Prism Attention: Proposal-Refined Index Sharing Mechanism for Efficient LLMs Inference

Zhenxu Tian[1,2 *] Kebin Liu[2 * †] Zhengwu Yang[2] Yi Su[1] Qingqing Dang[2]
Yanlin Sha[2] Kaipeng Deng[2] Dianhai Yu[2] Yanjun Ma[2 ‡] Juntao Li[1 ‡] Min Zhang[1]
[1]Soochow University [2]Baidu Inc.
zhenxutnlp@outlook.com liukebin@baidu.com

Abstract

Long-context decoding in large language models (LLMs) is increasingly bottlenecked by the memory and computation required to attend over an ever-growing Key-Value (KV) cache at each autoregressive step. Although token-level sparse attention reduces this burden by restricting computation to a small subset of relevant past tokens, its practical gains are often offset by repeatedly recomputing layer-wise sparse indices as the context grows. A common remedy is to share sparse indices across layers by computing them only at designated anchor layers, but such static reuse ignores the layer-wise evolution of attention patterns, causing the reused indices to progressively deviate from the layer-specific optima and incur substantial attention loss. By revisiting cross-layer sparsity from the perspective of attention dynamics, we find that **inter-layer salient token shift is highly localized**: although the exact top-κ indices differ across layers, the core high-attention tokens remain stable, with changes concentrated near the expanded boundary of previously selected regions. Motivated by this, we propose **Prism Attention**, a proposal-refined sparse attention framework. It replaces rigid cross-layer index reuse with a coarse-to-fine strategy, preserving layer-wise adaptability while retaining the efficiency benefits of cross-layer sharing. Extensive experiments show that Prism Attention achieves state-of-the-art performance among mainstream sparse attention methods on both reasoning and long-context benchmarks, while attaining a $3.4\times$ end-to-end decoding speedup over FlashAttention at 128K context length.

1 Introduction

As large language models (LLMs) [2, 10, 11, 15, 33] are pushed toward longer-context reasoning, autoregressive decoding increasingly becomes bottlenecked by attention over the growing Key-Value (KV) cache. At each decoding step, the newly generated token must attend over the entire accumulated context, causing the attention cost to scale linearly with sequence length. Sparse attention mitigates this by restricting computation to a small subset of relevant past tokens. Among existing paradigms, token-level [19, 38] sparsity achieves stronger empirical performance

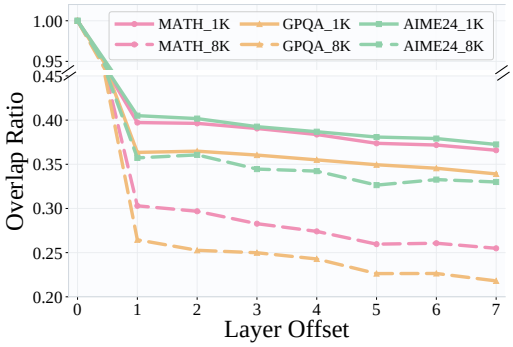
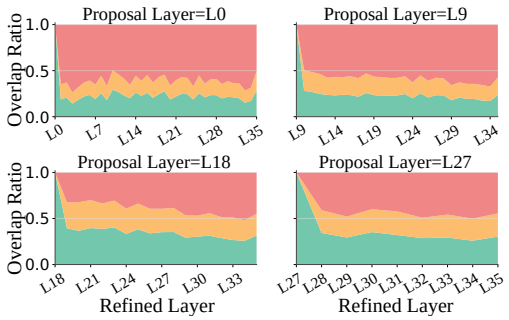


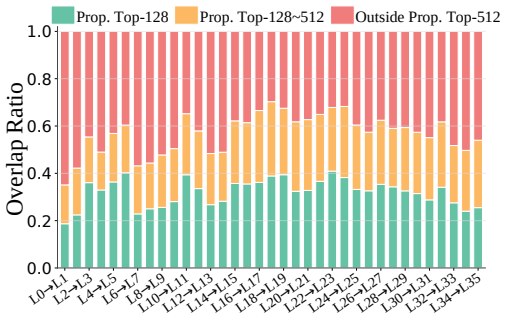
Figure 1: Cross-layer overlap ratio across varying distances of different layer offset.

*Equal contribution.
†Project lead. ‡Corresponding author.
Work done during an internship at Baidu Inc.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(a) Layer-wise evolution of refined-layer top-128 indices within expanded proposal regions.



(b) Adjacent-layer distribution of next-layer top-128 indices within preceding-layer proposal regions.

Figure 2: **Localized evolution of sparse indices across layers within proposal regions.** For each target (refined) layer, we categorize its own top-128 sparse indices according to their ranks in the corresponding reference (proposal) layer: "Prop. Top-128", "Prop. Top-128∼512", and "Outside Prop. Top-512" denote the ratios of target-layer top-128 indices falling in the reference-layer top-128 set, ranks 128∼511, and outside the top-512 set, respectively. The above two panels share the same legend. (a) tracks this distribution from proposal layers L_0, L_9, L_{18}, and L_{27} to their subsequent refined layers, showing that a slightly expanded proposal pool ($\rho = 4$, i.e., $4\times$ the base size) consistently captures a significant majority of layer-wise sparse-index variations. (b) measures, for each adjacent pair $L_\ell \to L_{\ell+1}$, how the top-128 indices of $L_{\ell+1}$ fall into the proposal regions of L_ℓ. This demonstrates that adjacent-layer shifts are heavily concentrated within the expanded proposal window, significantly reducing the risk of missing salient tokens while avoiding the prohibitive cost of full-context search. A more comprehensive analysis is provided in Appendix B.

than coarse-grained block-wise alternatives [8, 9, 17, 18, 21, 25, 26, 30, 32, 37]. However, its practical efficiency is often constrained by sparse index construction: identifying the exact top-κ relevant tokens for each query still requires scoring the entire context. As sequence length increases, repeatedly recomputing these layer-wise indices introduces substantial overhead, which can largely offset the theoretical benefits of sparsity.

A natural strategy to circumvent the per-layer selection overhead is cross-layer index sharing, which computes the exact top-κ token indices at a few designated anchor layers and reuses them across multiple subsequent layers. While effective in reducing redundant computation, existing approaches [1, 3, 5, 12, 34, 35] adopt a static sharing paradigm that ignores inter-layer differences by implicitly assuming attention patterns remain invariant with network depth. In reality, attention distributions evolve dynamically across layers. As the distance from the designated anchor layer increases in Figure 1, the index overlap ratio between the anchor layer and subsequent layers drops sharply and continues to decay. Notably, this degradation accelerates under longer contexts (8K vs. 1K), causing static reuse to repeatedly miss critical tokens and severely degrade long-context performance.

To better understand this limitation, we revisit cross-layer sparsity from the perspective of *attention dynamics*. Our empirical analysis reveals a key property: **inter-layer salient token shift is highly localized**. As illustrated in Figure 2a and 2b, although the exact top-κ indices differ across layers, these changes are far from arbitrary: the core high-attention tokens remain largely stable, while most changes occur near the boundary of previously selected regions. This indicates that preserving a slightly expanded candidate pool is sufficient to capture layer-wise variations.

Motivated by this observation, we propose **Prism Attention**, a proposal-refined sparse attention framework for efficient long-context decoding, driven by two parallel algorithmic components: **Dynamic Index Refinement** and **Quantized Proposal Generation**. Instead of rigidly reusing a fixed set of indices, Prism Attention first constructs a relaxed candidate pool at a few designated anchor layers (*Proposal Layer*) and then allows subsequent layers (*Refined Layer*) to perform lightweight token selection within this expanded boundary, thereby adapting to localized inter-layer shifts at low cost. To further reduce the overhead of proposal construction, Prism Attention employs a quantized proposal mechanism that substantially lowers the cost of full-context scoring. Together, these designs enable efficient cross-layer sparsity while preserving attention quality. Extensive experiments show that Prism Attention achieves state-of-the-art performance among mainstream sparse attention methods on both reasoning and long-context benchmarks, while attaining a $3.4\times$ end-to-end decoding speedup over FlashAttention at 128K context length.

Our contributions are as follows:

- Revisiting cross-layer sparse attention from the perspective of *attention dynamics* reveals a key property: inter-layer salient token shift is highly localized. This observation reveals the limitation of rigid index sharing and motivates localized cross-layer refinement.
- We propose **Prism Attention**, a proposal-refined sparse attention framework for efficient long-context decoding, which replaces static cross-layer index sharing with a coarse-to-fine selection strategy, enabling efficient adaptation to layer-wise attention variation.
- Two complementary algorithmic components make this design practical: **Dynamic Index Refinement**, which tracks localized layer-wise token shifts with low overhead, and **Quantized Proposal Generation**, which reduces the cost of full-context proposal construction via low-bit quantization.

2 Related Work

2.1 Sparse Attention

To mitigate the prohibitive quadratic complexity of standard self-attention, particularly during long-context inference, existing sparse mechanisms primarily explore two distinct paradigms: block-level and token-level sparsity. Block-level methods exploit coarse-grained activation patterns and are predominantly tailored to specific inference phases. For instance, works like [7, 9, 16, 17, 18, 20, 26, 29, 30, 32, 39] dynamically filter critical blocks to accelerate the computation-heavy prefill phase, whereas approaches such as [8, 24, 25, 36] exclusively optimize the decoding stage by alleviating memory bandwidth bottlenecks associated with KV cache loading. More recently, unified approaches like [14, 21, 27, 31, 37, 40] have leveraged end-to-end sparse training to accelerate both prefill and decoding phases, while mitigating the accuracy degradation typically associated with block-level sparsity. In contrast, token-level sparsity operates at a strictly finer granularity to preserve critical context that might be discarded by rigid block boundaries. DeepSeek Sparse Attention (DSA) [19] pioneers the large-scale production deployment of this paradigm, demonstrating its practical efficacy, subsequently validated and adopted by recent models like GLM5 [38]. However, recomputing fine-grained token indices at every layer repeatedly introduces substantial computational overhead, an inefficiency that motivates the exploration of cross-layer redundancy by Prism Attention.

2.2 Cross-Layer Index Sharing

Recent studies have revealed significant inter-layer correlation in attention activation patterns, motivating cross-layer index sharing techniques. To exploit this redundancy during sparse decoding, methods such as TidalDecode [34], LessIsMore [35], and OmniKV [12] directly propagate the top-κ indices from designated anchor layers to subsequent layers. To optimize anchor placement, Kascade [5] and HyLRA [1] frame anchor selection as a dynamic programming problem to discover optimal layer configurations. To mitigate the compounding effect of error propagation inherent in this process, IndexCache [3] employs a greedy search strategy to identify more robust anchor placements. Despite these advances, existing cross-layer index sharing remains constrained by a trade-off between anchor-layer cost and reuse-layer accuracy. Increasing the number of anchor layers provides fresher sparse support but raises index-construction overhead, while sharing indices across more reuse layers reduces overhead but causes reused indices to drift from layer-specific optima.

3 Observations

Motivated by the limitations of static cross-layer sharing discussed in Section 1, we conduct an empirical analysis of how sparse attention indices evolve across layers. Figure 1 indicates that, as the distance from the anchor layer increases, the index overlap ratio shifts progressively across depth instead of remaining stationary. As a result, the rigid and non-adaptive nature of static reuse fails to track natural inter-layer drift, causing the reused indices to systematically deviate from the true layer-wise optimum, resulting in severe information loss during autoregressive decoding.

More importantly, we observe that: **although directly reused indices progressively deviate from the layer-wise optimum, the resulting salient token shift is highly localized rather than arbitrary**. As shown in Figure 2, high-attention tokens remain stable across adjacent layers, while newly selected tokens beyond the original proposal top-128 are mostly captured by the expanded proposal region

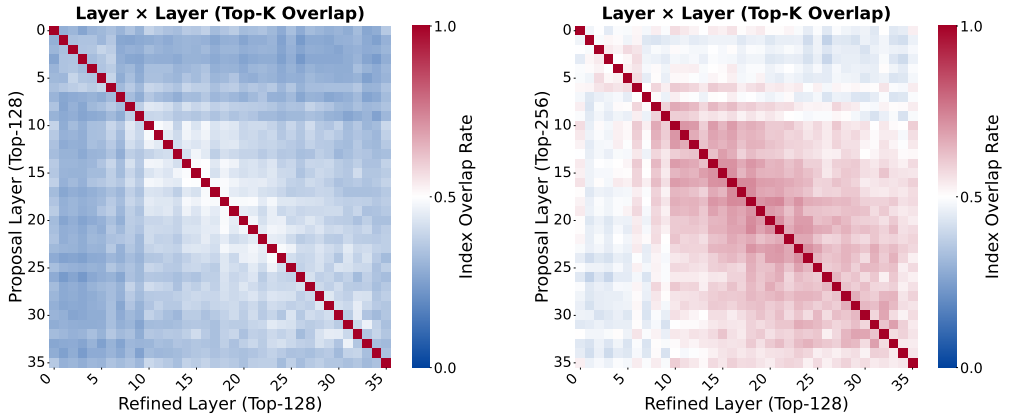


Figure 3: **Effect of expanded proposal regions on cross-layer sparse-index coverage.** We conduct the analysis on Qwen3-4B using decoding traces collected from AIME24, and report the average cross-layer coverage of refined-layer top-128 sparse indices by the reference-layer proposal set. **Left:** the reference proposal is restricted to its top-128 indices, corresponding to rigid top-κ index reuse by the refined layer. **Right:** the reference proposal is expanded from top-128 to top-256 ($\rho = 2$), doubling the candidate pool used for refinement while keeping the refined-layer sparse support fixed at top-128. Compared with rigid top-128 reuse, the expanded proposal region preserves substantially higher coverage across layers, showing that many refined-layer salient tokens missed by rigid reuse are still captured within a moderately enlarged candidate window.

between the original and expanded boundaries rather than scattered across the full context. This indicates that most inter-layer variation mainly arises from small local adjustments around previously selected tokens, rather than from a complete reorganization of attention.

This localized structure suggests that the main limitation of static reuse is not the act of reuse itself, but the rigidity of reusing an overly compact top-κ set. Since inter-layer variation is concentrated around previously selected tokens, a slightly expanded candidate pool from the anchor layer can already retain high coverage of the salient tokens required by subsequent layers. Figure 3 further shows that a moderately expanded proposal region can recover much of the lost cross-layer coverage, validating the effectiveness of using an enlarged candidate pool for refinement.

4 Method

Overview. Motivated by the localized inter-layer drift observed in Section 3, Prism Attention replaces rigid layer-wise inheritance of a fixed top-κ set with a coarse-to-fine cross-layer scheme (Figure 4): designated anchor layers (*Proposal Layer*) generate an expanded candidate pool, and downstream shared layers (*Refined Layer*) refine their final sparse support within this transmitted proposal region. To make proposal generation itself efficient, Prism further compresses the key cache used at proposal layers through an INT2-based construction scheme. Prism inherits the assignment of Proposal/Refined layers from the original offline IndexCache [3] calibration procedure, while redesigning the information transmitted by proposal layers and the way it is consumed by refined layers. Together, these designs make token-level sparse attention adaptive to layer-wise variation while retaining the computational advantages of cross-layer reuse.

4.1 Proposal-Refined Index Sharing

Consider autoregressive decoding at step t in transformer layer ℓ, with query representation q_t^ℓ and historical key cache $K^\ell = \{k_1^\ell, k_2^\ell, \ldots, k_{L_t}^\ell\}$, where L_t denotes the current context length. Let

$$s_t^\ell(i) = \phi\big(q_t^\ell, k_i^\ell\big), \qquad i \in [L_t], \tag{1}$$

be the relevance score between the current query and the i-th historical key, where $\phi(\cdot,\cdot)$ denotes the sparse scoring function used for token selection, as mentioned in Equation 9. The ideal layer-specific sparse support is defined as

$$\mathcal{T}_t^\ell = \mathrm{TopK}\left(s_t^\ell, \kappa\right), \tag{2}$$

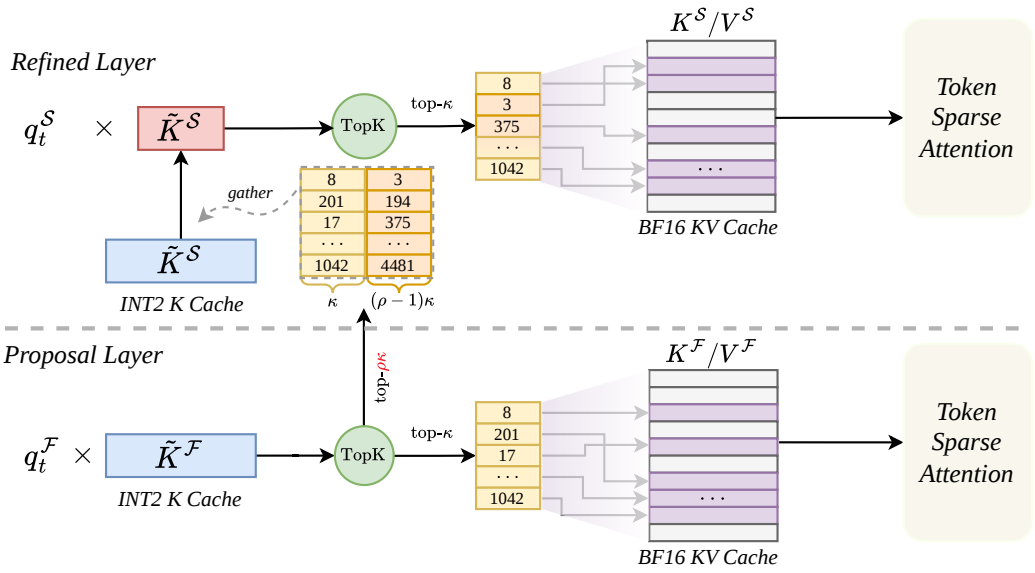


Figure 4: An overview of Prism Attention. Algorithm 1 presents the per-step decoding procedure.

where $\mathrm{TopK}(\cdot)$ returns the indices of the top-κ largest entries. Computing $\mathcal{T}_t^{\ell}$ exactly requires scoring against all L_t cached tokens, incurring $\mathcal{O}(L_t)$ selection cost per layer.

We partition layers into a set of proposal layers and a set of refined layers. For each refined layer $\mathcal{S}$, let $f(\mathcal{S}) \in \mathcal{F}$ denote its associated preceding proposal layer. Conventional static reuse directly copies the selected sparse support from the proposal layer:

$$\mathcal{T}_t^{\mathcal{S}} \leftarrow \mathcal{T}_t^{\mathcal{F}}, \tag{3}$$

but this strategy assumes that the exact identity of salient tokens remains unchanged across layers. In practice, the layer-wise optimum evolves with depth, and the mismatch between reused indices and true salient tokens accumulates as the distance from the proposal layer increases.

Prism replaces exact support sharing with *proposal-refined index sharing*. Instead of exporting only a top-κ set, the proposal layer $\mathcal{F}$ produces an expanded candidate pool

$$\mathcal{C}_t^{\mathcal{F}} = \mathrm{TopK}\left(s_t^{\mathcal{F}}, n\right), \qquad n = \rho\kappa,\ \rho \geq 1, \tag{4}$$

where ρ is a small expansion factor and n is the size of candidate pool. The term $s_t^{\mathcal{F}}$ denotes the proposal score used to construct the candidate pool. The proposal layer itself still attends to its own top-κ tokens, while caching $\mathcal{C}_t^{\mathcal{F}}$ for downstream reuse.

Subsequently, for the refined layer $\mathcal{S}$, Prism avoids full-context search and refines the inherited proposal locally. Specifically, it recomputes scores only within $\mathcal{C}_t^{\mathcal{F}}$:

$$\widehat{s}_t^{\mathcal{S}}(i) = \phi\left(q_t^{\mathcal{S}}, k_i^{\mathcal{S}}\right), \qquad i \in \mathcal{C}_t^{\mathcal{F}}, \tag{5}$$

and then selects its final sparse support as

$$\widehat{\mathcal{T}}_t^{\mathcal{S}} = \mathrm{TopK}\left(\widehat{s}_t^{\mathcal{S}}, \kappa\right). \tag{6}$$

This design transforms cross-layer reuse from *rigid support sharing* into *proposal-constrained local search*: $\mathcal{C}_t^{\mathcal{F}}$ provides a high-recall candidate region to the layer-wise optimum, and the downstream refinement step absorbs the localized inter-layer deviations.

4.2 Quantized Proposal Construction

While proposal-refined index sharing restricts full-context scoring to only a few proposal layers, proposal construction remains bandwidth-intensive during long-context decoding, since each proposal layer must repeatedly access a large historical key cache. Prism addresses this bottleneck by compressing the key cache to INT2 for candidate scoring.

Specifically, we quantize historical keys using group-wise asymmetric b-bit quantization with $b = 2$. To implement this group-wise strategy, we partition the head dimension d of each full-precision key vector $k_i^{\mathcal{F}}$ into $M = d/g$ independent sub-vectors of size g. The original key vector can thus be denoted as the concatenation of these sub-vectors:

$$k_i^{\mathcal{F}} = [k_{i,1}^{\mathcal{F}} \parallel k_{i,2}^{\mathcal{F}} \parallel \cdots \parallel k_{i,M}^{\mathcal{F}}] \tag{7}$$

where $k_{i,m}^{\mathcal{F}} \in \mathbb{R}^g$ represents the m-th group. Under the asymmetric quantization scheme, each group $k_{i,m}^{\mathcal{F}}$ maintains its own specific scaling factor $\alpha_{i,m}^{\mathcal{F}} = (\max(k_{i,m}^{\mathcal{F}}) - \min(k_{i,m}^{\mathcal{F}}))/(2^b - 1)$ and zero-point $\beta_{i,m}^{\mathcal{F}} = \min(k_{i,m}^{\mathcal{F}})$. The quantized 2-bit integer representation $\tilde{k}_{i,m}^{\mathcal{F}}$ for the m-th group is formally defined as:

$$\tilde{k}_{i,m}^{\mathcal{F}} = \text{clip}\left(\left\lfloor \frac{k_{i,m}^{\mathcal{F}} - \beta_{i,m}^{\mathcal{F}}}{\alpha_{i,m}^{\mathcal{F}}} \right\rceil, 0, 2^b - 1\right) \tag{8}$$

where $\lfloor \cdot \rceil$ denotes rounding to the nearest integer. Consequently, the full quantized key vector $\tilde{k}_i^{\mathcal{F}}$ is formulated by concatenating these quantized sub-vectors: $\tilde{k}_i^{\mathcal{F}} = [\tilde{k}_{i,1}^{\mathcal{F}} \parallel \tilde{k}_{i,2}^{\mathcal{F}} \parallel \cdots \parallel \tilde{k}_{i,M}^{\mathcal{F}}]$.

During the autoregressive decoding step t, rather than relying on the memory-intensive full-precision cache, we compute approximate proposal scores utilizing the quantized 2-bit keys. Specifically, under the Grouped Query Attention (GQA) architecture, let $q_{t,j}^{\mathcal{F}}$ denote the j-th query head assigned to a shared KV group $\mathcal{G}$. We define the aggregated proposal score for the i-th historical token as:

$$\tilde{s}_t^{\mathcal{F}}(i) = \max_{j \in \mathcal{G}} \left\langle q_{t,j}^{\mathcal{F}}, \tilde{k}_i^{\mathcal{F}} \right\rangle, \qquad i \in [L_t]. \tag{9}$$

The max-reduction aggregates multiple query heads into one score per KV head, so each KV head is loaded only once per decoding step, avoiding redundant memory accesses across grouped heads. Finally, selecting the top-$\rho\kappa$ scores yields the optimal expanded candidate pool $\mathcal{C}_t^{\mathcal{F}}$.

5 Experiment

5.1 Experimental Setup

Models and Benchmarks. Our evaluation is based on Qwen3-4B and Qwen3-8B [33], both of which are specifically trained for reasoning tasks and perform the thinking process by generating extensive tokens. For long-context understanding, we use LongBench [4] and select the subsets whose average input length exceeds 8192 tokens to focus on genuinely long-input scenarios, capping the maximum generation length at 4096 tokens. For reasoning evaluation, we generate 64, 8, and 16 answers for each problem in the AIME-24/25 [22], MATH500 [13], GPQA-Diamond [23] datasets and compute the Pass@1 accuracy, with the maximum generation length set to 32768 tokens.

Baselines. We compare Prism Attention against standard full attention and four representative sparse decoding baselines: Quest [25], LessIsMore (LIM) [35], SeerAttention-R (Seer-R) [8], and IndexCache (IDC) [3]. **Quest** exploits query-aware sparsity by estimating the criticality of each KV cache page from the current query and per-page key extrema, and then attends only to the top-κ selected pages. **LessIsMore** merges per-head importance rankings at a single selection layer into one cross-head token set, which is then unioned with a fixed-ratio recency window and reused by subsequent sparse layers. **Seer-R** learns block-level attention sparsity through a self-distilled attention gate, enabling query-aware sparse block selection. **IndexCache** retains indexers only at a few anchor layers whose top-κ indices are reused by the consecutive shared layers that follow.

Implementation Details. All experiments are conducted on NVIDIA H800 GPUs. For Quest, we use its default block size of 32. For Seer-R, we follow its original configuration, where the block size is set to 64 and all layers perform sparse attention. For IndexCache, we adapt the training-free offline calibration from GLM5 [38] to Qwen3. Specifically, we treat each transformer layer as either an *anchor* layer or a *shared* layer at a 1:3 ratio, and determine the layer assignment by greedy calibration on a held-out subset of OpenR1-MATH-220k [6]. In our implementation, calibration uses 64 samples with a maximum sequence length of 32768 and sets the sparse token budget to 512. Following the loss-based search strategy of IndexCache, we use token-level cross-entropy as the calibration objective. The resulting partition is fixed for all downstream evaluations. For Prism Attention,

Table 1: Accuracy comparison on reasoning benchmarks using Qwen3-4B and Qwen3-8B.

LLMs	Methods	AIME24	AIME25	GPQA	MATH500	Avg.
Qwen3-4B	Full Attention	71.25	66.41	56.19	93.93	71.95
	Quest	46.67	39.79	24.74	73.40	46.15
	Seer-R	69.32	58.59	55.40	94.10	69.35
	LessIsMore	71.48	62.87	**56.23**	**94.12**	71.18
	IndexCache	70.99	61.30	54.29	94.05	70.16
	Prism	**72.60**	**64.27**	54.83	94.00	**71.43**
Qwen3-8B	Full Attention	74.48	67.86	60.54	94.43	74.33
	Quest	59.79	57.71	47.22	87.80	63.13
	Seer-R	71.35	57.81	60.48	94.00	70.91
	LessIsMore	73.00	65.24	**60.65**	**94.55**	73.36
	IndexCache	74.84	64.47	60.39	94.40	73.53
	Prism	**75.05**	**65.52**	60.45	94.30	**73.83**

Table 2: Performance comparison of different methods across various LLMs on LongBench.

	Methods	2WikiMQA	TriviaQA	HotpotQA	TREC	Qasper	PCount	MF-en	PRe	GovReport	Avg.
Qwen3-4B	Full Attention	62.68	89.69	69.79	45.00	29.35	22.26	49.16	99.06	28.66	55.07
	Seer-R	59.07	**90.42**	69.59	48.00	19.19	**24.36**	**50.49**	**100.00**	28.19	54.37
	LessIsMore	**62.57**	82.62	**70.73**	41.00	21.43	16.67	49.83	98.00	28.42	52.36
	IndexCache	57.73	87.28	70.64	37.00	**26.92**	22.12	49.16	**100.00**	28.54	53.27
	Prism	60.36	89.76	70.16	**52.00**	23.89	20.35	49.75	**100.00**	**28.57**	**54.98**
Qwen3-8B	Full Attention	64.77	89.02	70.45	31.33	31.52	24.95	50.54	100.00	28.66	54.58
	Seer-R	**66.17**	89.59	68.58	34.00	27.53	21.67	46.51	**100.00**	28.70	53.64
	LessIsMore	58.78	89.32	68.63	31.00	24.74	21.17	37.25	**100.00**	27.93	50.98
	IndexCache	60.50	**91.49**	**71.46**	27.00	**30.58**	21.19	43.45	99.12	**29.07**	52.65
	Prism	65.19	91.19	70.61	**34.03**	25.67	**27.11**	**52.12**	**100.00**	28.81	**54.97**

we adopt the same offline calibration pipeline as IndexCache to determine the Proposal/Refined partition and set the final sparse budget to $\kappa = 2048$, with a proposal expansion factor of $\rho = 4$ for long-context tasks ($n = \rho\kappa = 8192$) and $\rho = 2$ for reasoning tasks ($n = 4096$).

5.2 Accuracy Results

Performance on Reasoning Benchmarks. As shown in Table 1, Prism achieves the best average accuracy among all evaluated sparse baselines on both Qwen3 models. On Qwen3-4B, Prism reaches an average score of 71.43, improving over Quest, Seer-R, and IndexCache by 25.28, 2.08, and 1.27 points, respectively. This advantage transfers to Qwen3-8B, where Prism narrows the gap with Full Attention to 0.50 points (73.83 vs. 74.33). Notably, the improvements are most pronounced on the two most challenging subsets: AIME24 and AIME25, where the average gain reaches 1.46 points over IndexCache and 1.21 points over LessIsMore. These show that proposal-refined index sharing improves performance on difficult long-reasoning tasks, without degrading general reasoning accuracy.

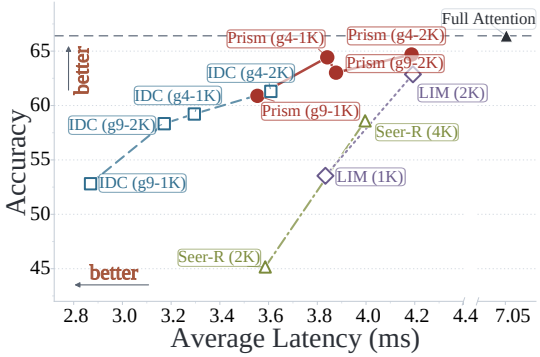


Figure 5: Accuracy-Efficiency Pareto Comparison.

Accuracy on Long-Context Understanding. As reported in Table 2, Prism achieves the strongest average performance among all sparse attention methods. On Qwen3-4B, Prism obtains an average score of 54.98, nearly matching Full Attention and improving over IndexCache by 1.71 points. On Qwen3-8B, Prism reaches 54.97, slightly exceeding Full Attention and improving over Seer-R and IndexCache by 1.33 and 2.32 points, respectively. These results support our hypothesis that rigid cross-layer index reuse accumulates attention loss as reused indices drift from layer-wise optima, whereas Prism allows subsequent refined layers to recover their own salient tokens within the proposal region, retaining task-relevant context that a compact top-κ set would miss.

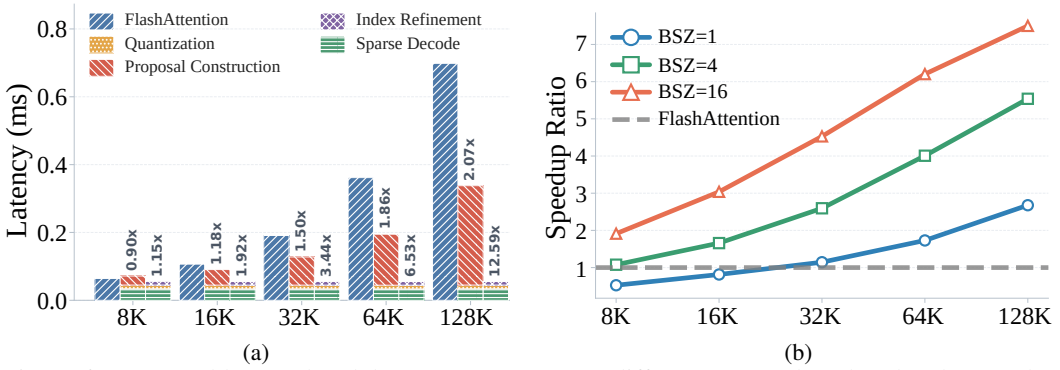


Figure 6: (a) Kernel latency breakdown at BSZ=4 across different context lengths. (b) The speedup of the average per-layer kernel latency over FlashAttention across context lengths and batch sizes.

5.3 Efficiency Analysis

We analyze efficiency at three levels: end-to-end serving latency, attention kernel latency, and the resulting accuracy-efficiency trade-off.

End-to-end latency. We measure end-to-end decoding latency on Qwen3-4B with SGLang under a sparse budget of 2048. As shown in Figure 7, Prism reduces the per-step latency of FlashAttention from 15.03 ms to 4.47 ms at 128K (BSZ=16), a 3.36× speedup that increases further with context length. More importantly, the latency overhead of Prism-g4 over IndexCache-g4—incurred by the additional refinement at every refined layer—decreases monotonically as the context grows: Prism is 15.7% slower at 16K, 8.0% at 64K, and only 0.7% at 128K (4.467 ms vs. 4.434 ms). Refinement scans only the inherited candidate pool, so its cost is independent of context length. These results demonstrate that the lightweight refinement overhead is increasingly amortized in long-context serving, enabling substantially stronger layer-wise adaptability with only marginal additional latency.

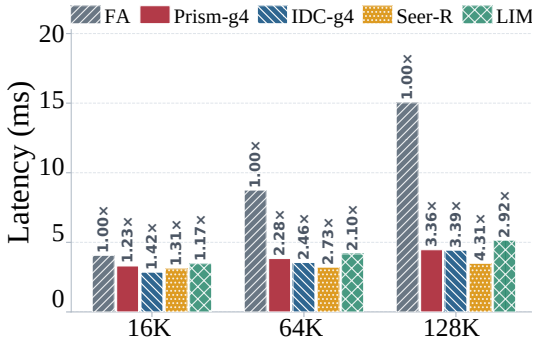


Figure 7: End-to-end latency comparison across different context lengths on Qwen3-4B at BSZ=16.

Kernel-level analysis. We evaluate the efficiency of Prism using our customized Triton [28] decoding kernels by CUDA graph execution time ($\kappa = 1024$, $\rho = 4$). For proposal layers, the latency consists of key-cache 2bit quantization, proposal construction, and sparse decoding, i.e.,

$$T_{\text{proposal}} = T_{\text{INT2Decode}} + T_{\text{ProposalConstruction}} + T_{\text{SparseDecode}},$$

while refined layers replace full proposal construction with lightweight local index refinement:

$$T_{\text{refine}} = T_{\text{INT2Decode}} + T_{\text{IndexRefinement}} + T_{\text{SparseDecode}},$$

giving an effective per-layer latency of $T = (T_{\text{proposal}} + 3T_{\text{refine}})/4$ under the calibrated 1:3 ratio. Figure 6a shows that the two layer types differ almost only in the index stage: at BSZ=4, proposal construction grows linearly with context, while index refinement remains constant at about 0.011 ms, as do INT2 Quantization and sparse decoding. A proposal layer is 2.07× faster than FlashAttention at 128K whereas a refined layer reaches 12.59×, with a latency that is independent of context length. Since three out of every four layers are refined, the costly proposal construction is amortized in the per-layer latency T: Figure 6b shows that its speedup over FlashAttention grows with both context length and batch size, reaching 7.50× at 128K with BSZ=16.

Accuracy-efficiency trade-off. We provide an end-to-end accuracy-efficiency Pareto comparison using SGLang, where each point represents the average per-token decoding latency over the corresponding average generation length. We evaluate Prism against representative sparse baselines

Table 3: Ablation study of index construction and proposal-refined sharing on Qwen3-4B. Group x denotes that one in every x layers acts as a proposal layer (i.e., a proposal-to-refined ratio of $1{:}(x-1)$). All configurations use a sparse budget of $\kappa = 1024$; in proposal-refined configurations, each refined layer selects its final κ tokens from a candidate pool of $n = \rho\kappa$ with $\rho = 4$.

Per-Layer Indices	Reuse Indices	Quantization	Group	Calibration	Proposal-Refined	Qwen3-4B	
						AIME24	AIME25
✓						71.35	63.87
✓		✓				72.08	63.26
	✓	✓	4			65.68	50.05
	✓	✓	4	✓		72.19	59.22
	✓	✓	4		✓	71.56	62.60
	✓	✓	4	✓	✓	**72.03**	**64.43**
	✓	✓	9			61.88	43.33
	✓	✓	9	✓		67.66	52.81
	✓	✓	9		✓	69.58	59.01
	✓	✓	9	✓	✓	70.83	60.89

on AIME 25 with Qwen3-4B under different sparse budgets. The notation gx denotes that one in every x layers acts as a proposal layer, while 1K/2K denote the final sparse budget κ. As shown in Figure 5, Prism occupies the high-accuracy end of the Pareto frontier: Prism-g4-1K reaches 64.43 accuracy at 3.84 ms, $1.84\times$ faster than Full Attention while retaining 97.0% of its accuracy, and strictly dominates both LIM-2K (62.87 at 4.19 ms) and Seer-R-4K (58.59 at 4.00 ms); IDC-g4-1K is 0.55 ms faster but 5.21 points less accurate. The results show that Prism achieves a favorable accuracy-efficiency trade-off across different sparse budgets and Proposal/Refined configurations.

5.4 Ablation

We conduct ablation studies on Qwen3-4B to isolate the effect of each design component. As shown in Table 3, directly reusing sparse indices across layers leads to a clear accuracy drop, especially on AIME25 ($63.87 \rightarrow 50.05$), indicating that rigid cross-layer sharing fails to track layer-wise attention variations. Offline calibration substantially improves the reuse baseline by selecting more suitable proposal/refined layer assignments, while proposal-refined sharing further recovers the lost accuracy by allowing each refined layer to perform local token selection within the inherited candidate region. Combining calibration and proposal-refined sharing gives the best overall result, improving AIME25 from 50.05 to 64.43 under the group-4 setting (i.e., 1:3 ratio of proposal to refined layers). The same trend also holds for the more aggressive group-9 setting (i.e., 1:8 ratio of proposal to refined layers), where proposal-refined sharing consistently improves over direct reuse, suggesting that local refinement becomes increasingly important when indices are shared across more layers.

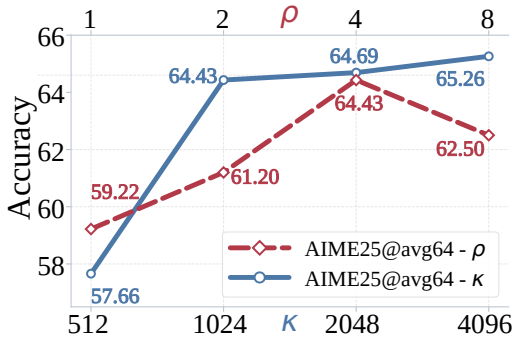


Figure 8: Sensitivity to proposal expansion factor ρ and sparse budget κ on AIME25.

We further study the sensitivity to the proposal expansion factor ρ and sparse budget κ in Figure 8. Increasing ρ from 1 to 4 improves accuracy from 59.22 to 64.43, confirming that a moderately expanded proposal region is necessary to cover localized cross-layer token shifts. However, further increasing ρ to 8 slightly degrades performance, suggesting that an overly large proposal may introduce more noisy candidates and weaken the refinement stage. In contrast, increasing the final sparse budget κ steadily improves accuracy from 57.66 at $\kappa = 512$ to 65.26 at $\kappa = 4096$, with most of the gain achieved at $\kappa = 1024$. These results indicate that ρ between 2 and 4, alongside a sparse budget κ between 1024 and 2048, strikes a highly effective balance between accuracy and efficiency.

6 Conclusion

We propose Prism Attention, a proposal-refined sparse attention framework that replaces rigid cross-layer index reuse with a coarse-to-fine strategy. Prism Attention lets proposal layers generate expanded candidate regions and refined layers re-select their own top-κ tokens within the corresponding proposed region, at a refinement cost independent of context length; INT2-quantized scoring further reduces the cost of proposal construction. Extensive experiments on reasoning and long-context benchmarks show that Prism achieves the best average accuracy among all evaluated sparse attention methods, while attaining a $3.4\times$ end-to-end decoding speedup over FlashAttention on 128K-token contexts, demonstrating that preserving layer-wise adaptability within a shared proposal region is an effective path toward accurate and efficient long-context inference.

Acknowledgments and Disclosure of Funding

We want to thank all the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China (NSFC No. 62622608 and 62576232), the National Engineering Research Center of Deep Learning Technology and Application, and sponsored by CCF-Baidu Open Fund. The authors declare no competing interests.

References

[1] Xuan Ai, Qingqing Yang, Peng Wang, Lei Deng, Lin Zhang, Renhai Chen, and Gong Zhang. Hylra: Hybrid layer reuse attention for efficient long-context inference. *arXiv preprint arXiv:2602.00777*, 2026.

[2] Anthropic. Claude sonnet 4.6, 2026.

[3] 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. *arXiv preprint arXiv:2603.12201*, 2026.

[4] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137, 2024.

[5] Dhruv Deshmukh, Saurabh Goyal, Nipun Kwatra, and Ramachandran Ramjee. Kascade: A practical sparse attention method for long-context llm inference. *arXiv preprint arXiv:2512.16391*, 2025.

[6] Hugging Face. Open r1: A fully open reproduction of deepseek-r1, January 2025.

[7] Qihang Fan, Huaibo Huang, Zhiying Wu, Juqiu Wang, Bingning Wang, and Ran He. Flashprefill: Instantaneous pattern discovery and thresholding for ultra-fast long-context prefilling. *arXiv preprint arXiv:2603.06199*, 2026.

[8] Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun, Tianzhu Ye, Li Dong, et al. Seerattention-r: Sparse attention adaptation for long reasoning. *arXiv preprint arXiv:2506.08889*, 2025.

[9] Yizhao Gao, Zhichen Zeng, Dayou Du, Shijie Cao, Peiyuan Zhou, Jiaxing Qi, Junjie Lai, Hayden Kwok-Hay So, Ting Cao, Fan Yang, et al. Seerattention: Learning intrinsic sparse attention in your llms. *arXiv preprint arXiv:2410.13276*, 2024.

[10] Google DeepMind. Introducing gemini 3, 2025.

[11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[12] Jitai Hao, Yuke Zhu, Tian Wang, Jun Yu, Xin Xin, Bo Zheng, Zhaochun Ren, and Sheng Guo. Omnikv: Dynamic context selection for efficient long-context llms. In *The Thirteenth International Conference on Learning Representations*, 2025.

[13] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

[14] Yuxuan Hu, Jianchao Tan, Jiaqi Zhang, Wen Zan, Pingwei Sun, Yifan Lu, Yerui Sun, Yuchen Xie, Xunliang Cai, and Jing Zhang. Optimizing native sparse attention with latent attention and local global alternating strategies. *arXiv preprint arXiv:2511.00819*, 2025.

[15] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[16] Xiaodong Ji, Hailin Zhang, Fangcheng Fu, and Bin Cui. Sale: Low-bit estimation for efficient sparse attention in long-context llm prefilling. *arXiv preprint arXiv:2505.24179*, 2025.

[17] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

[18] Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

[19] Aixin Liu, Aoxue Mei, Bangcai Lin, Bing Xue, Bingxuan Wang, Bingzheng Xu, Bochao Wu, Bowei Zhang, Chaofan Lin, Chen Dong, et al. Deepseek-v3. 2: Pushing the frontier of open large language models. *arXiv preprint arXiv:2512.02556*, 2025.

[20] Siran Liu, Guoxia Wang, Sa Wang, Jinle Zeng, HaoYang Xie, Siyu Lou, JiaBin Yang, DianHai Yu, Haifeng Wang, and Chao Yang. Rrattention: Dynamic block sparse attention via per-head round-robin shifts for long-context inference. *arXiv preprint arXiv:2602.05853*, 2026.

[21] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

[22] Art of Problem Solving. Aime problems and solutions. `https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions`.

[23] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

[24] Yutao Sun, Tianzhu Ye, Li Dong, Yuqing Xia, Jian Chen, Yizhao Gao, Shijie Cao, Jianyong Wang, and Furu Wei. Rectified sparse attention. *arXiv preprint arXiv:2506.04108*, 2025.

[25] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

[26] Zecheng Tang, Quantong Qiu, Yi Yang, Zhiyi Hong, Haiya Xiang, Kebin Liu, Qingqing Dang, Juntao Li, and Min Zhang. Elastic attention: Test-time adaptive sparsity ratios for efficient transformers. *arXiv preprint arXiv:2601.17367*, 2026.

[27] MiniCPM Team, Chaojun Xiao, Yuxuan Li, Xu Han, Yuzhuo Bai, Jie Cai, Haotian Chen, Wentong Chen, Xin Cong, Ganqu Cui, et al. Minicpm4: Ultra-efficient llms on end devices. *arXiv preprint arXiv:2506.07900*, 2025.

[28] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, pages 10–19, 2019.

[29] Xinghao Wang, Pengyu Wang, Xiaoran Liu, Fangxu Liu, Jason Chu, Kai Song, and Xipeng Qiu. Prism: Spectral-aware block-sparse attention. *arXiv preprint arXiv:2602.08426*, 2026.

[30] Yixuan Wang, Huang He, Siqi Bao, Hua Wu, Haifeng Wang, Qingfu Zhu, and Wanxiang Che. Proxyattn: Guided sparse attention via representative heads. *arXiv preprint arXiv:2509.24745*, 2025.

[31] Guangxuan Xiao, Junxian Guo, Kasra Mazaheri, and Song Han. Optimizing mixture of block attention. *arXiv preprint arXiv:2511.11571*, 2025.

[32] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. *arXiv preprint arXiv:2503.16428*, 2025.

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

[34] Lijie Yang, Zhihao Zhang, Zhuofu Chen, Zikun Li, and Zhihao Jia. Tidaldecode: Fast and accurate llm decoding with position persistent sparse attention. *arXiv preprint arXiv:2410.05076*, 2024.

[35] Lijie Yang, Zhihao Zhang, Arti Jain, Shijie Cao, Baihong Yuan, Yiwei Chen, Zhihao Jia, and Ravi Netravali. Less is more: Training-free sparse attention with global locality for efficient reasoning. *arXiv preprint arXiv:2508.07101*, 2025.

[36] Shang Yang, Junxian Guo, Haotian Tang, Qinghao Hu, Guangxuan Xiao, Jiaming Tang, Yujun Lin, Zhijian Liu, Yao Lu, and Song Han. Lserve: Efficient long-sequence llm serving with unified sparse attention. *Proceedings of Machine Learning and Systems*, 7, 2025.

[37] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, et al. 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, 2025.

[38] Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, et al. Glm-5: from vibe coding to agentic engineering. *arXiv preprint arXiv:2602.15763*, 2026.

[39] Jintao Zhang, Chendong Xiang, Haofeng Huang, Jia Wei, Haocheng Xi, Jun Zhu, and Jianfei Chen. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. *arXiv preprint arXiv:2502.18137*, 2025.

[40] Weilin Zhao, Zihan Zhou, Zhou Su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, et al. Infllm-v2: Dense-sparse switchable attention for seamless short-to-long adaptation. *arXiv preprint arXiv:2509.24663*, 2025.

A Detailed Settings for the Observation Experiments

We provide the detailed experimental settings of Figure 1 and Figure 2 below. The analysis covers multiple datasets, decoding steps, layers, and attention heads, providing a more comprehensive characterization of the inter-layer salient-token shift behavior.

Figure 1.

- **Model:** Qwen3-4B with 36 layers and 32 attention heads.
- **Dataset:** AIME24, MATH500, and GPQA; 10 randomly sampled instances per dataset.
- **Context length / decoding traces:** Approximately 1K and 8K decoding lengths; 32 consecutive decoding steps per trace (1024–1055 for 1K and 8192–8223 for 8K).
- **Layer offset:** Reference layers L_0–L_{28}, with offsets 1–7, i.e., comparing L_l and L_{l+d}.
- **Averaging procedure:** We compute the overlap ratio for each sample, decoding step, attention head, and layer pair, and average over the 32 heads, 32 decoding steps, 10 samples, and all valid layer pairs with the same offset. Results are reported separately for each dataset.

Figure 2.

- **Model:** Qwen3-4B with 36 layers and 32 attention heads.
- **Dataset:** AIME24, MATH500, and GPQA; 10 randomly sampled instances per dataset.
- **Context length / decoding traces:** Approximately 8K decoding length, with 32 consecutive decoding steps recorded per trace (steps 8192–8223).
- **Averaging procedure:** For each target layer, its top-128 indices are categorized according to their ranks in the reference layer as top-128, 128–511, or outside top-512. The ratios are averaged over the 32 heads, 32 decoding steps, and all samples. All datasets are pooled together for cross-dataset averaging.

B Generalization of Cross-Layer Attention Locality Across Architectures, Context Lengths, and Task Types

An analysis based only on the Qwen model family is insufficient to establish whether the observed cross-layer attention characteristics generalize across model architectures, context lengths, and task types. To address this concern, we conduct additional experiments along the following three dimensions:

- **Different model architectures:** Qwen3-4B (GQA), DeepSeek-R1-Distill-Llama-8B (GQA), and GLM-4.7-Flash (MLA);
- **Different sequence lengths:** 4K, 8K, and 16K;
- **Different task types:** mathematical reasoning, summarization, few-shot learning, and multi-document question answering.

Unless otherwise specified, all experiments follow the same analysis setup as Figure 2. For each target layer (i.e., Refined layer), we categorize its own top-128 sparse indices according to their ranks in the corresponding reference layer (i.e., Proposal layer): **Prop. Top-128**, **Prop. Top-128∼512**, and **Outside Prop. Top-512** denote the ratios of target-layer top-128 indices falling in the reference-layer top-128 set, ranks 128∼511, and outside the top-512 set, respectively.

B.1 Different Model Architectures

To examine whether this phenomenon generalizes beyond the Qwen model family, we additionally evaluate DeepSeek-R1-Distill-Llama-8B and GLM-4.7-Flash on 8K-token mathematical reasoning sequences, with Qwen3-4B included as a reference. Figure 9 demonstrates that the localized cross-layer drift observed in our study not only occurs in Qwen but also across different model architectures and attention mechanisms, including both GQA and MLA, and that proposal expansion continues to recover additional attention scores under the same trend across all evaluated architectures.

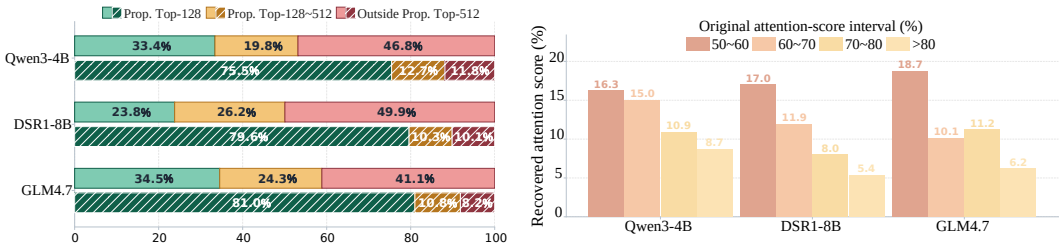


Figure 9: Cross-layer index/attention-mass distribution (Left) and attention score recovered by proposal expansion (Right) across model architectures. **Left figure**: within each pair of bars, the **solid** bar denotes the **index overlap ratio** (the share of the target/refined layer's 128 indices) and the **hatched** bar denotes the **attention-mass ratio** (the share of the total attention score those indices carry in the target layer). Both are normalized over the target layer's own Top-128, so the horizontal axis is a percentage and each bar sums to 100%. **Right figure**: the legend groups samples by the original attention coverage, i.e., the share of the target layer's Top-128 attention score that is already covered by reusing only the proposal layer's Top-128 indices. The vertical axis reports the additionally recovered attention score (in %) obtained by expanding the proposal region from Top-128 to Top-512, measured against the same target-layer Top-128 attention total, so both quantities are expressed on a common scale. The recovered attention decreases consistently as the original coverage grows: when the proposal layer's Top-128 already captures most of the target layer's attention, little is left for expansion to recover, whereas in the low-coverage regime—exactly the cases where directly reusing the proposal indices would be most lossy—the expanded region recovers the largest amount of attention score. **Note:** for conciseness, we abbreviate DeepSeek-R1-Distill-Llama-8B as DSR1-8B and GLM-4.7-Flash as GLM4.7 in both figures above.

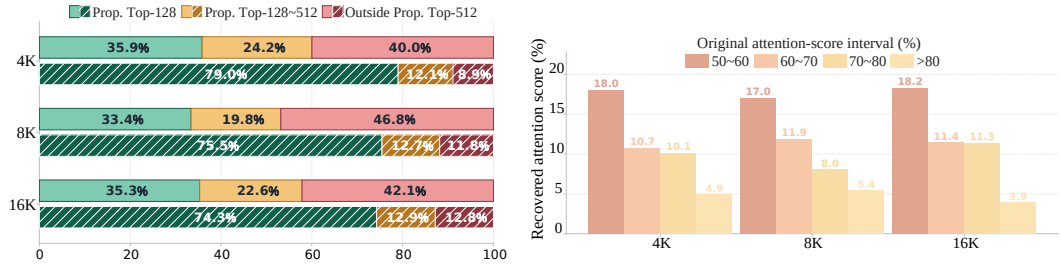


Figure 10: Cross-layer index/attention-mass distribution (Left) and attention score recovered by proposal expansion (Right) across context lengths. Both follow the conventions of Figure 9.

B.2 Different Sequence Lengths

To investigate whether the locality of cross-layer attention shifts changes with context length, we further conduct experiments on Qwen3-4B using mathematical reasoning sequences of 4K, 8K, and 16K tokens. All other analysis settings are kept unchanged. Figure 10 indicates that both the localized cross-layer shift pattern and the benefit of proposal expansion in recovering additional attention score hold robustly across all evaluated context lengths.

B.3 Different Task Types

To verify that this observation is not specific to mathematical reasoning, we further analyze four contextual task categories using Qwen3-4B, with all tasks based on 8K-token sequences:

- **Mathematical Reasoning:** 10 samples are randomly selected from each of AIME24, MATH500, and GPQA;
- **Summarization:** 10 samples are randomly selected from the summarization task category in LongBench;
- **Few-shot Learning:** 10 samples are randomly selected from the few-shot learning task category in LongBench;
- **Multi-Document QA:** 10 samples are randomly selected from the multi-document question-answering task category in LongBench.

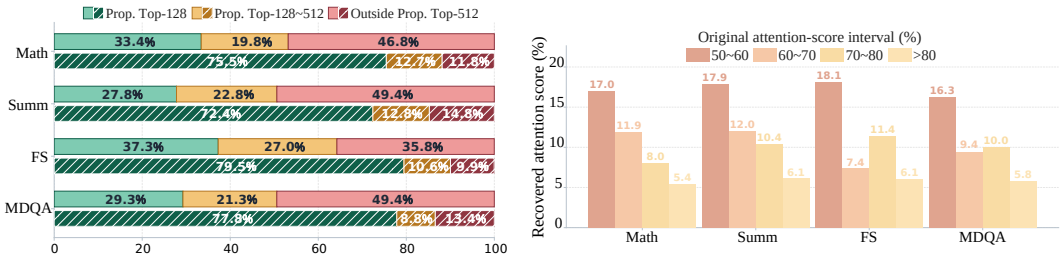
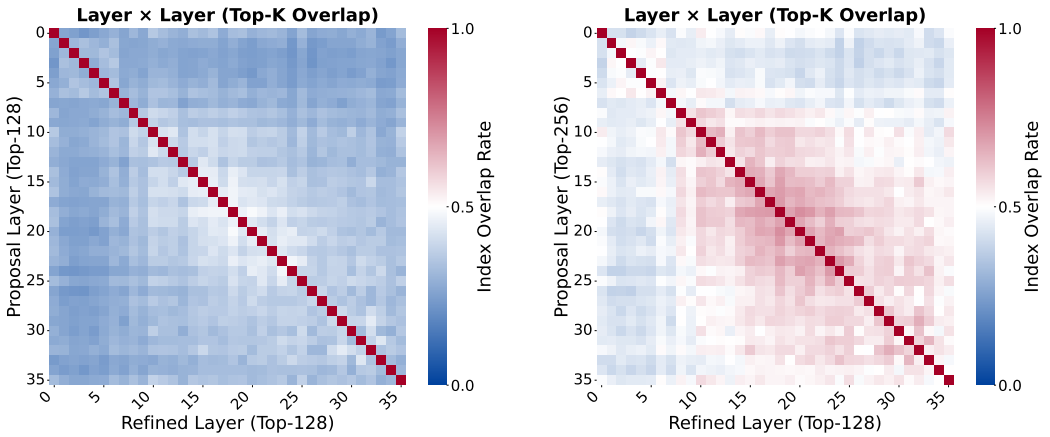
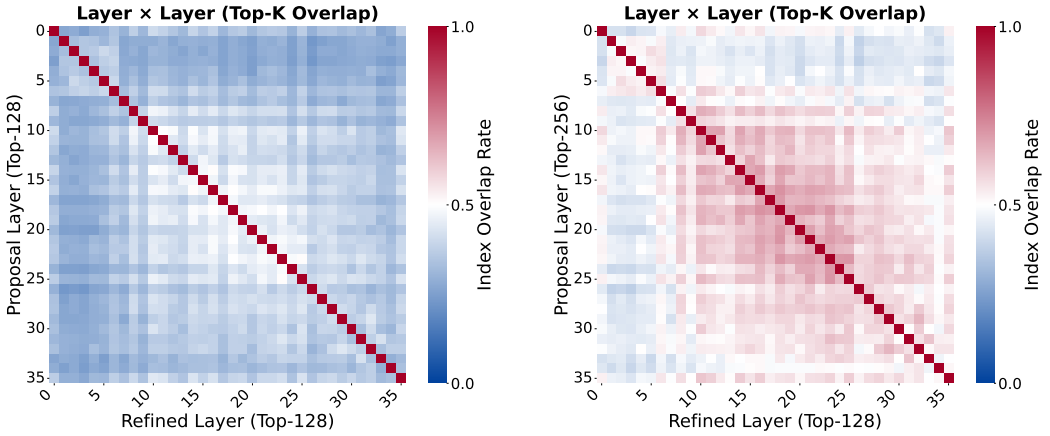


Figure 11: Cross-layer index/attention-mass distribution (Left) and attention score recovered by proposal expansion (Right) across task types. Both follow the conventions of Figure 9. **Note:** for conciseness, we abbreviate Mathematical Reasoning, Summarization, Few-shot Learning, and Multi-Document QA as Math, Summ, FS, and MDQA, respectively, in both figures above.



(a) GPQA, proposal=top-128.

(b) GPQA, proposal=top-256.



(c) MATH500, proposal=top-128.

(d) MATH500, proposal=top-256.

Figure 12: Effect of expanded proposal regions on cross-layer sparse-index coverage, with the refined-layer sparse support fixed at top-128 (Qwen3-4B; GPQA [23] and MATH500 [13] traces).

As shown in Figure 11, different tasks exhibit different degrees of cross-layer index variation, indicating that task type affects cross-layer attention dynamics. Nevertheless, despite these differences in the magnitude of cross-layer drift, the expanded proposal region consistently covers a set of candidate tokens with substantial attention contributions across all evaluated tasks. These results demonstrate that the localized cross-layer attention shifts observed in our study are not limited to mathematical reasoning, but also occur in summarization, few-shot learning, and multi-document question answering, which exhibit substantially different context-access patterns.

Table 4: Searched Proposal/Refined layer assignment patterns (**F**: Proposal layer, **S**: Refined layer).

Model	Config	Pattern
Qwen3-4B	group-4	FSSSSSSFFSSSSFSFSFFSSSSSSSFSSSSSSSSF
	group-9	FSSSSSSFSSSSSFSSSSFSSSSSSSSSSSSSSSS
Qwen3-8B	group-4	FSSSSSSFFSSSSFFSSSFSFFSSSSSSSFSSSSSSS
	group-9	FSSSSSSFSSSSSFFSSSSSSSSSSSSSSSSSSSSS

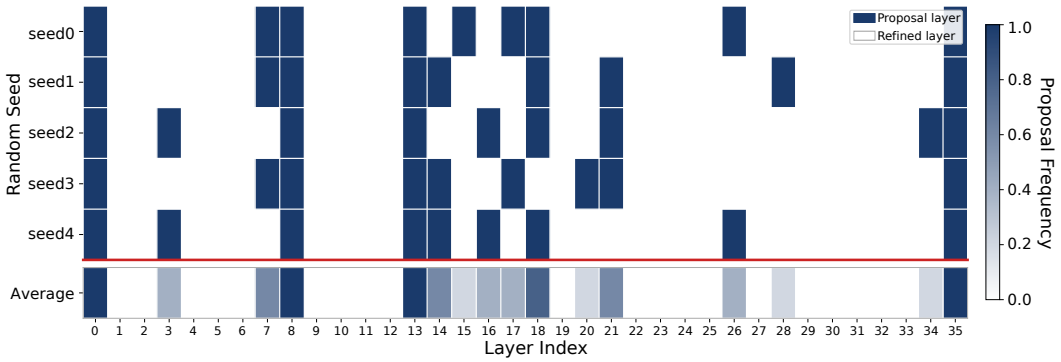


Figure 13: Sensitivity of proposal layer assignment to calibration sample selection on Qwen3-4B.

To further verify the universality of our observations on AIME24, we present additional index-overlap visualizations for the GPQA and MATH500 datasets. As shown in Figure 12, attention index dynamics are highly consistent regardless of the task domain. Rigid index reuse (left panels, top-128) yields poor overlap for distant layers, reflecting how salient tokens inherently shift across network depth. Conversely, expanding the proposal pool to top-256 ($\rho = 2$, right panels) effectively preserves cross-layer coverage. This confirms that inter-layer token shifting is a universal phenomenon addressed by our localized expansion strategy across various applications.

C Sensitivity to Calibration Sample Selection

We perform sensitivity analyses regarding the selection of calibration samples. As illustrated in Figure 13, the majority of proposal layers exhibit high stability across different sample subsets selected via various random seeds. Notably, layers 0, 8, 13, and 35 are almost always designated as proposal layers, which is broadly consistent with the searched patterns reported in Table 4. These observations underscore the robustness and consistency of the offline calibration procedure.

To further validate these findings, we evaluate the downstream performance of the varying layer assignment patterns generated across different random seeds. We maintain a sparse budget of $\kappa = 1024$ and an expansion factor of $\rho = 4$. The results, presented in Table 5, indicate that the final accuracies achieved by these different patterns remain remarkably stable. This consistent downstream performance further demonstrates that our proposal-refined mechanism is inherently robust to minor fluctuations in layer assignments, and that the offline calibration reliably captures generalized attention redundancy without overfitting to the specific calibration subsets.

D Complete Efficiency Results: From Kernels to End-to-End

Latency of individual kernel components. We first measure our customized Triton kernels in isolation, timing each attention-path component separately under CUDA Graph replay at the full head configuration of Qwen3-4B, without tensor-parallel sharding and without any model-level or framework-level overhead. Figure 14 provides additional measurements for BSZ=1 and BSZ=16, and Table 6 reports the exact execution time of every component alongside the FlashAttention baseline. The decomposition shows that index refinement stays essentially constant at roughly 0.011

Table 5: Performance across layer assignment patterns from random calibration seeds.

Metric	Seed 0	Seed 1	Seed 2	Seed 3	Seed 4
AIME25	64.43	64.32	64.69	64.06	63.91

Table 6: Detailed latency breakdown (ms) of Prism Attention kernels compared with FlashAttention across various batch sizes and sequence lengths.

Batch Size	**Component**	**8K**	**16K**	**32K**	**64K**	**128K**
BSZ=1	FlashAttention	0.0293	0.0463	0.0679	0.1097	0.1945
	Quantization	0.0111	0.0109	0.0111	0.0109	0.0109
	Proposal Construction	0.0132	0.0182	0.0270	0.0446	0.0807
	Index Refinement	0.0108	0.0111	0.0109	0.0109	0.0109
	Sparse Decode	0.0333	0.0331	0.0333	0.0332	0.0334
BSZ=4	FlashAttention	0.0644	0.1066	0.1911	0.3617	0.6985
	Quantization	0.0109	0.0110	0.0110	0.0110	0.0109
	Proposal Construction	0.0268	0.0459	0.0831	0.1503	0.2930
	Index Refinement	0.0109	0.0108	0.0109	0.0109	0.0106
	Sparse Decode	0.0340	0.0337	0.0337	0.0335	0.0340
BSZ=16	FlashAttention	0.1864	0.3539	0.6871	1.3633	2.6978
	Quantization	0.0110	0.0109	0.0111	0.0110	0.0107
	Proposal Construction	0.0824	0.1492	0.2853	0.5482	1.1019
	Index Refinement	0.0149	0.0152	0.0151	0.0150	0.0150
	Sparse Decode	0.0547	0.0569	0.0580	0.0605	0.0623

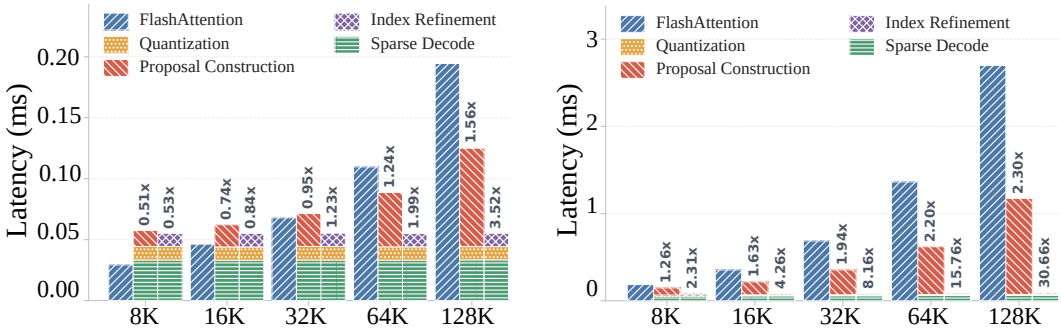
ms from 8K to 128K, in sharp contrast to proposal construction, which scales with context length, confirming that the local refinement step performed by every refined layer incurs negligible and length-independent cost, and that the length-sensitive cost is confined to proposal layers.

Per-layer latency under the SGLang integration. We further evaluate the same kernels after integrating them into SGLang, where each layer is timed as a single fused CUDA Graph chain rather than as separate components, using the tensor shapes of one TP8 rank (4 query heads, 1 KV head, head dimension 128) and a 2K sparse budget. Table 7 reports the cost of an index-construction layer (Gate) and an index-reuse layer (Refine), together with the effective per-layer cost under the $1{:}3$ ratio adopted throughout our experiments. The reuse-layer cost again remains nearly flat across all context lengths while the construction-layer cost grows only mildly, so the effective cost scales far more slowly than dense FlashAttention and reaches a $5.42\times$ speedup at 128K with BSZ$= 16$. Prism carries a modest overhead relative to IndexCache at short contexts and small batches, since INT2 quantization and the enlarged candidate pool introduce a fixed cost that is not yet amortized; this gap narrows as the workload grows and reverses at 128K with BSZ$= 16$, where Prism's cheaper index construction makes it the faster of the two.

End-to-end serving latency. Table 8 provides complete decoding latency measured inside the SGLang framework, so that the kernel-level gains above can be assessed under a realistic deployment.

E Scaling Prism Attention to GLM-5.2

We further apply Prism Attention to GLM-5.2 (78 layers), evaluating under a unified sparse budget of $\kappa = 2048$. Table 9 lists the three evaluated layer-assignment patterns, and Table 10 reports the overall accuracy on LongBench-V2 (128K) and RULER (64K/128K); the per-subtask RULER results are further detailed in Table 11. Introducing Prism Attention recovers the accuracy that IndexCache loses when its $\mathcal{F}$ layers are reduced, matching the default pattern with far fewer proposal layers.



(a) Batch Size = 1 (b) Batch Size = 16

Figure 14: Component-wise latency breakdown of Prism Attention compared to the FlashAttention baseline across different sequence lengths at (a) BSZ=1 and (b) BSZ=16.

Table 7: Attention-path kernel latency breakdown under a 2K sparse budget, measured with our Triton kernels integrated into the **SGLang** serving framework. **Gate** and **Refine** denote the full layer chain of an index-construction layer (Prism: proposal, IndexCache: anchor) and an index-reuse layer (Prism: refined, IndexCache: shared), respectively. **Eff.** is the effective cost at the 1:3 construction-to-reuse ratio used throughout our experiments, i.e., $T = (T_{\text{construct}} + 3\,T_{\text{reuse}})/4$; parenthesized values are speedups over FlashAttention (FA). All latencies in ms.

		Prism			IndexCache			
BSZ	Seq	Gate	Refine	Eff.	Gate	Reuse	Eff.	FA
1	8K	0.0328	0.0208	0.0238 (0.61×)	0.0279	0.0116	0.0157 (0.93×)	0.0145
	16K	0.0491	0.0198	0.0271 (0.67×)	0.0561	0.0116	0.0227 (0.80×)	0.0182
	32K	0.0490	0.0196	0.0269 (0.79×)	0.0565	0.0117	0.0229 (0.93×)	0.0213
	64K	0.0518	0.0197	0.0278 (0.89×)	0.0578	0.0115	0.0231 (1.07×)	0.0247
	128K	0.0545	0.0198	0.0285 (1.50×)	0.0603	0.0118	0.0240 (1.78×)	0.0426
8	8K	0.0377	0.0233	0.0269 (0.81×)	0.0321	0.0123	0.0173 (1.26×)	0.0218
	16K	0.0666	0.0222	0.0333 (1.17×)	0.0553	0.0124	0.0231 (1.68×)	0.0388
	32K	0.0653	0.0221	0.0329 (1.81×)	0.0711	0.0123	0.0270 (2.20×)	0.0594
	64K	0.0846	0.0223	0.0379 (2.70×)	0.0944	0.0121	0.0327 (3.14×)	0.1025
	128K	0.1169	0.0226	0.0461 (4.10×)	0.1393	0.0123	0.0441 (4.29×)	0.1892
16	8K	0.0432	0.0270	0.0311 (1.22×)	0.0417	0.0133	0.0204 (1.86×)	0.0379
	16K	0.0731	0.0262	0.0380 (1.55×)	0.0747	0.0134	0.0287 (2.05×)	0.0589
	32K	0.0905	0.0268	0.0427 (2.38×)	0.0979	0.0135	0.0346 (2.94×)	0.1017
	64K	0.1230	0.0281	0.0518 (3.60×)	0.1432	0.0135	0.0459 (4.06×)	0.1863
	128K	0.1794	0.0279	0.0658 (5.42×)	0.2326	0.0137	0.0684 (5.22×)	0.3569

Table 8: End-to-end per-step decoding latency on Qwen3-4B with the SGLang serving framework under a 2K sparse budget. Each cell reports the absolute latency (ms) and, in parentheses, the speedup over FlashAttention (FA).

	Seq = 16K		Seq = 64K		Seq = 128K	
Method	BSZ=1	BSZ=16	BSZ=1	BSZ=16	BSZ=1	BSZ=16
FA	2.151 (1.00×)	4.057 (1.00×)	2.623 (1.00×)	8.748 (1.00×)	3.074 (1.00×)	15.025 (1.00×)
Seer-R	2.520 (0.85×)	3.101 (1.31×)	2.580 (1.02×)	3.208 (2.73×)	2.659 (1.16×)	3.483 (4.31×)
LIM	2.399 (0.90×)	3.465 (1.17×)	2.517 (1.04×)	4.166 (2.10×)	2.667 (1.15×)	5.147 (2.92×)
IDC-g4	2.261 (0.95×)	2.854 (1.42×)	2.317 (1.13×)	3.550 (2.46×)	2.450 (1.26×)	4.434 (3.39×)
IDC-g9	2.026 (1.06×)	2.557 (1.59×)	2.062 (1.27×)	2.912 (3.00×)	2.156 (1.43×)	3.348 (4.49×)
Prism-g4	2.492 (0.86×)	3.302 (1.23×)	2.565 (1.02×)	3.835 (2.28×)	2.678 (1.15×)	4.467 (3.36×)
Prism-g9	2.314 (0.93×)	3.091 (1.31×)	2.374 (1.11×)	3.343 (2.62×)	2.474 (1.24×)	3.703 (4.06×)

Algorithm 1 PrismAttention Decoding at Step t

Require: Input representation X_t; layers $1, \ldots, N$ partitioned into proposal layers $\mathcal{F}$ and refined layers $\mathcal{S}$; association function $f : \mathcal{S} \to \mathcal{F}$; sparse budget κ; expansion factor $\rho \geq 1$.
// $\mathcal{C}_t^{\mathcal{F}}$: shared candidate cache populated by proposal layers.

for $\ell = 1$ **to** N **do**
  if $\ell \in \mathcal{F}$ **then** ▷ Proposal layer
    $\tilde{s}_t^{\mathcal{F}} \leftarrow \mathrm{QuantizedProposal}_\ell(X_t)$ ▷ approx. scoring with INT2 keys
    $\mathcal{C}_t^{\mathcal{F}} \leftarrow \mathrm{TopK}\big(\tilde{s}_t^{\mathcal{F}},\, \rho\kappa\big)$ ▷ expanded candidate pool, size $n = \rho\kappa$
    $\mathcal{T}_t^{\mathcal{F}} \leftarrow \mathcal{C}_t^{\mathcal{F}}[1{:}\kappa]$ ▷ proposal layer attends to its own top-κ
  else ▷ Refined layer, $\ell \in \mathcal{S}$
    $\widehat{s}_t^{\mathcal{S}} \leftarrow \mathrm{Refine}_\ell\big(X_t,\, \mathcal{C}_t^{f(\mathcal{S})}\big)$ ▷ rescore only within the inherited pool with INT2 keys
    $\widehat{\mathcal{T}}_t^{\mathcal{S}} \leftarrow \mathrm{TopK}\big(\widehat{s}_t^{\mathcal{S}},\, \kappa\big)$
  end if
  $\mathrm{SparseAttn}_\ell\big(X_t,\, \mathcal{T}_t^{\ell}\big)$ ▷ $\mathcal{T}_t^{\ell} = \mathcal{T}_t^{\mathcal{F}}$ if $\ell \in \mathcal{F}$, else $\widehat{\mathcal{T}}_t^{\mathcal{S}}$
end for

Table 9: Layer-assignment patterns evaluated on GLM-5.2 under a unified sparse budget $\kappa = 2048$ and proposal expansion factor $\rho = 4$. **Note:** the meanings of F, S, and R here differ from those in Table 4, and throughout this table follow the definitions given below. **F** (Full) layers construct sparse indices from the full context, **S** (Shared) layers directly reuse the indices of their nearest preceding index-producing layer, and **R** (Refined) layers refine an inherited candidate pool. *21F57S* is the default pattern of GLM-5.2 itself, i.e., its native trained IndexCache configuration, where each F layer builds its own κ indices and each S layer reuses these κ indices from the nearest preceding F layer. *10F68S* is an IndexCache variant calibrated to fewer F layers. *10F11R57S* incorporates Prism as a hybrid of F, R, and S layers: each F layer builds an expanded pool of $\rho\kappa$ candidate indices, each R layer selects its own top-κ from the $\rho\kappa$ pool of the nearest preceding F layer, and each S layer reuses the κ indices of its nearest preceding F or R layer.

Config	Pattern
21F57S (Default)	FFFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSSFSSS
10F68S	FFSSSSSSSSSSSSFSSSFSSSSSSSFSSSSSSSFSSSSSSSFSSSSSSSSSSSSSSSSSSSFSSSSSSSFSSSFSSS
10F11R57S	FFRSSSRSSSRSSSFSSSFSSSRSSSFSSSRSSSFSSSRSSSFSSSRSSSRSSSRSSSRSSSFSSSRSSSFSSSFSSS

Table 10: Accuracy on GLM-5.2 with the SGLang framework under a unified sparse budget $\kappa = 2048$. Each cell reports accuracy and, to the right of the slash, the average output length. The $_L$ suffix in each dataset name denotes the maximum input length.

Config	LongBenchV2_128K	Ruler_64K	Ruler_128K
21F57S	65.91 / 4696	96.03 / 3802	95.53 / 3254
10F68S	65.96 / 4972	95.68 / 3507	93.05 / 3224
10F11R57S	66.06 / 4798	96.24 / 3086	95.15 / 2752

Table 11: Per-subtask accuracy (%) on RULER (64K and 128K) with GLM-5.2 under the SGLang framework and a unified sparse budget $\kappa = 2048$.

	Config	S-NIAH-1	S-NIAH-2	S-NIAH-3	MK-NIAH-1	MK-NIAH-2	MK-NIAH-3	MQ-NIAH	MV-NIAH	HotpotQA	SQuAD	CWE	FWE	VT	**AVG**
64K	21F57S	100.00	100.00	98.00	100.00	100.00	100.00	99.00	100.00	79.00	84.50	97.90	90.00	100.00	96.03
	10F68S	100.00	100.00	100.00	100.00	94.00	99.00	100.00	100.00	75.00	85.75	94.70	95.33	100.00	95.68
	10F11R57S	100.00	100.00	100.00	99.00	100.00	100.00	97.00	100.00	78.00	82.83	99.30	95.00	100.00	96.24
128K	21F57S	100.00	100.00	100.00	100.00	100.00	100.00	100.00	98.00	76.00	88.17	95.00	84.67	100.00	95.53
	10F68S	100.00	100.00	100.00	100.00	86.00	88.00	96.00	98.50	76.00	84.83	89.60	90.67	100.00	93.05
	10F11R57S	100.00	100.00	98.00	100.00	98.00	100.00	100.00	94.50	80.00	79.50	95.00	92.00	100.00	95.15

F Limitation and Future Work

Prism inherits its Proposal/Refined layer partition from the offline greedy calibration procedure of IndexCache: the partition is searched once on a held-out mathematical-reasoning corpus under a token-level cross-entropy objective and then frozen for all downstream evaluations. We adopt this recipe as is and do not treat the calibration procedure itself as a design variable. Since our analysis shows cross-layer index drift to be markedly task-dependent, a systematic exploration of offline calibration along three axes—search strategies stronger than greedy placement, objectives aligned more directly with downstream accuracy, and calibration corpora whose domain and length distribution match the target workload—is likely to yield a more favorable layer assignment at no additional inference cost.

G Broader Societal Impacts

Our work on accelerating long-context LLM inference with Prism has both positive and negative societal implications. On the positive side, it aligns with the goals of Green AI by substantially reducing the energy consumption and carbon footprint of LLM deployment, and it broadens access to advanced long-context models by lowering the hardware barrier for resource-constrained researchers. At the same time, cheaper and faster inference also lowers the cost for malicious actors to mount large-scale misinformation or phishing campaigns. Our algorithmic improvements thus amplify existing risks rather than introduce qualitatively new ones. Mitigating this dual-use potential requires deploying such efficiency optimizations alongside robust safety alignment and content-filtering protocols.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We reflect the paper's contributions and scope in the abstract and introduction.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Appendix.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: We provide the full set of assumptions and a complete (and correct) proof in Section Method.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We have provided all necessary information to reproduce our results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We will open source all the code when the paper is accepted.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We specify all the experimental detail in Section Experiment.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We provide the details about initialization and dataset.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have stated the full range of compute resources in Section Experiment

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have read the ethics guidelines, and we believe our paper conforms to them.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss them in Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We create no new data or models to release, as our method proposes a modification to existing pretrained models for inference efficiency.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All existing code, datasets and models adopted in this paper are properly cited with original authorship, and their corresponding open-source licenses as well as usage terms are clearly stated and fully respected throughout the work.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All newly introduced datasets, code and models will be well-documented and publicly released together with full documentation in the final version.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Not applicable

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Not applicable.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.