Context Transformers

Anonymous ACL submission

Abstract

Long contexts have become standard in pretrained LLMs, yet they remain expensive to run: prefill compute grows quadratically with sequence length, and every decode step re-reads a key-value cache that grows linearly with it. Sparse attention cuts these costs by attending only to a relevant subset of past tokens, but selecting that subset is itself expensive. We present SpotAttention, a lightweight selector that attaches to a frozen pretrained transformer and learns by KL distillation to estimate its attention distribution. The selector picks the top-K keys each query attends to, and because its estimate is a calibrated distribution, a dual top-p rule reads the per-query, per-layer budget directly from it. Across Qwen3 (dense, 4B–32B) and Qwen3.5 (hybrid linear/full attention, 4B–9B), SpotAttention matches dense accuracy at contexts up to 128K tokens, eight times the training length. Decode at $L = 128$K runs $3.9\times$ faster than FlashAttention and $1.8\times$ faster than Twilight, the strongest training-free baseline. Quantizing the selector's K-cache to INT4 or FP4 microscale shrinks it $3.5\times$ at no accuracy cost.

1 Introduction

Long context has become one of the defining capabilities of modern large language models, letting them follow long documents, work across entire codebases, and ground their answers in large amounts of evidence. The same trend holds for multimodal models, where images and especially video expand into long token sequences. In just a few years, context windows have grown from a few thousand tokens to more than a million (Gemini Team, 2024), and they continue to grow.

Self-attention makes long context expensive (Vaswani et al., 2017). At prefill, its compute grows quadratically with sequence length; at decode, every new token re-reads the entire KV cache, whose size grows linearly with the context and soon dominates memory. FlashAttention (Dao et al., 2022) made attention far more IO-efficient and removed the need to materialize the full attention matrix in memory, but it changed neither the quadratic prefill nor the growing per-step read. Attention therefore dominates both latency and memory at long context, and serving these models becomes slow and costly.

Figure 1: **Cost of sparse attention.** Per-step decoding cost decomposed into *selector* and *attention*. Dense pays for full attention with no selector. Twilight cuts the attention bar but pays a heavy selector. SpotAttention keeps the same lighter attention with a 3x cheaper learned selector.

Recently, a growing body of work has shown that attention in trained LLMs is highly redundant: each query attends to only a small fraction of past tokens (Liu et al., 2023; Xiao et al., 2024b). Sparse-attention methods exploit that redundancy by attending to a selected subset of tokens, cutting the per-step read from the KV cache. Tang et al. (2024), for example, scores 16-token pages with min/max key bounds, and Lin et al. (2025) adds top-p pruning on top of that selector. These methods do reduce the attention cost, but selecting which tokens to attend to is itself expensive, and the selector becomes the new bottleneck. DeepSeek Sparse Attention (DSA) (DeepSeek-AI, 2025) answers it with the *lightning indexer*: a small module trained jointly with the backbone that estimates each layer's attention distribution, so the layer at-

tends only to its top-K keys.

In this paper, we propose SpotAttention, a learned selector that retrofits onto an already-trained backbone. We adapt the lightning-indexer architecture from DSA: the selector is trained alone via a KL loss against the dense attention, and scores blocks of queries and keys rather than individual tokens, so selection runs as a single tensor-core matmul. We introduce a dual top-p rule that reads the per-query, per-layer budget directly from the selector's estimated attention distribution, with sink and recency blocks reserved so they do not dominate the nucleus. Twilight (Lin et al., 2025) also adapts the budget per query, but as a separate top-p stage layered onto Quest's selector, paying the selector cost twice, while SpotAttention's nucleus is read from the distribution the selector already produces. Figure 1 previews the cost gap. At $L = 128$K decode on Qwen3-8B, SpotAttention reaches $3.9\times$ the throughput of FlashAttention and $1.8\times$ that of Twilight (Section 4.2); INT4 or FP4 microscale quantization on the selector's K-cache shrinks it $3.5\times$ while maintaining BF16 accuracy. We validate SpotAttention on five backbones across two architecturally distinct families, Qwen3 (Yang et al., 2025a) (dense attention) and Qwen3.5 (hybrid linear/full attention) at 4B–32B parameters, and show that it matches dense accuracy at contexts up to 128K (Section 4).

Contributions.

- **Plug-in block-sparse attention for pretrained transformers.** A small selector attached to every full-attention layer, trained alone by KL distillation against the dense attention, runs as a single tensor-core matmul.
- **Dual top-p on the learned selection distribution.** We introduce a dual top-p rule that reads the per-query, per-layer budget directly from the selector's estimated attention distribution, with sink and recency blocks reserved so they do not dominate the nucleus, and with no separate pruning stage. It matches static top-K accuracy at a smaller mean budget.
- **Selector training and inference behaviour.** We show that the selector matches each backbone's teacher distribution across five backbones from 4B to 32B. Once trained, the selector reveals a large per-layer budget asymmetry that no static K can match, giving the dual top-p rule a mechanism-level justification.

2 Method

SpotAttention attaches a small selector to every full-attention layer of the pretrained backbone, training it by KL divergence to match the layer's head-averaged attention distribution. During inference, the selector scores the past keys of each query and picks a sparse subset, under either a fixed-budget static top-K rule or a content-adaptive dual top-p rule. Figure 2 illustrates the method for a single query.

2.1 Selector architecture

The selector is a tiny multi-head Q-K scorer with H_{idx} heads of reduced width d_{idx}. For a query at position q and a key at position t, it computes a head-weighted ReLU score $s_s[q,t] = \sum_{h=1}^{H_{\text{idx}}} w_h \operatorname{ReLU}\big(\mathbf{q}_h^I \cdot \mathbf{k}_{h,t}^I\big)$, whose softmax over t gives the estimated attention distribution $p^{\sigma}_{q,t}$. Here $\mathbf{q}_h^I, \mathbf{k}_{h,t}^I$ are the per-head query/key projections and W_w a small linear map producing the per-query mixing weights w_h from the hidden state $\mathbf{h}_q$. Scoring is block-wise: one score per query-block $\times$ key-block tile rather than per token, so selection runs as a single tensor-core matmul.

2.2 Training objectives

The selector is trained by KL divergence to match the backbone's attention, the teacher path shown in the dashed top band of Figure 2. We consider two KL variants that differ in their support: *DenseKL* computes the loss over all T keys, and *SparseKL* computes it only over the selector's own top-K selected set. Writing $s_t[h,q,\cdot]$ for the backbone's attention scores in head h and $s_s[q,\cdot]$ for the selector's scores, the teacher p^{τ} and student p^{σ} distributions are

$$p^{\tau}_{b,q,t} = \tfrac{1}{H_q} \sum_{h=1}^{H_q} \operatorname{softmax}_t\big(s_t[h,q,\cdot]\big), \qquad (1)$$

$$p^{\sigma}_{b,q,t} = \operatorname{softmax}_t\big(s_s[q,\cdot]\big). \qquad (2)$$

The teacher averages the H_q per-head softmaxes (softmax then mean), matching DSA's construction (DeepSeek-AI, 2025); the student is a single softmax over s_s.

We train the selector by forward KL from the teacher to the student. *DenseKL* matches the full distribution over all T keys:

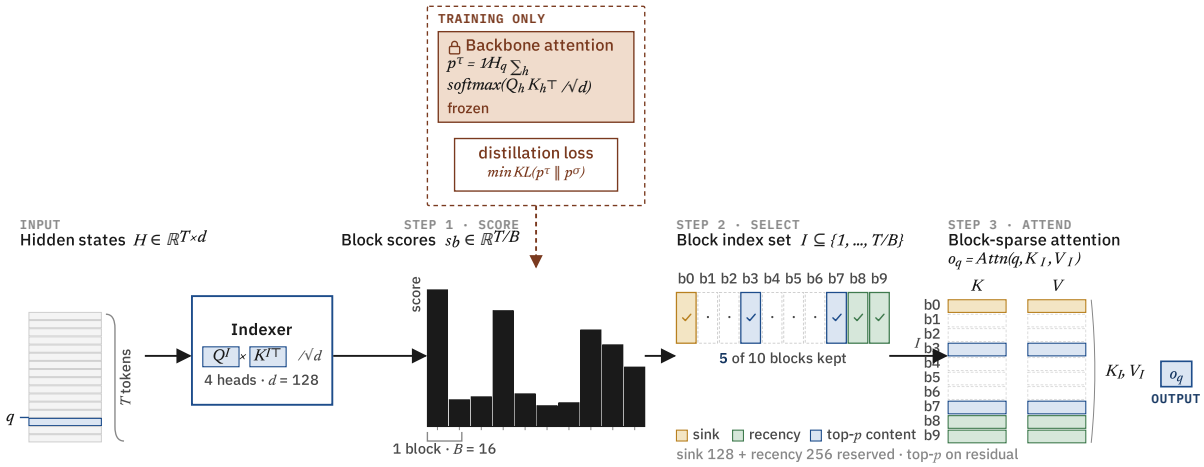

Figure 2: **SpotAttention for a single query.** The lightweight *Indexer* (a small selector) scores every past block; a dual top-p rule then selects a block index set I comprising a sink prefix, a recency window, and the top-p content blocks, and attention runs only over the gathered K_I, V_I. The dashed top band shows the teacher path used during training: the backbone's attention p^τ serves as the target, and the selector's distribution p^σ is fit to it by KL divergence.

$$\mathcal{L}_{\mathrm{KL}}^{\mathrm{Dense}} = \frac{1}{BL} \sum_{b,q} \sum_{t=1}^{T} p^{\tau}_{b,q,t} \log \frac{p^{\tau}_{b,q,t}}{p^{\sigma}_{b,q,t}}. \quad (3)$$

SparseKL matches only over the selector's own top-K selected set $\mathcal{S}_q$, with both distributions renormalized to that set:

$$\mathcal{L}_{\mathrm{KL}}^{\mathrm{Sparse}} = \frac{1}{BL} \sum_{b,q} \sum_{t\in\mathcal{S}_q} \widetilde{p}^{\tau}_{b,q,t} \log \frac{\widetilde{p}^{\tau}_{b,q,t}}{\widetilde{p}^{\sigma}_{b,q,t}}, \quad (4)$$

where $\widetilde{p}$ denotes a distribution restricted and renormalized to $\mathcal{S}_q$. The KL variant chosen at training time determines the shape of the learned distribution, which controls how far dynamic selection extrapolates (Section 5.2).

The total training loss is the KL alone:

$$\mathcal{L} = \mathcal{L}_{\mathrm{KL}}, \qquad \mathcal{L}_{\mathrm{KL}} \in \{\mathcal{L}_{\mathrm{KL}}^{\mathrm{Dense}}, \mathcal{L}_{\mathrm{KL}}^{\mathrm{Sparse}}\}. \quad (5)$$

2.3 Selection modes

At inference, the selector scores every past block for each query, and the layer attends only to keys in a selected subset $\mathcal{S}_q \subseteq \mathcal{B}$ of blocks. Both modes use the per-block maximum $\bar{s}_s[q,b] = \max_{t\in b} s_s[q,t]$, and differ only in how they pick $\mathcal{S}_q$ from those values.

Static top-K. A fixed budget of K blocks per query,

$$\mathcal{S}_q = \underset{b\in\mathcal{B}}{\mathrm{argTop}\text{-}K}\ \bar{s}_s[q,b]. \quad (6)$$

The fixed budget measures how well the selector ranks blocks against the teacher it learned from, and serves as the baseline for the adaptive mode below. A single K cannot match the per-layer budget asymmetry the selector reveals (Section 5.1), which motivates dual top-p.

Dual top-p (dynamic budget). Dual top-p sets the budget from the selector's own distribution, in two tiers. First, a fixed prefix of *sink* blocks $\mathcal{S}_{\mathrm{sink}}$ and a fixed suffix of *recency* blocks $\mathcal{S}_{\mathrm{rec}}$ (Xiao et al., 2024b) are reserved: these carry a large share of attention mass at any length. Second, nucleus selection (Holtzman et al., 2020) runs on the residual distribution

$$\widetilde{p}^{\sigma}_{q,b} = \underset{b\in\mathcal{B}_{\mathrm{res}}}{\mathrm{softmax}}\ \bar{s}_s[q,b], \quad (7)$$

with $\mathcal{B}_{\mathrm{res}} = \mathcal{B} \setminus (\mathcal{S}_{\mathrm{sink}} \cup \mathcal{S}_{\mathrm{rec}})$ the residual blocks, yielding the selected set

$$\mathcal{S}_q = \mathcal{S}_{\mathrm{sink}} \cup \mathcal{S}_{\mathrm{rec}} \cup \mathcal{S}_{\mathrm{nuc}}(q), \qquad |\mathcal{S}_q| \geq K_{\min}, \quad (8)$$

where $\mathcal{S}_{\mathrm{nuc}}(q)$ is the smallest prefix of $\mathcal{B}_{\mathrm{res}}$, sorted by $\widetilde{p}^{\sigma}_{q,\cdot}$ descending, whose cumulative mass reaches p. The floor $K_{\min}$ guards against degenerate-peaky tiles by extending $\mathcal{S}_{\mathrm{nuc}}$ along descending scores when nucleus alone falls short. The budget adapts per query and per layer, with no separate pruning stage.

3 Experimental setup

Backbones and training. We train the selector on five backbones spanning two families: Qwen3

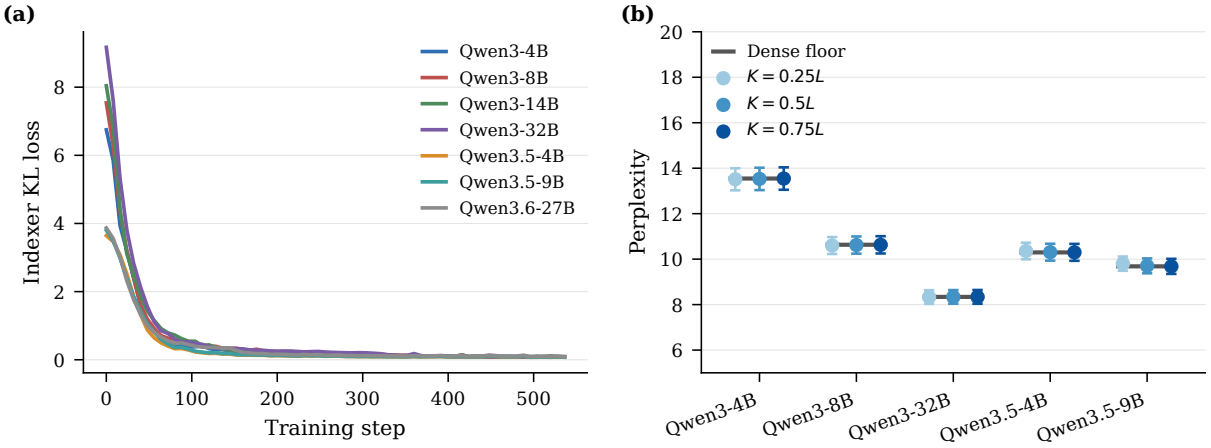

Figure 3: **Training dynamics.** **(a)** Selector KL loss vs. training step for seven backbones across three families (Qwen3 4B–32B, Qwen3.5-4B/9B, Qwen3.6-27B); every curve falls to near zero within the first ∼200 steps, so one recipe converges across scale and architecture. **(b)** Held-out perplexity for the five evaluation backbones under sparse selection at $K \in \{0.25, 0.5, 0.75\}L$, against the dense floor (gray); sparse perplexity matches the floor at every budget (error bars: ± 1 standard error over held-out 16K blocks).

(Yang et al., 2025a), and Qwen3.5, at sizes 4B–32B. The five long-context evaluations cover Qwen3-4B/8B/32B with dense attention and Qwen3.5-4B/9B with hybrid linear/full attention. We use FineWeb-Edu (Penedo et al., 2024) for 100M tokens at $L = 16$K context with $K_{\text{train}} = 8$K selected tokens. We optimize with AdamW (Loshchilov and Hutter, 2019) under a warmup–constant–decay schedule. The full training recipe and all hyperparameters are in Appendix A.

Evaluation. We evaluate accuracy on four long-context benchmarks at context lengths up to 128K: RULER (Hsieh et al., 2024), BABILong (Kuratov et al., 2024) (qa1–qa3), InfiniteBench (Zhang et al., 2024), and LongBench-v2 (Bai et al., 2025); per-task sample counts and length bins are in Appendix C.1. Qwen3 is capped at 32K, its native maximum position; Qwen3.5 runs to the full 128K. We score with exact match after answer extraction. Each model runs in non-thinking mode, with sampling parameters as shown in Appendix C.2. We report both selection modes (Section 2.3): static top-K at $K \in \{0.25, 0.5, 0.75\}\,L$ and dual top-p at $p \in \{0.7, 0.8, 0.9\}$. Prefill is dense throughout; sparse selection is applied only at decode.

Baselines. We compare SpotAttention against the dense backbone and two representative training-free baselines: Quest (Tang et al., 2024) and Twilight (Lin et al., 2025).

Implementation. To isolate the routing algorithm from implementation drift, we implement SpotAttention, Quest, and Twilight in Triton with with their paper-exact configurations, with FlashAttention as the dense baseline (Dao et al., 2022). Latency is measured on a single NVIDIA B200.

Quantization. We also quantize the selector's queries and key cache to INT8, INT4, FP8, and FP4, and measure the resulting accuracy and memory/latency trade-offs (Section 5.3).

4 Results

In this section we trace the selector's training dynamics and show that its post-training perplexity holds against the dense baseline. We then report the latency breakdown of selector and attention for Qwen3-8B across the $L = 8$K to $L = 512$K sweep, alongside real-world throughput improvements across sequence lengths, and finally accuracy at $K{=}0.5L$ across all five evaluation backbones and generalisation to $L = 128$K on Qwen3.5.

4.1 Training dynamics

Figure 3(a) shows the selector trains quickly and the KL loss converging from its initial value to near zero within the first ∼200 steps, across every backbone we trained. Figure 3(b) shows that held-out perplexity under sparse selection sits on the dense floor at every budget we test ($K = 0.25L, 0.5L, 0.75L$), within error bars: sparse selection costs no measurable perplexity. Perplexity is measured over full 16K windows of held-out FineWeb-Edu against the dense floor; because the backbone is frozen and the selector never trains on

Method	$L = 8$K		$L = 16$K		$L = 32$K		$L = 64$K		$L = 128$K		$L = 256$K		$L = 512$K	
	Sel.	Attn	Sel.	Attn	Sel.	Attn	Sel.	Attn	Sel.	Attn	Sel.	Attn	Sel.	Attn
FlashAttention (Dao et al., 2022)	—	0.85	—	1.10	—	1.61	—	2.62	—	4.69	—	8.74	OOM	
Twilight (Lin et al., 2025)	0.37	0.22	0.66	0.32	1.22	0.42	2.35	0.84	4.59	1.49	OOM		OOM	
SpotAttention (top-K=0.5L)	**0.24**	0.29	**0.29**	0.47	**0.53**	0.83	1.12	2.28	**1.55**	2.97	**2.87**	5.81	**5.52**	11.41
SpotAttention (top-p=0.7)	**0.24**	**0.18**	**0.29**	**0.21**	**0.53**	**0.29**	**0.84**	**0.76**	**1.55**	**1.32**	2.88	**2.04**	5.54	**4.04**
SpotAttention (top-p=0.9)	**0.24**	0.25	**0.29**	0.33	**0.53**	0.56	**0.84**	1.48	**1.55**	2.83	**2.87**	4.63	5.53	9.18
SpotAttention (K=0.5L, INT4)	0.32	0.29	0.41	0.47	0.60	0.83	1.04	1.54	1.91	2.93	3.65	5.75	7.06	11.28
SpotAttention (K=0.5L, FP4)	0.31	0.29	0.46	0.47	0.72	0.83	1.32	1.55	2.51	2.96	4.87	5.80	9.53	11.37

Table 1: **Per-token decode latency on Qwen3-8B (ms).** *Sel.* = time the selector takes to decide which keys to attend to; FlashAttention has no selector step. *Attn* = time to run attention on the selected (or full) KV cache. *Attn* is unchanged across precisions because quantization affects only the selector's K-cache, not the backbone attention. $L > 32$K runs through YaRN factor 16.

cross-entropy, the gap from the floor measures the cost of selection alone.

4.2 Decode latency

To isolate the cost of selection, we profile decode latency of the attention kernel and selector kernel on Qwen3-8B against FlashAttention (Dao et al., 2022) and Twilight (Lin et al., 2025) across sequence lengths from $L = 8$K to $L = 512$K, with YaRN factor 16 extending the model beyond its native 32K window. Table 1 reports the per-step decomposition; Figure 4 reports end-to-end throughput.

Per-step decomposition. Table 1 shows that SpotAttention's sparse attention is 1.6–3.6× faster than FlashAttention at $L = 128$K depending on selection mode, and the BF16 selector is ∼3× faster than Twilight at the same length, 1.55 ms vs 4.59 ms. Quantizing the selector's K-cache to INT4 or FP4 cuts its memory footprint with no accuracy loss; the per-step selector cost rises slightly (1.91 ms INT4 and 2.51 ms FP4 at $L = 128$K) because the dequantization overhead exceeds the savings from smaller K-cache reads when the BF16 kernel is already memory-light. Twilight runs out of memory at $L = 256$K and FlashAttention at $L = 512$K, where SpotAttention's static K=0.5L variant still routes in 5.52 ms.

End-to-end throughput. Figure 4 shows that SpotAttention's static K=0.5L reaches 107 tok/s at $L = 32$K (3.0× FlashAttention, 1.6× Twilight) and 70 tok/s at $L = 128$K (3.9× FlashAttention, 1.8× Twilight); at $L = 512$K it sustains 36 tok/s, where both baselines have run out of memory. Throughput falls only slowly with L because the

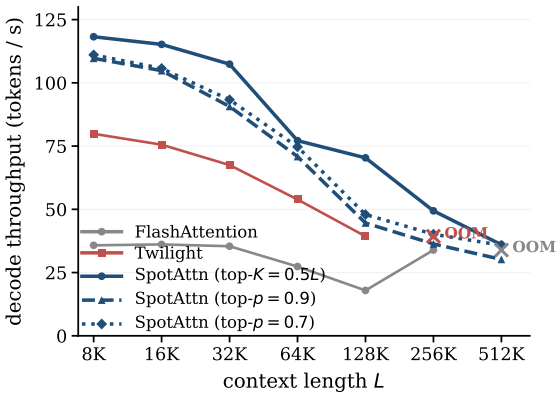

Figure 4: **Decode throughput across context length** on Qwen3-8B. SpotAttention sustains the highest throughput across all three selection modes from $L = 8$K to $L = 512$K. Twilight OOMs at $L = 256$K, while FlashAttention OOMs at $L = 512$K.

per-step cost is dominated by the backbone matmul and KV reads. FlashAttention degrades smoothly through $L = 128$K but rebounds at $L = 256$K, which we attribute to a tiled-attention kernel switching at that length. Implementation choices shape end-to-end throughput as much as the algorithm itself, and the same routing scheme can show different speedups across kernels and hardware.

4.3 Accuracy

At matched budget K=0.5L, SpotAttention lands within a single bootstrap standard error of the dense backbone on all five models, and dual top-p at p=0.9 tracks the static variant; Quest and Twilight at matched budgets on Qwen3-4B sit in the same bootstrap-error band, so the distinguishing axis across selectors is per-step cost, not accuracy. We also show that INT4/FP4 quantization on the selec-

Model	Vanilla	SpotAttention			
		K=0.5L	K=0.75L	top-p=0.7	top-p=0.9
Qwen3-4B	0.543 (0.019)	0.524 (0.019)	0.528 (0.018)	0.505 (0.020)	0.523 (0.019)
Qwen3-8B	0.537 (0.019)	0.544 (0.019)	0.544 (0.019)	0.525 (0.020)	0.549 (0.020)
Qwen3-32B	0.599 (0.020)	0.581 (0.020)	0.577 (0.019)	0.516 (0.022)	0.572 (0.020)
Qwen3.5-4B	0.561 (0.014)	0.569 (0.014)	0.557 (0.014)	0.563 (0.014)	0.572 (0.014)
Qwen3.5-9B	0.558 (0.015)	0.566 (0.014)	0.566 (0.013)	0.573 (0.013)	0.566 (0.013)

Table 2: **Accuracy parity across backbones.** Each cell is the mean over RULER, BABILong, and LongBench-v2 cell-overalls with bootstrap standard errors in parentheses (Appendix C.5); InfiniteBench is excluded so the average is apples-to-apples across Qwen3 (32K cap) and Qwen3.5 (128K). All five backbones land within a single standard error of dense across both static budgets and both dual top-p settings.

tor's K-cache stays within bootstrap error of BF16 across the Qwen3 family, as shown in Section 5. Full per-dataset grid, the bootstrap protocol, and quantization results are detailed in Appendices D, C.5, and G.

4.4 Training-length invariance

We show that a selector trained at fixed sequence length generalizes to any evaluation length. Figure 5 shows accuracy by context length on RULER, InfiniteBench, and LongBench-v2 for the dense backbone, SpotAttention static K=0.5L, and SpotAttention dual top-p=0.9 on Qwen3.5. Both SpotAttention variants track the dense curve at every length bin on every dataset, and neither InfiniteBench's 64K–128K span nor LongBench-v2's four bins shows any decay of SpotAttention relative to dense as length grows.

5 Analysis and ablations

In this section we analyse the trained selector: what it has learned, what the KL objective shapes it into, and how it tolerates lower-precision storage.

5.1 Per-layer K asymmetry

The dual top-p selector chooses how many blocks each layer keeps, and the spread it picks is informative. Figure 6 plots the mean per-layer K at p=0.9 on RULER at 16K. In the three dense Qwen3 backbones, early layers ask for $K/L \approx 0.7$–0.85 of the context, while layers past two-thirds depth drop to $K/L \approx 0.05$–0.15.

The hybrid Qwen3.5 backbones carry only eight full-attention layers, yet those eight still span a U-shape: the asymmetry is not a dense-stack artifact. Appendix E shows that the same per-layer signature persists across BABILong, LongBench-v2, and $p \in \{0.7, 0.8, 0.9\}$, so the asymmetry is structural to the backbone rather than a property of the workload or budget.

5.2 Selector generalization

Empirical shape gap. We trained the selector under DenseKL to test whether it helps the model generalize better or faster than SparseKL. DenseKL turns out razor-sharp: at $L = 32$K on a $L_{\text{train}} = 16$K Qwen3-4B checkpoint, the top-1 block carries a mean of 72% of the residual mass per query, ranging 24%–99% across layers. SparseKL on the same backbone averages 34%, ranging 2%–95%. On Qwen3.5-4B the gap is wider still: 84% vs 5% on average. The DenseKL cumulative-mass curve in Figure 7 saturates the top-p threshold at $K \approx 10$ blocks regardless of p, so peaky distributions cap content reach long before the budget can grow.

Cross-backbone transfer. We trained the selector on backbones whose teacher patterns differ in scale and architecture. In every case the SparseKL student tracks the teacher's spread closely; the full heatmap grid is in Appendix F. One selector geometry therefore produces a distribution that matches whichever backbone it sits on: the recipe transfers shape, not just convergence.

5.3 Quantized selector: a memory and prefill lever

The selector keeps its own lightweight K cache to compute attention scores. Halving its precision is a constant-factor saving on memory and bandwidth; at decode the backbone KV-cache bandwidth dominates the total, so the per-step selector cost barely moves under quantization. We quantize the selector's queries and key cache to INT8, INT4, FP8, and FP4 (group-32 microscale) and measure the resulting accuracy and memory/latency trade-offs.

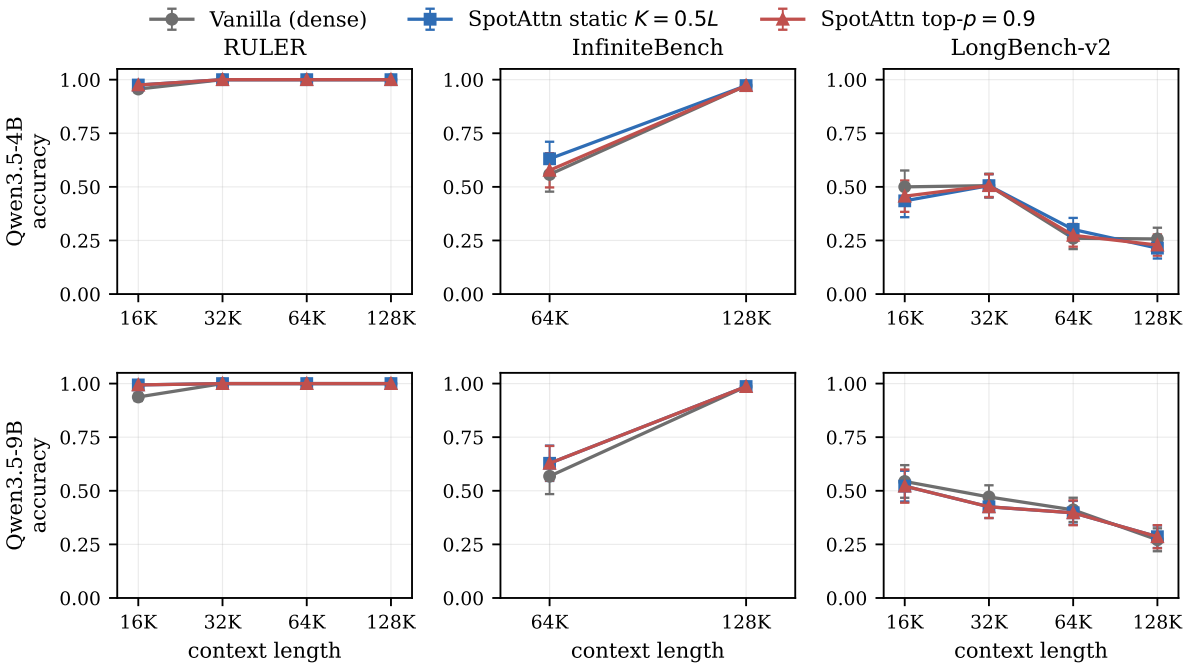

Figure 5: **Training-length generalization.** Mean accuracy across context lengths for SpotAttention static $K{=}0.5L$ (blue) and SpotAttention dual top-$p{=}0.9$ (red) follows the dense curve without an accuracy gap.

Accuracy. Table 9 in Appendix G reports Qwen3-4B, -8B, and -32B accuracy under dual top-$p{=}0.9$ with the selector's queries and key cache quantized. Every variant stays within bootstrap error of the BF16 reference on the dataset-mean column at every backbone. The selector is trained under BF16 and quantized at inference only; the backbone KV cache is unchanged throughout.

Memory and latency. The selector's K-cache scales with L, so a precision cut is a linear deployment lever. INT4 / FP4 microscale at group 32 stores 4.5 bits per element, $0.281\times$ the BF16 footprint: on Qwen3-8B at $L = 128$K the selector K-cache shrinks from 4.50 GiB at BF16 to 1.27 GiB at INT4 or 1.20 GiB at FP4. Per-step selector latency is essentially flat under INT4 at 1.91 ms versus BF16's 1.55 ms, and moderately higher under FP4 at 2.51 ms; the dequantization overhead at decode exceeds the savings from smaller K-cache reads when the BF16 kernel is already memory-light. Single-stream decode throughput follows: at $L = 128$K, INT4 matches BF16 at 70 tok/s and FP4 drops to 58 tok/s. The wins from quantization therefore land in memory and in prefill, where the selector's K-cache size is the dominant cost, rather than in single-stream decode latency.

6 Related work

Sparse attention trained with the backbone. DeepSeek Sparse Attention (DeepSeek-AI, 2025), whose lightning indexer and KL distillation we build on, trains the indexer jointly with the backbone, and the DeepSeek Engram follow-up adds conditional, lookup-based memory as a further axis of sparsity (Cheng et al., 2026). NSA (Yuan et al., 2025) and MoBA (Lu et al., 2025) learn block-sparse attention from scratch, while Longformer (Beltagy et al., 2020) and BigBird (Zaheer et al., 2020) fix the pattern entirely, attending over static windows and global tokens. SpotAttention trains the selector alone, against the dense attention of an already-trained backbone.

Training-free KV-cache selection. Many methods select from the KV cache at inference, with no training. Fixed-budget selectors score blocks of keys (Quest (Tang et al., 2024), InfLLM (Xiao et al., 2024a)), compress the cache during prefill (SnapKV (Li et al., 2024), Ada-KV (Feng et al., 2025)), pick query-aware patterns (MInference (Jiang et al., 2024)), evict or window tokens (H2O (Zhang et al., 2023), StreamingLLM (Xiao et al., 2024b), DuoAttention (Xiao et al., 2025)), sample by hashing (MagicPIG (Chen et al., 2025), HashEvict (Liu et al., 2024a)), or target specific regimes

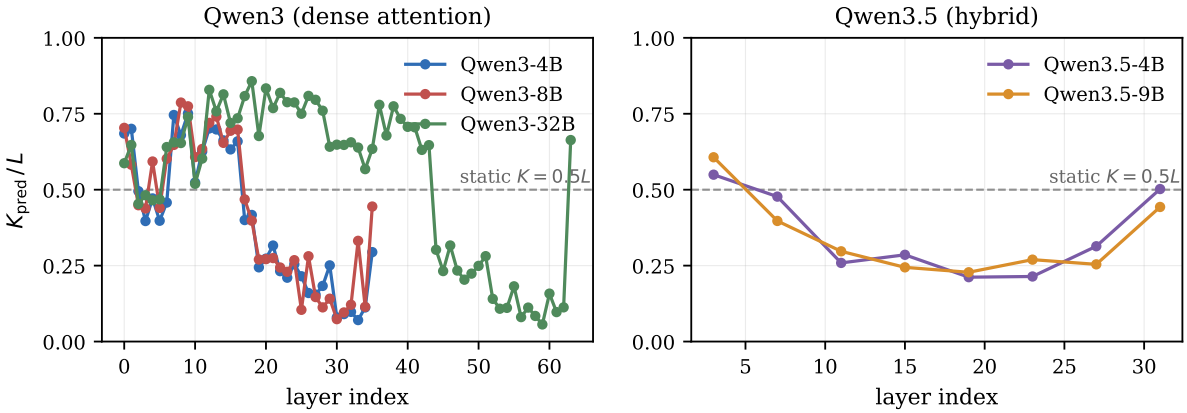

Figure 6: **Per-layer K selected under dual top-p.** Mean K per attention layer at p=0.9 on RULER at 16K context. Dashed line marks the static K=0.5L reference. **Left:** the three dense Qwen3 backbones show early-to-middle layers reaching $K/L \approx$ 0.7–0.85, while the last third of each network uses very few tokens. **Right:** Qwen3.5's hybrid backbones carry only eight full-attention layers, and even at that depth the selector picks a U-shaped range.

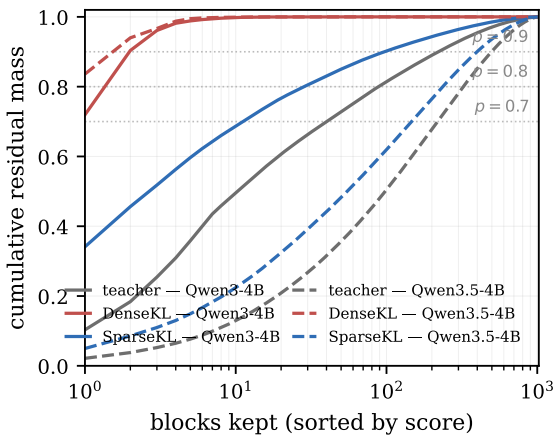

Figure 7: **KL-scope determines selector shape.** Cumulative residual mass against blocks kept in descending score order, measured at the last query. Curves: teacher (gray), DenseKL student (red), SparseKL student (blue). DenseKL hits cumulative mass 0.9 within ∼10 blocks; SparseKL tracks the teacher curve.

(SpargeAttention (Zhang et al., 2025), LessIsMore (Yang et al., 2025b)). Adaptive variants adjust the budget: Twilight (Lin et al., 2025) layers hierarchical top-p pruning on Quest's selector, Tactic (Zhu et al., 2026) fits a distribution to an attention-mass target, FlexPrefill (Lai et al., 2025) thresholds per head at prefill, and PyramidKV (Cai et al., 2025) adapts across layers. These selectors are all hand-designed; SpotAttention's is learned, and its distribution is calibrated enough for a nucleus rule to threshold directly. Orthogonal to selection, the cache itself can be compressed via paged management (Kwon et al., 2023) or low-precision storage (Hooper et al., 2024; Liu et al., 2024b), and these compose with SpotAttention.

Learned retrofit selectors. The methods closest to our setting also learn a small selector for a frozen backbone. SeerAttention (Gao et al., 2024) trains a pooled attention gate with regression supervision; MISA (Zhou et al., 2026), concurrent with our work, swaps the lightning indexer for a mixture-of-experts router; both keep a fixed budget. SparQ (Ribar et al., 2024) approximates the query–key product cheaply, again at a fixed budget and without learning. SpotAttention distils per-token KL from the dense teacher, which calibrates the selector's distribution enough to drive a dynamic budget. The Sparse Frontier survey (Nawrot et al., 2025) catalogues the representative methods from which we draw our baselines.

7 Conclusion

We train a lightweight selector to estimate dense attention distributions and apply a dual top-p rule at inference to pick a variable numberof blocks per query and per layer. It learns quickly, transfers across backbones, and matches dense accuracy at every budget we test, generalizing from a 16K training length to 128K evaluation without an accuracy gap. On Qwen3-8B SpotAttention sustains 70 tok/s at 128K, 3.9× faster than FlashAttention and 1.8× faster than Twilight, and reaches 36 tok/s at 512K where both baselines have run out of memory. Quantization shrinks the selector's K-cache 3.5× at no accuracy cost.

Limitations

Our largest evaluated backbone is 32B and all five evaluation backbones come from the Qwen family on English-only long-context benchmarks; behaviour on other architectures (Llama, Mistral) and non-English data is untested. Sparse-prefill latency is not benchmarked, though the selector trains and runs in the prefill regime, so the path to a sparse-prefill kernel is clear. Dual top-p assumes a well-calibrated selector distribution; under DenseKL training this calibration collapses (Section 5.2), and the rule then under-selects. Selector quantization improves memory and prefill cost but not single-stream decode latency, which is bounded by backbone KV bandwidth; its behaviour under large-batch multi-GPU decode is also not measured.

References

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. 2025. LongBench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3639–3664.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. 2025. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Conference on Language Modeling (COLM)*.

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

Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, Han Zhang, Huishuai Zhang, Dongyan Zhao, and Wenfeng Liang. 2026. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv:2601.07372*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*.

DeepSeek-AI. 2025. DeepSeek-V3.2: Pushing the frontier of open large language models. *arXiv:2512.02556*.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. 2025. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. 2024. SeerAttention: Learning intrinsic sparse attention in your LLMs. *arXiv:2410.13276*.

Gemini Team. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv:2403.05530*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: What's the real context size of your long-context language models? In *Conference on Language Modeling (COLM)*.

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Yuri Kuratov, Aydar Bulatov, Petr Anokhin, Ivan Rodkin, Dmitry Sorokin, Artyom Sorokin, and Mikhail Burtsev. 2024. BABILong: Testing the limits of LLMs with long context reasoning-in-a-haystack. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*.

Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. 2025. FlexPrefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *International Conference on Learning Representations (ICLR)*.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM knows what you are looking for before generation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Minghui Liu, Tahseen Rabbani, Tony O'Halloran, Ananth Sankaralingam, Mary-Anne Hartley, Furong Huang, Cornelia Fermüller, and Yiannis Aloimonos. 2024a. HashEvict: A pre-attention KV cache eviction strategy using locality-sensitive hashing. *arXiv:2412.16187*.

Zichang Liu, Jue Wang, Tri Dao, Tianyi Zhou, Binhang Yuan, Zhao Song, Anshumali Shrivastava, Ce Zhang, Yuandong Tian, Christopher Ré, and Beidi Chen. 2023. Deja vu: Contextual sparsity for efficient LLMs at inference time. In *Proceedings of the 40th International Conference on Machine Learning (ICML)*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, and 6 others. 2025. MoBA: Mixture of block attention for long-context LLMs. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Piotr Nawrot, Robert Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo M. Ponti. 2025. The sparse frontier: Sparse attention trade-offs in transformer LLMs. *arXiv:2504.17768*.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The FineWeb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2024. SparQ attention: Bandwidth-efficient LLM inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, pages 42558–42583.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024a. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2025. DuoAttention: Efficient long-context LLM inference with retrieval and streaming heads. In *International Conference on Learning Representations (ICLR)*.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025a. Qwen3 technical report. *arXiv:2505.09388*.

Lijie Yang, Zhihao Zhang, Arti Jain, Shijie Cao, Baihong Yuan, Yiwei Chen, Zhihao Jia, and Ravi Netravali. 2025b. Less is more: Fast and accurate reasoning with cross-head unified sparse attention. *arXiv:2508.07101*.

Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. 2025. SpargeAttention: Accurate and training-free sparse attention accelerating any model inference. In *International Conference on Machine Learning (ICML)*.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, and Maosong Sun. 2024. ∞Bench: Extending long context evaluation beyond 100K tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

Kan Zhu, Tian Tang, Qinyu Xu, Yile Gu, Zhichen Zeng, Rohan Kadekodi, Liangyu Zhao, Ang Li, Arvind Krishnamurthy, and Baris Kasikci. 2026. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context LLMs. In *International Conference on Learning Representations (ICLR)*.

A Hyperparameters

Table 3 lists the full selector retrofit recipe. Per-backbone runs vary only the pretrained checkpoint identifier; the selector geometry, optimiser, schedule, and batch arithmetic are identical across all five backbones.

B Held-out perplexity protocol

We report token-level perplexity, $\mathrm{PPL} = \exp(\overline{\mathrm{CE}})$, with $\overline{\mathrm{CE}}$ the mean next-token cross-entropy.

Data. We score on FineWeb-Edu (Penedo et al., 2024) (the `sample-10BT` split) through the same streaming, EOS-delimited packing loader as selector training, packed to 16,384-token blocks. The only change from training is the shuffle seed: a held-out seed (`987654321`) distinct from the training seed, so the document ordering differs. Contamination is moot regardless: the backbone is frozen and the selector never optimizes cross-entropy, so no component is fit to this metric.

Scoring. We score all $L-1$ next-token predictions in each 16K block and forward enough blocks to score at least $65{,}536$ tokens (≈ 82K scored tokens per cell at batch size 1).

Forward regime. Each block is one prefill forward (`use_cache=False`), so for the sparse model the selector and block-sparse kernels run exactly as in training. Positions run sequentially $0 \ldots L-1$ under a full causal mask, matching training; the loader's per-document position resets are disabled here, since resetting RoPE mid-block while attention still crosses document seams produces incoherent logits.

Operating points. The sparse series pin the selector budget to $\lfloor cL \rfloor$ rounded to the block size 16, for $c \in \{0.25, 0.5, 0.75\}$: 4,096, 8,192, and 12,288 keys at $L = 16$K. The dense floor runs the same checkpoint with the selector disabled, attending over the full context.

Error bars. Each 16K block is one independent unit. We take the standard error of the mean cross-entropy as $\mathrm{stdev}/\sqrt{n}$ over blocks and propagate it to perplexity by the delta method, $\mathrm{SE}(\mathrm{PPL}) = \mathrm{PPL} \cdot \mathrm{SE}(\overline{\mathrm{CE}})$.

C Evaluation protocol

C.1 Data configuration

Table 4 summarises the per-benchmark task selection, length bins, and sample counts. RULER and BABILong are evaluated in fixed length bins (16384, 32768, 65536, 131072); InfiniteBench and LongBench-v2 are capped at 131072 tokens. LongBench-v2 uses all available samples. Samples are then capped to each model's maximum context: 32K for Qwen3, 128K for Qwen3.5.

C.2 Decoding

Table 5 lists the per-model sampling settings. All evaluation uses sampling under each model's recommended non-thinking preset; the Qwen3 thinking preset is listed for reference only.

C.3 Prompts and answer extraction

Prompt templates. All four benchmarks share the same system prompt and two user templates, dispatched per example as free-text or multiple-choice. Both templates instruct the model to wrap its final answer in `<answer>...</answer>` so a regex extractor can recover it. The system prompt is

```
You are a helpful assistant for question-answering
tasks. Always answer based solely on the provided
context.
```

The free-text template (RULER, BABILong, InfiniteBench `passkey` / `kv_retrieval` / `longbook_qa_eng`):

```
# Context
{context}
# Question
{question}
# Instructions
Answer the question based on the context. Provide
your final answer wrapped in <answer> tags, e.g.
<answer>X</answer>.
```

The multiple-choice template (LongBench-v2, InfiniteBench `longbook_choice_eng`) reuses the context + question blocks and adds a four-option list (`(A)...(D)`), with the instruction to return the letter inside `<answer>...</answer>`. Both templates run through `tokenizer.apply_chat_template(..., add_generation_prompt=True)` so the chat formatting is the model's own.

Answer extraction. We extract the last `<answer>...</answer>` payload from the completion. For non-thinking models on every benchmark, the opening tag is appended to the prompt so the model resumes generation immediately after it; the same regex still catches the closing tag. In thinking mode, tag-prefilling

is suppressed so it does not clobber the model's native <think> block. For content-word answers (BABILong, multiple-choice, free-form F1) we score 0 when no closing tag is found; for unique-token retrieval (RULER NIAH, passkey, kv) we fall back to substring-matching the full completion, since the gold values are random nonces unlikely to match by accident.

C.4 Scoring

Each example is scored by one of three rules:

- **Substring** (RULER NIAH, BABILong, InfiniteBench `passkey` / `kv_retrieval`): the score is the fraction of gold answers that appear as case-insensitive substrings of the extracted text. With a single gold this collapses to a 0/1 indicator.
- **SQuAD-style F1** (InfiniteBench `longbook_qa_eng`): token F1 between extracted and gold after lowercasing, stripping articles and punctuation, and whitespace tokenisation; maximum across gold candidates.
- **Multiple-choice letter** (LongBench-v2, InfiniteBench `longbook_choice_eng`): the first A–D token in the extracted text, scored 1 if it matches the gold letter and 0 otherwise.

Per-example scores enter the bootstrap unchanged.

C.5 Bootstrap protocol

Every main-text and appendix accuracy number is a bootstrap mean with bootstrap standard error. We treat each cell, one (model, mode, dataset) triple, as an independent estimation problem. The unit of resampling is the per-example score the evaluator recorded: 0/1 under substring and multiple-choice scorers; $[0, 1]$ under the InfiniteBench QA F1 scorer. For each cell we draw $B = 1000$ resamples with replacement and report the cell mean alongside the standard deviation of the resample means; the seed is fixed so re-runs are bit-stable.

Standard errors of dataset-mean rows (the rightmost columns of every accuracy table) propagate the per-cell standard errors under independence across datasets, which holds because the example sets are disjoint.

Per-cell sample sizes vary by benchmark: RULER $\approx$160, BABILong 300, InfiniteBench 64–101, LongBench-v2 $\approx$270 examples per cell. A single standard error spans roughly ± 0.015 at p=0.5 on RULER, BABILong, and LongBench-v2, and ± 0.03–0.05 on InfiniteBench.

D Full accuracy results

Qwen3 family (4B, 8B, 32B). Table 6 reports the per-dataset grid for the Qwen3 family. All sparse selectors land within bootstrap error of the dense backbone at every matched budget. The 32K context cap on this family excludes InfiniteBench, whose bins start at 64K. Quest and Twilight are reported where their runs have completed: Qwen3-4B fully; Qwen3-8B and Qwen3-32B partially.

Qwen3.5 family (4B, 9B). Table 7 reports the per-dataset grid for the Qwen3.5 family at the native 128K context. Quest and Twilight have not yet been run on Qwen3.5 and are omitted.

Per-subtask resolution. Table 8 reports subtask-level accuracy on Qwen3-4B against SpotAttention static K=0.5L and dual top-p=0.9. The family-table dataset rows above are means over these subtasks.

E Per-layer K across datasets and budgets

Figure 8 extends the per-layer plot across datasets and top-p settings. The signature is dataset-robust: each row has the same shape across RULER, BABILong, and LongBench-v2. It is also budget-robust: the three top-p curves within a panel are nearly parallel, shifted up or down. Both observations support the claim in Section 5.1 that the asymmetry is structural to the backbone rather than a property of the workload.

F Selector distribution heatmaps

Figure 9 compares the teacher distribution against the DenseKL and SparseKL students at $L = 32$K on Qwen3-4B and Qwen3.5-4B. The teacher distributions are diffuse: mass spread across content positions, with a recency stripe at the right edge. DenseKL students collapse the mass into a single bin per query (the panels read as black with bright pixels only at the recency edge); SparseKL students recover the teacher's spread.

We capture one 32K FineWeb-Edu forward pass per checkpoint and bin the last-query softmax into 1024 buckets. Qwen3-4B uses production K_{train}=8K checkpoints; Qwen3.5-4B uses the matched-K pair at K_{train}=2K (no 8K DenseKL Qwen3.5 was trained).

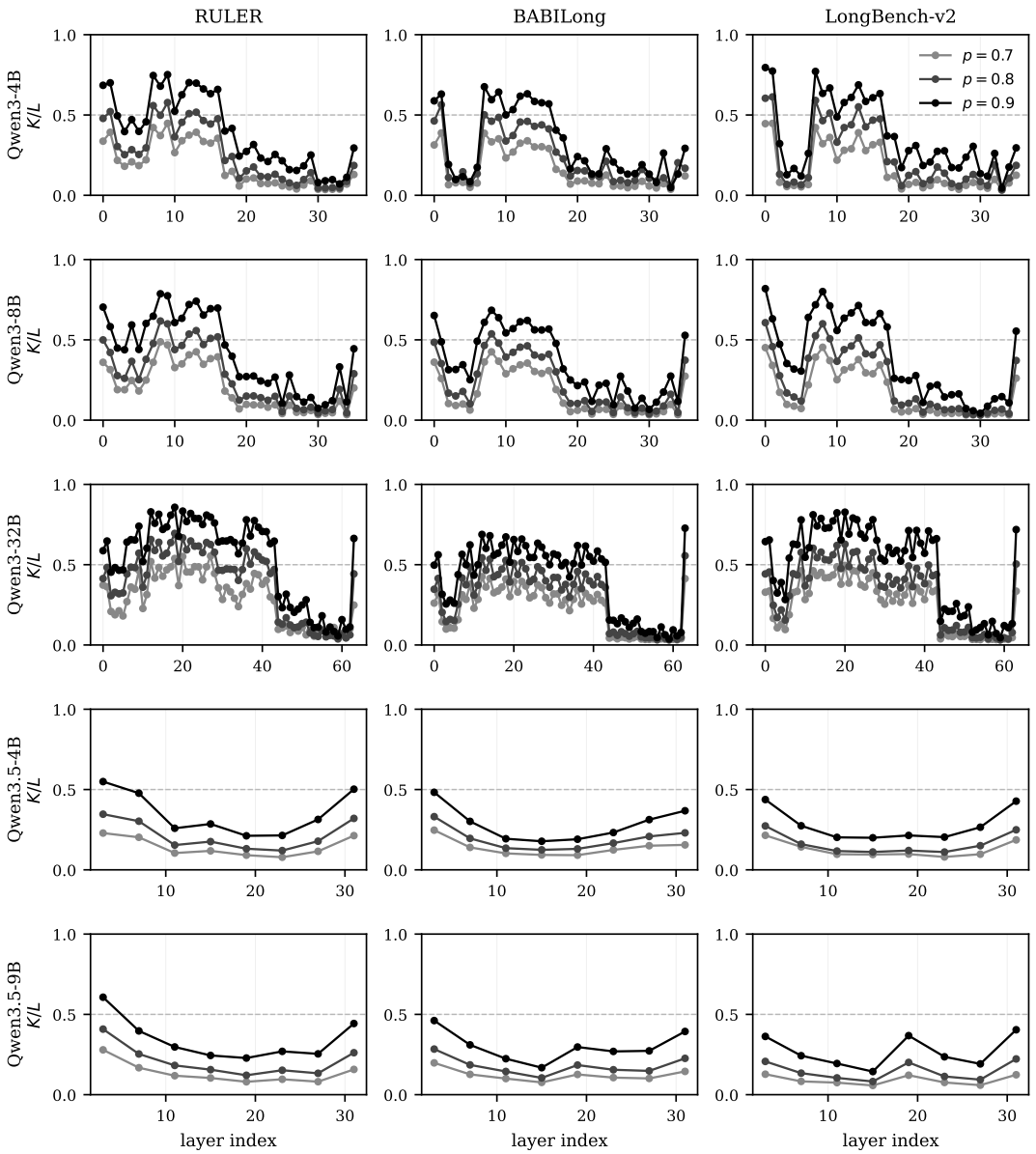

Figure 8: **Per-layer K/L across datasets and top-p values**, 16K context. Rows: backbones; columns: datasets; lines within each panel: $p \in \{0.7, 0.8, 0.9\}$ (light → dark). Dashed reference at K/L=0.5.

Figure 10 extends the comparison across the backbone family. The teacher pattern differs by scale and architecture: Qwen3.5 hybrids carry only eight full-attention layers, while the dense Qwen3 family spans 36–40. Inside each row the SparseKL panel tracks its teacher's spread closely. This is the visual evidence behind the cross-backbone transfer claim in Section 5.2.

G Selector quantization

Table 9 reports accuracy at five selector K-cache precisions. The K-cache tolerates 4-bit microscale storage with negligible accuracy loss.

H Use Of AI Assistants

AI assistance was used for polishing, copy-editing, and code review during manuscript preparation. All experimental design, scientific claims, numerical results, and final wording were authored and verified by the human authors.

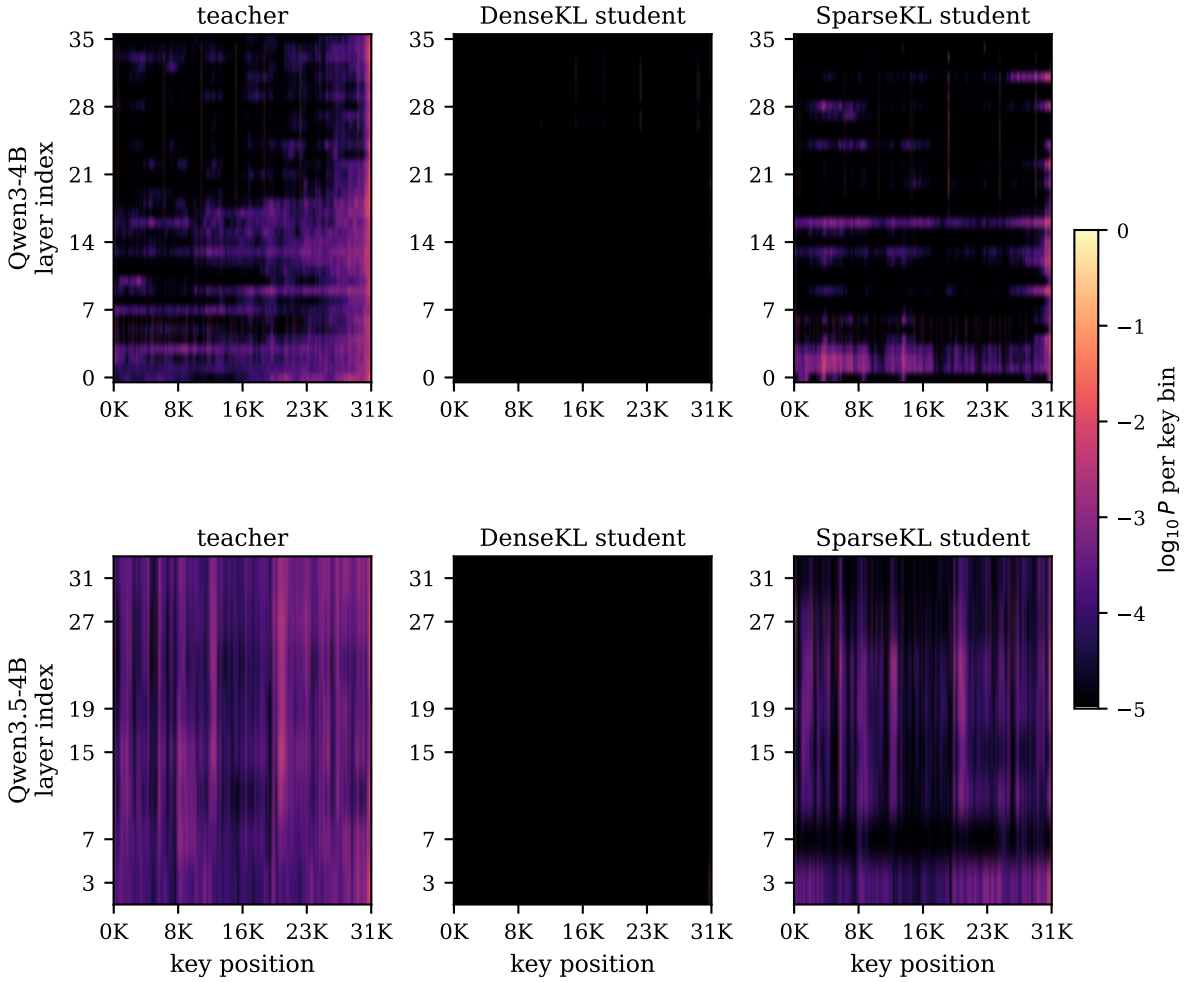

Figure 9: **Last-query softmax distributions, [layer $\times$ key bin], $L = 32$K.** Top row: Qwen3-4B (36 full-attention layers, K_{train}=8K). Bottom row: Qwen3.5-4B (8 full-attention layers, K_{train}=2K). Columns: teacher / DenseKL student / SparseKL student. Colour: $\log_{10} P$ per bin (lower bound -5). DenseKL collapses essentially all mass into a single bin per query (panel reads as black with sparse bright pixels at the right edge), while SparseKL preserves the teacher's diffuse pattern.

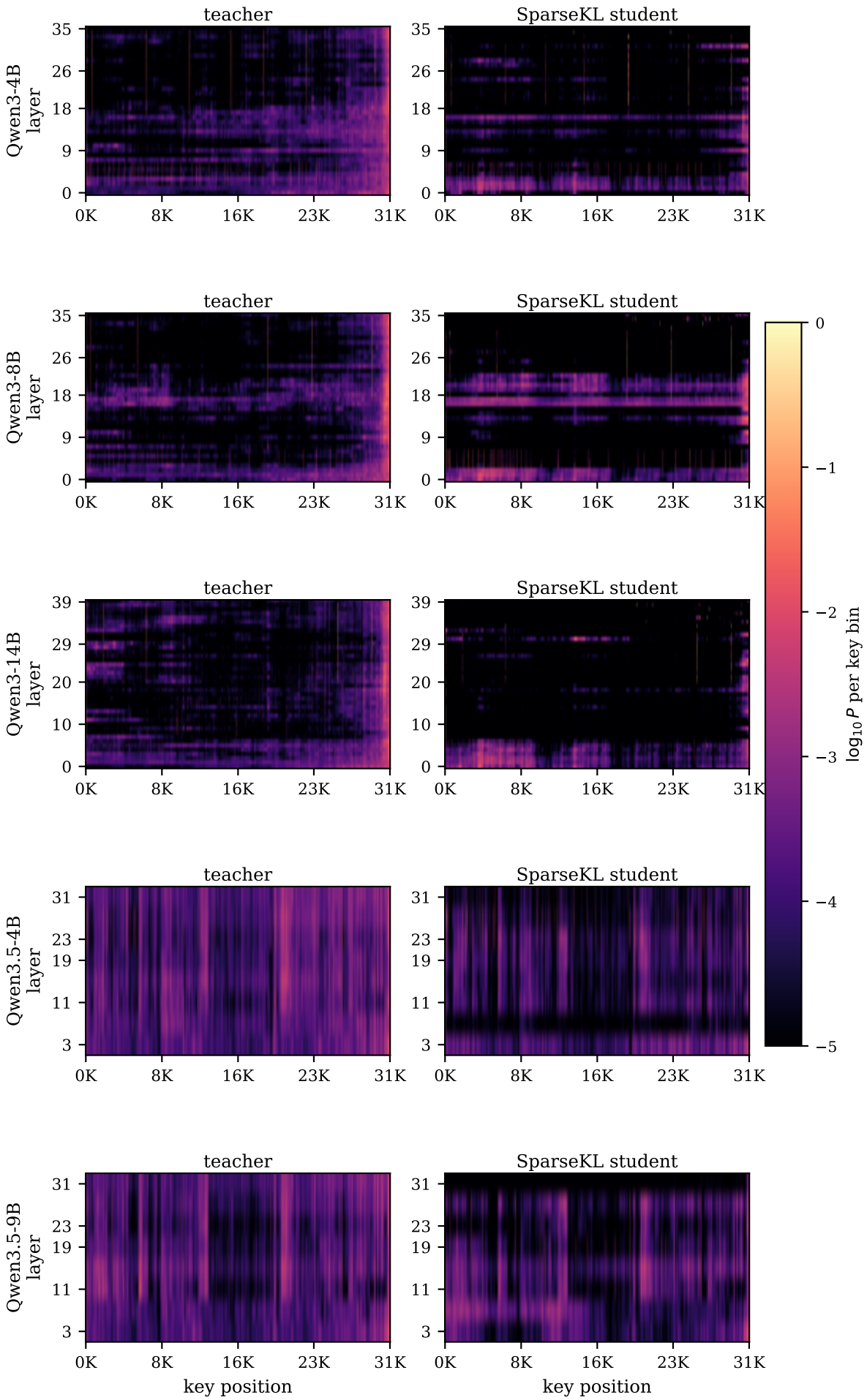

Figure 10: **Teacher vs. SparseKL student across the backbone family.** Each row is one backbone; columns are the frozen-backbone dense attention (teacher, left) and the trained SparseKL selector (right). Cells are last-query [layer $\times$ key bin] softmax heatmaps at $L = 32$K, colour $\log_{10} P$. Across Qwen3-{4B, 8B, 14B} and Qwen3.5-{4B, 9B},

Hyperparameter	Value
Selector architecture	
heads H_{idx}	4
head dim d_{idx}	128
RoPE head dim	64
block size B	16
score	$\sum_h w_h \operatorname{ReLU}(\mathbf{q}_h^I \cdot \mathbf{k}_h^I)$
selection kernel	fused-Triton top-K
Training data	
corpus	FineWeb-Edu `sample-10BT`
packing	streaming, EOS-delimited
sequence length L	16,384
training tokens	$1{\times}10^8$
seed	42
Selector training	
backbone	frozen
trainable	selector projections only
KL objective	SparseKL (Eq. 4)
K_{train}	$8{,}192 = 0.5\,L$
precision	bfloat16
Optimiser & schedule	
optimiser	AdamW
(β_1, β_2)	$(0.9, 0.95)$
ϵ	10^{-8}
weight decay	0.1
peak LR	$1{\times}10^{-3}$
min LR	$5{\times}10^{-5}$
warmup steps	100
constant-phase fraction	0.30
decay phase	cosine to min LR
gradient clip (global ℓ_2)	1.0
Batch arithmetic	
per-device batch	2 sequences
gradient accumulation	4 steps
processes (DDP)	1 GPU
effective batch	8 sequences (131,072 tokens)
optimiser steps	$\lceil 10^8/131{,}072 \rceil = 763$
Inference (selection modes)	
static top-K fractions	$K \in \{0.25,\ 0.5,\ 0.75\}\,L$
dual top-p masses	$p \in \{0.7,\ 0.8,\ 0.9\}$
sink boundary	128 tokens (absolute prefix)
recency window	256 tokens (per tile)
$K_{\min}$ floor	512 tokens (32 blocks)
softmax temperature	1.0

Table 3: **Default hyperparameters for the selector retrofit.** Training settings (top groups) and inference selection settings (bottom group) are identical across all five backbones; only the pretrained-checkpoint identifier varies per run.

Benchmark	Tasks	Lengths	Samples
RULER	all	16/32/64/128K	32
BABILong	qa1, qa2, qa3	16/32/64/128K	100
InfiniteBench	passkey, kv_retrieval, longbook_choice_eng, longbook_qa_eng	$\leq$128K	64
LongBench-v2	all	$\leq$128K	all

Table 4: **Evaluation-data configuration.** "Samples" is per task. Lengths 16/32/64/128K = 16384, 32768, 65536, 131072.

Model	Mode	temp	top-p	top-k	min-p
Qwen3	thinking	0.6	0.95	20	0.0
Qwen3	non-thinking	0.7	0.80	20	0.0
Qwen3.5	non-thinking	0.7	0.80	20	0.0

Table 5: **Decoding (sampling) settings.** Evaluation uses the non-thinking presets across all four benchmarks; the Qwen3 thinking preset is listed for reference.

Model	Method	RULER	BABILong	LongBench-v2
Qwen3-4B	Vanilla	0.981 (0.011)	0.333 (0.027)	0.315 (0.049)
	Quest (Tang et al., 2024) @ $0.25L$	0.981 (0.011)	0.317 (0.026)	0.281 (0.047)
	Quest @ $0.5L$	0.981 (0.011)	0.337 (0.027)	0.292 (0.048)
	Quest @ $0.75L$	0.981 (0.011)	0.330 (0.028)	0.281 (0.048)
	Twilight (Lin et al., 2025) @ $0.25L$	0.919 (0.022)	0.317 (0.026)	0.281 (0.047)
	Twilight @ $0.5L$	0.975 (0.012)	0.307 (0.027)	0.281 (0.046)
	Twilight @ $0.75L$	0.981 (0.011)	0.317 (0.026)	0.270 (0.045)
	SpotAttention static @ $0.25L$	0.963 (0.015)	0.320 (0.027)	0.270 (0.047)
	SpotAttention static @ $0.5L$	0.969 (0.014)	0.323 (0.027)	0.281 (0.048)
	SpotAttention static @ $0.75L$	0.981 (0.011)	0.323 (0.027)	0.281 (0.046)
	SpotAttention top-p=0.7	0.931 (0.020)	0.313 (0.027)	0.270 (0.048)
	SpotAttention top-p=0.8	0.956 (0.016)	0.330 (0.028)	0.270 (0.047)
	SpotAttention top-p=0.9	0.975 (0.013)	0.323 (0.028)	0.270 (0.047)
Qwen3-8B	Vanilla	0.988 (0.009)	0.330 (0.027)	0.292 (0.048)
	Quest (Tang et al., 2024) @ $0.25L$	0.994 (0.006)	0.333 (0.027)	0.315 (0.048)
	Quest @ $0.5L$	0.988 (0.009)	0.323 (0.026)	0.326 (0.051)
	Quest @ $0.75L$	0.988 (0.009)	0.333 (0.028)	0.326 (0.049)
	Twilight (Lin et al., 2025) @ $0.25L$	0.938 (0.019)	0.320 (0.027)	0.315 (0.049)
	Twilight @ $0.5L$	0.975 (0.012)	0.303 (0.026)	0.326 (0.051)
	Twilight @ $0.75L$	0.988 (0.009)	0.323 (0.028)	0.326 (0.050)
	SpotAttention static @ $0.25L$	0.981 (0.011)	0.333 (0.027)	0.315 (0.050)
	SpotAttention static @ $0.5L$	0.988 (0.009)	0.330 (0.028)	0.315 (0.049)
	SpotAttention static @ $0.75L$	0.988 (0.009)	0.320 (0.026)	0.326 (0.051)
	SpotAttention top-p=0.7	0.931 (0.021)	0.330 (0.028)	0.315 (0.050)
	SpotAttention top-p=0.8	0.963 (0.015)	0.343 (0.028)	0.315 (0.049)
	SpotAttention top-p=0.9	0.981 (0.011)	0.350 (0.028)	0.315 (0.051)
Qwen3-32B	Vanilla	1.000 (0.000)	0.403 (0.029)	0.393 (0.054)
	Quest (Tang et al., 2024) @ $0.25L$	1.000 (0.000)	0.390 (0.028)	0.382 (0.050)
	Quest @ $0.5L$	1.000 (0.000)	0.403 (0.028)	0.393 (0.053)
	Quest @ $0.75L$	1.000 (0.000)	0.407 (0.028)	0.382 (0.050)
	Twilight (Lin et al., 2025) @ $0.25L$	0.963 (0.015)	0.363 (0.028)	0.371 (0.050)
	Twilight @ $0.5L$	0.994 (0.006)	0.383 (0.028)	0.404 (0.050)
	Twilight @ $0.75L$	1.000 (0.000)	0.403 (0.028)	0.393 (0.051)
	SpotAttention static @ $0.25L$	1.000 (0.000)	0.377 (0.028)	0.348 (0.052)
	SpotAttention static @ $0.5L$	1.000 (0.000)	0.383 (0.029)	0.360 (0.052)
	SpotAttention static @ $0.75L$	1.000 (0.000)	0.383 (0.029)	0.348 (0.050)
	SpotAttention top-p=0.7	0.838 (0.029)	0.350 (0.027)	0.360 (0.052)
	SpotAttention top-p=0.8	0.900 (0.025)	0.353 (0.027)	0.371 (0.050)
	SpotAttention top-p=0.9	0.994 (0.006)	0.363 (0.029)	0.360 (0.051)

Table 6: **Full accuracy grid: Qwen3 family.** Three Qwen3 backbones (32K context cap) across RULER, BABILong, and LongBench-v2. Static-K rows use $K \in \{0.25, 0.5, 0.75\}\, L$; dual top-$p$ rows use $p \in \{0.7, 0.8, 0.9\}$ and live below the static block on each backbone. Cells are mean accuracy (bootstrap standard error in parentheses, Appendix C.5) aggregated across length bins the cell ran at. InfiniteBench is excluded because its 64K / 128K bins exceed the Qwen3 context cap.

Model	Method	RULER	BABILong	InfiniteBench	LongBench-v2
Qwen3.5-4B	Vanilla	0.964 (0.014)	0.343 (0.028)	0.853 (0.033)	0.377 (0.030)
	SpotAttention static @ $0.25L$	0.979 (0.011)	0.347 (0.027)	0.872 (0.031)	0.366 (0.030)
	SpotAttention static @ $0.5L$	0.979 (0.010)	0.363 (0.027)	0.874 (0.030)	0.366 (0.028)
	SpotAttention static @ $0.75L$	0.979 (0.010)	0.340 (0.028)	0.844 (0.035)	0.351 (0.029)
	SpotAttention top-p=0.7	0.979 (0.010)	0.337 (0.027)	0.860 (0.033)	0.373 (0.030)
	SpotAttention top-p=0.8	0.979 (0.010)	0.350 (0.028)	0.873 (0.031)	0.380 (0.029)
	SpotAttention top-p=0.9	0.979 (0.011)	0.370 (0.029)	0.859 (0.033)	0.366 (0.028)
Qwen3.5-9B	Vanilla	0.948 (0.016)	0.310 (0.027)	0.866 (0.032)	0.417 (0.030)
	SpotAttention static @ $0.25L$	0.995 (0.005)	0.303 (0.027)	0.883 (0.029)	0.395 (0.029)
	SpotAttention static @ $0.5L$	0.995 (0.005)	0.303 (0.027)	0.883 (0.030)	0.399 (0.031)
	SpotAttention static @ $0.75L$	0.995 (0.005)	0.303 (0.027)	0.883 (0.030)	0.399 (0.029)
	SpotAttention top-p=0.7	0.995 (0.005)	0.337 (0.027)	0.866 (0.033)	0.388 (0.029)
	SpotAttention top-p=0.8	0.990 (0.008)	0.357 (0.028)	0.894 (0.028)	0.391 (0.030)
	SpotAttention top-p=0.9	0.995 (0.006)	0.303 (0.026)	0.883 (0.029)	0.399 (0.029)

Table 7: **Full accuracy grid: Qwen3.5 family.** Two hybrid Qwen3.5 backbones (128K native context) across the four long-context benchmarks. Static-K rows use $K \in \{0.25, 0.5, 0.75\}\, L$; dual top-$p$ rows use $p \in \{0.7, 0.8, 0.9\}$. Cells are mean accuracy (bootstrap standard error in parentheses, Appendix C.5) aggregated across length bins the cell ran at. Quest and Twilight have not yet been run on this family; the Qwen3 family table is the only complete head-to-head against training-free baselines.

Task	Vanilla	K=0.5L	p=0.9
RULER			
niah_multikey_1	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
niah_multikey_2	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
niah_multikey_3	0.957 (0.043)	0.870 (0.070)	0.913 (0.060)
niah_single_1	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
niah_single_2	1.000 (0.000)	1.000 (0.000)	1.000 (0.000)
niah_single_3	0.939 (0.041)	0.939 (0.042)	0.939 (0.040)
BABILong			
babilong_qa1	0.680 (0.047)	0.710 (0.045)	0.700 (0.046)
babilong_qa2	0.170 (0.037)	0.180 (0.040)	0.180 (0.039)
babilong_qa3	0.150 (0.037)	0.080 (0.028)	0.090 (0.029)
LongBench-v2			
Code Repository	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Long ICL	0.200 (0.173)	0.200 (0.176)	0.200 (0.181)
Long Structured Data	0.000 (0.000)	0.000 (0.000)	0.000 (0.000)
Long-dialogue Hist.	0.250 (0.122)	0.167 (0.106)	0.083 (0.076)
Multi-Doc QA	0.368 (0.113)	0.316 (0.107)	0.316 (0.107)
Single-Doc QA	0.347 (0.065)	0.327 (0.068)	0.327 (0.069)

Table 8: **Per-task accuracy on Qwen3-4B.** Vanilla against SpotAttention static K=0.5L and dual top-p=0.9 on every subtask of RULER, BABILong, and LongBench-v2; mean over all length bins the model ran at, with bootstrap standard error in parentheses (Appendix C.5).

Backbone	Precision	RULER	BABILong	LongBench-v2	Mean
Qwen3-4B	BF16	0.975 (0.013)	0.323 (0.028)	0.270 (0.047)	0.523 (0.019)
	INT8	0.975 (0.013)	0.317 (0.028)	0.270 (0.046)	0.520 (0.018)
	FP8	0.975 (0.013)	0.317 (0.026)	0.270 (0.047)	0.520 (0.018)
	INT4	0.969 (0.013)	0.317 (0.026)	0.270 (0.048)	0.518 (0.019)
	FP4	0.975 (0.013)	0.320 (0.026)	0.270 (0.047)	0.522 (0.018)
Qwen3-8B	BF16	0.981 (0.011)	0.350 (0.028)	0.315 (0.051)	0.549 (0.020)
	INT8	0.975 (0.013)	0.343 (0.027)	0.315 (0.049)	0.544 (0.019)
	FP8	0.975 (0.013)	0.340 (0.027)	0.315 (0.051)	0.543 (0.020)
	INT4	0.981 (0.011)	0.327 (0.028)	0.315 (0.049)	0.541 (0.019)
	FP4	0.981 (0.010)	0.340 (0.026)	0.315 (0.049)	0.545 (0.019)
Qwen3-32B	BF16	0.994 (0.006)	0.363 (0.029)	0.360 (0.051)	0.572 (0.020)
	INT8	0.994 (0.006)	0.390 (0.027)	0.360 (0.050)	0.581 (0.019)
	FP8	0.994 (0.006)	0.373 (0.028)	0.360 (0.050)	0.576 (0.019)
	INT4	0.994 (0.006)	0.397 (0.027)	0.360 (0.050)	0.583 (0.019)
	FP4	0.988 (0.009)	0.403 (0.028)	0.371 (0.053)	0.587 (0.020)

Table 9: **Indexer quantization across the Qwen3 family.** Top-p=0.9 accuracy with the indexer's queries and key cache stored at five precisions, on Qwen3-4B, -8B, and -32B. Cells are mean accuracy (bootstrap standard error in parentheses, Appendix C.5). Every quantized variant stays within bootstrap error of the BF16 reference on the dataset-mean column at every backbone. Indexer K-cache footprint per precision: INT8 / FP8 store at $0.500\times$ BF16; INT4 / FP4 use group-32 microscale at 4.5 bits per element ($0.281\times$). The backbone KV cache is unchanged; the indexer learns under BF16 and is quantized at inference only.