AdaCal: Adaptive Calibration for Robust Sparse Attention in Long-Context LLMs

Leqin Xiang
School of Computer
University of South China
Hengyang, China
leqinxiang@gmail.com

Yongbin Liu*
School of Computer
University of South China
Hengyang, China
yongbinliu03@gmail.com

Chunping Ouyang
School of Computer
University of South China
Hengyang, China
ouyangcp@126.com

Ying Yu
School of Computer
University of South China
Hengyang, China
yuying@usc.edu.cn

Abstract

Large language models face substantial computational challenges in long-sequence inference due to the quadratic cost of self-attention, with the prefill stage being particularly bottlenecked. Static sparse attention reduces compute via predefined masks but can miss task-critical evidence, while dynamic sparsification is more flexible yet can be brittle across tasks, leading to either insufficient coverage or distractor-induced noise. We propose AdaCal, an Adaptive Calibration framework for robust sparse attention that synergizes coarse, experience-driven task dispatch with head-wise physical profiling. By deriving entropy, concentration, and mid-zone coverage metrics from a lightweight probe pattern, AdaCal dynamically governs head activation and augmentation policies, allocating additional computation strictly on demand. This design is realized through task-conditional policies that effectively suppress unnecessary expansion for structure-dominant inputs while enhancing long-range coverage for retrieval-intensive tasks. Experiments show that AdaCal improves robustness over strong static and dynamic baselines, with clearer gains in long-context settings beyond 32K tokens.

1 Introduction

In recent years, large language models have made substantial progress in long-context understanding, including long-text question answering Kryściński et al. [2022], Porwal et al. [2023], Chang et al. [2023], long-text summarization Caciularu et al. [2023], Pang et al. [2022], Fan et al. [2019], and repository-level code masking Wang et al. [2025, 2024]. However, Transformer-based models incur quadratic time and space complexity as sequence length grows, with the $O(n^2)$ prefill stage during inference becoming a major bottleneck for practical long-sequence deployment Fu [2024]. Fortunately, attention is inherently sparse, as softmax normalization often produces many near-zero entries Deng et al. [2024], suggesting that computation can be reduced by constructing a sparse mask and restricting attention to selected key regions.

Existing sparse attention methods mainly fall into three categories. Predefined sparse patterns, such as BigBird Zaheer et al. [2020], Longformer Beltagy et al. [2020], and LongLora Chen et al.

*Corresponding author.

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

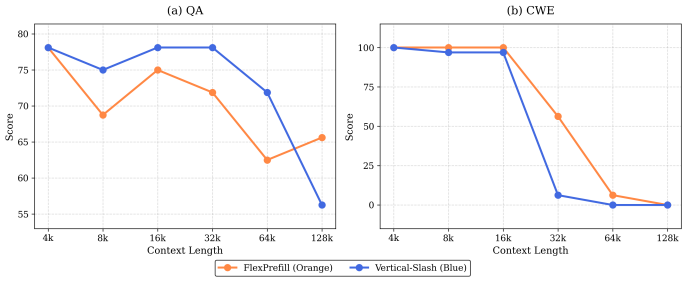


Figure 1: The Noise-Recall Dilemma on RULER. (a) QA: Static Vertical-Slash (Blue) outperforms FlexPrefill (Orange), revealing that blind dynamic sparsity injects detrimental noise. (b) CWE: Conversely, Vertical-Slash collapses on long contexts (> 32k), confirming its failure in retrieval. This trade-off necessitates our profile-guided adaptive routing to balance noise rejection and global recall.

[2023], improve efficiency but limit sparsity flexibility. Training-based approaches Kitaev et al. [2020], Pagliardini et al. [2023], Zhang et al. [2025] learn locally sparse attention, but may sacrifice global dependency modeling and struggle to generalize. Dynamic sparse methods Jiang et al. [2024] leverage pre-computed static patterns for efficiency, but remain limited in capturing transient semantic dependencies. FlexPrefill Lai et al. [2025] improves flexibility by categorizing heads into Structured or Query-Aware patterns using Jensen–Shannon divergence.

Despite this progress, our pilot study on RULER Hsieh et al. [2024] reveals a task-dependent noise–recall dilemma in long-context sparsification, as shown in Figure 1. On QA-style tasks, a dynamic sparsification baseline can underperform a fixed-mask Vertical-Slash (VS) pattern, suggesting that unnecessary expansion may introduce distractors when a stable sparse structure is sufficient. In contrast, fixed VS deteriorates rapidly on CWE retrieval as context length increases, exposing the limited recall of static patterns for long-range evidence. These observations indicate that sparse attention should make task-aware decisions: preserving the base topology for structure-dominant inputs while selectively allocating additional computation for retrieval-intensive ones.

Motivated by this observation, we propose AdaCal, an Adaptive Calibration framework that moves beyond estimator-driven routing and bases sparse decisions on representation-centric profiling. AdaCal uses a lightweight Vertical-Slash probe to obtain low-cost attention summaries, computes physically interpretable head profiles including entropy, concentration, and mid-zone coverage, and triggers additional computation only when these signals indicate clear necessity. Since attention patterns differ across tasks even under full attention, we further introduce coarse, experience-driven task dispatch to constrain subsequent sparse attention decisions.

Our mechanism instantiates three task-conditional policies: 1) No-Expand for structure-dominant inputs to suppress interference by locking to the base topology; 2) Global-TopK for reasoning-intensive inputs to enable limited global augmentation on demand; and 3) Coverage-Sampling for retrieval-intensive inputs to adapt sampling density (via mid-zone coverage) to recover long-range signals. This design separates noise suppression from coverage recovery, and avoids one-size-fits-all expansion. Across our evaluated settings, the resulting targeted allocation provides a favorable robustness–efficiency balance in practice.

We summarize our contributions as follows:

1. **Identification of Noise-Recall Dilemma.** Using RULER as a representative long-context evaluation, we observe the following trends in Figure 1: on QA-style settings, a dynamic sparsification baseline can underperform a simple static Vertical-Slash pattern, while on the CWE retrieval setting, the static Vertical-Slash baseline degrades sharply as context length grows. These trends motivate task-aware sparse attention decisions.
2. **Constrain-then-Calibrate Framework.** We propose a Constrain-then-Calibrate design that combines coarse task dispatch with head-wise physical profiling, enabling targeted sparse attention computation without relying on unstable estimation proxies.
3. **Task-Conditional Routing.** We instantiate the mechanism with three mutually exclusive policies, No-Expand, Global-TopK, and Coverage-Sampling, that address noise sensitivity and long-range coverage within a unified mechanism.

The results show that our method often improves upon strong static and dynamic baselines across context lengths up to 128K. Across the evaluated settings, it tends to provide a more favorable robustness–efficiency balance in practice, without requiring task-specific retraining.

2 Related Work

Fixed Sparse Attention. Fixed sparse attention uses predefined topologies to reduce the quadratic cost of attention. Representative methods such as Longformer Beltagy et al. [2020] and BigBird Zaheer et al. [2020] combine local windows with a small number of global connections, making them efficient and kernel-friendly. Similarly, StreamingLLM Xiao et al. [2023] exploits attention sinks by retaining initial tokens together with local attention for streaming generation. Although effective in their target settings, these methods largely rely on fixed connectivity patterns and may under-cover sparse or irregular long-range evidence in retrieval-intensive contexts.

Adaptive Sparsification. Adaptive sparsification instead adjusts the attended tokens or blocks according to runtime signals. Some methods mainly target decoding-time KV-cache management: H2O Zhang et al. [2023] and SnapKV Li et al. [2024] reduce memory usage by evicting or compressing less salient KV pairs based on accumulated attention statistics. Another line focuses on prefill-time sparse attention. MInference Jiang et al. [2024] estimates sparse indices through approximate attention over down-sampled inputs, while FlexPrefill Lai et al. [2025] switches between structured patterns, such as Vertical-Slash, and query-aware patterns using Jensen–Shannon divergence. These adaptive methods improve flexibility over fixed sparsity, but their estimator-driven routing can be brittle across heterogeneous tasks: a confident sparse estimate does not necessarily imply that expansion is beneficial, and unnecessary expansion may introduce distractors for structure-sensitive inputs. AdaCal differs by first constraining the sparse decision space with coarse task dispatch and then using head-wise physical profiling to decide when and how additional sparse computation should be allocated.

3 Methodology

3.1 Preliminaries

Transformer-based large language models rely on self-attention to aggregate contextual information. Given an input sequence of length L, the standard sparse self-attention computes:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^\top}{\sqrt{d}} + M\right) V, \tag{1}$$

where $Q, K, V \in \mathbb{R}^{L\times d}$ denote the query, key, and value matrices, and M is the sparse mask. While exact attention preserves full contextual expressivity, its quadratic computational and memory complexity in L makes it prohibitively expensive for long-context inference.

In long-context inference, the quadratic cost of attention becomes most acute in the prefill stage, which must process the entire prompt before decoding. As prompts grow to tens or hundreds of thousands of tokens, reducing prefill cost is a key bottleneck.

Sparse attention mitigates this by letting each query attend to only a subset of keys. In practice, this is often implemented at the block level, the sequence is partitioned into blocks, and query q attends to a selected block set $\mathcal{M}(q)$, yielding cost $\sum_q |\mathcal{M}(q)|$ instead of L^2.

We study budgeted sparse attention for long-context prefill, where $\mathcal{M}(\cdot)$ must satisfy a compute budget while preserving task performance. Crucially, we treat sparsification as a decision problem, under limited compute, which connections are necessary for a given task and input? This framing motivates the task-dependent analysis that follows.

3.2 Motivating Observations

Despite recent progress in sparse and dynamic attention, existing approaches still show consistent failure modes across heterogeneous long-context tasks, which become more pronounced as context length increases (Figure 1).

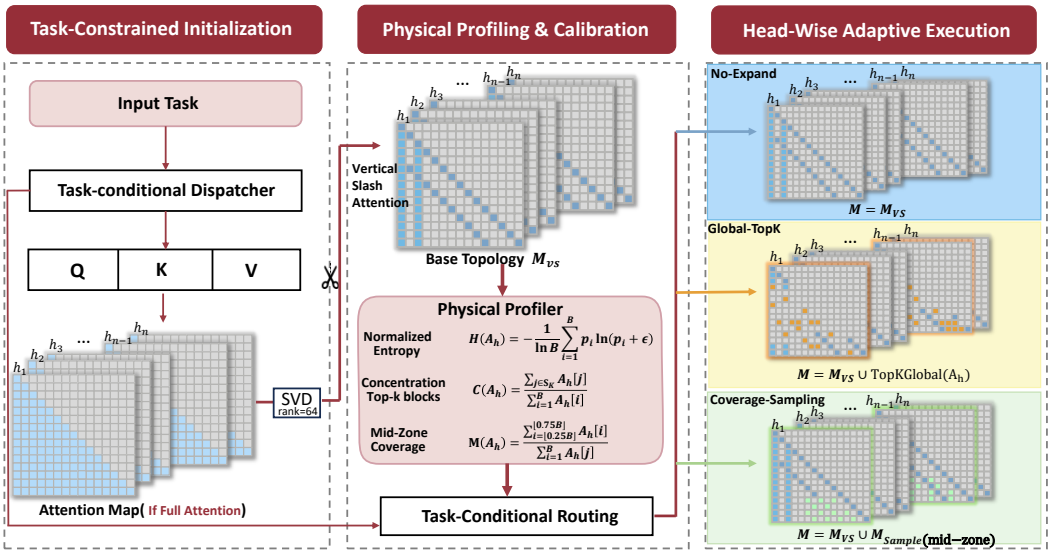
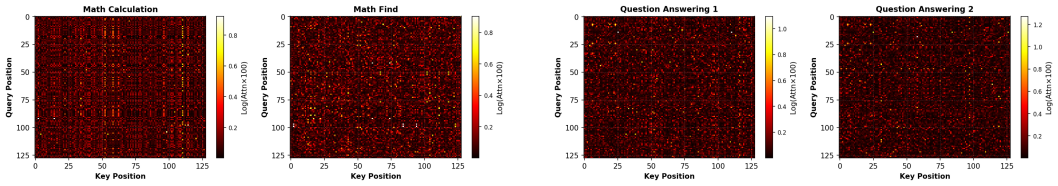


Figure 2: Overview of AdaCal. **Left:** Coarse task dispatch constrains the sparse attention policies. **Middle:** A lightweight Vertical-Slash probe provides head-wise entropy, concentration, and mid-zone coverage for adaptive calibration. **Right:** The task-constrained policies retain the base topology (No-Expand), selectively add global blocks (Global-TopK), or supplement mid-zone coverage (Coverage-Sampling). This constrain-then-calibrate design allocates additional computation only when indicated by the head profiles.

Two observations in long-context sparsification. On structure-dominant settings (e.g., RULER QA), dynamic sparsification can underperform a simple static baseline, global expansion may introduce task-irrelevant yet semantically related blocks, degrading accuracy. On retrieval-intensive settings such as CWE, static patterns often under-cover long-range evidence, and performance drops sharply with longer contexts (Figure 1). Together, these results expose a task-dependent trade-off, static sparsity is often more robust but may miss long-range evidence, whereas dynamic sparsity improves coverage but risks introducing distractors, suggesting that a single sparsification rule is unlikely to be uniformly effective.

Attention patterns differ across tasks even before sparsification. To connect these trends to attention structure, we visualize block-level heatmaps under full attention with matched context lengths (Figure 3). Notably, these patterns are measured before any sparsification is applied. We observe qualitatively different attention distributions across tasks, indicating that the appropriate attention topology is not universal.

Implication: coarse task dispatch as a prior. Motivated by the above, we introduce an explicit task dispatch step. Rather than predicting fine-grained task identities, we rely on coarse, experience-driven grouping of task characteristics (*structure-dominant*, *reasoning-intensive*, *retrieval-intensive*), and use the resulting task prior to constrain subsequent sparse attention decisions. Within this constraint, head-wise adaptation can be performed more safely and efficiently, forming the basis of the Constrain-then-Calibrate mechanism introduced next.



(a) Attention heatmaps on math tasks (two instances). (b) Attention heatmaps on QA tasks (two instances).

Figure 3: Full-attention heatmaps on representative tasks.

3.3 Constrain-then-Calibrate Framework

We now describe our profile-guided sparse attention mechanism for long-context prefill. The design follows a simple principle, constrain first, calibrate after. We first use a coarse task prior (from task dispatch) to restrict allowable sparse attention behaviors, and then allocate additional computation only when low-cost diagnostic signals indicate clear necessity. Figure 2 provides an overview.

Notation, block partition, and task-constrained initialization. Given a sequence of length L, we partition it into token blocks of size b, yielding $N = \lceil L/b \rceil$ blocks. For each head h, we denote by $\mathbf{A}_h \in \mathbb{R}^N$ the block-level attention mass, aggregated under a base sparse topology; unless stated otherwise, all profiling metrics and triggers are computed per head. We further apply a coarse, experience-driven task dispatch that assigns each input to one of three categories (*structure-dominant*, *reasoning-intensive*, *retrieval-intensive*), and use this task prior to constrain subsequent sparse attention decisions: structure-dominant inputs forbid dynamic expansion to suppress noise, reasoning-intensive inputs permit limited global augmentation on demand, and retrieval-intensive inputs enable coverage-oriented sampling to recover long-range signals.

Base topology as a universal probe. We adopt Vertical-Slash (VS) attention as a universal low-cost base topology throughout the pipeline. VS combines a diagonal local band with sparse vertical landmarks, providing limited global access while keeping prefill cost low. We emphasize that VS is not assumed to be optimal by itself; instead, it serves as a stable and efficient probe. We first compute a lightweight attention distribution under the VS mask $\mathcal{M}_{VS}$. Optionally, before this probe computation we apply a rank-r randomized-SVD reconstruction to (K, V) (decomposition and cast back), yielding $(\tilde{K}, \tilde{V})$ while keeping the downstream interface unchanged. Empirically, this optional preconditioning can improve memory behavior during long-context prefill (see the Supplement for detailed evidence and analysis). We then derive diagnostic signals from the resulting block-level attention mass $\mathbf{A}_h$ to decide whether and how to allocate extra computation.

Physical profiling as necessity signals. We compute three inexpensive profiling metrics from $\mathbf{A}_h$ that guide sparse attention decisions: (i) normalized entropy to quantify dispersion (as a proxy for uncertainty), (ii) top-κ concentration to quantify focus and veto unnecessary expansion, and (iii) mid-zone coverage to diagnose insufficient attention allocation to mid-range context.

1. **Normalized Entropy.** We compute the normalized Shannon entropy of the block-level attention distribution:

$$H(\mathbf{A}_h) = -\frac{1}{\ln N} \sum_{i=1}^{N} p_i \ln(p_i + \epsilon), \tag{2}$$

   where $p_i = \mathbf{A}_h[i] \big/ \sum_{j=1}^{N} \mathbf{A}_h[j]$ is the normalized attention mass of block i. The factor $\ln N$ normalizes $H(\mathbf{A}_h)$ to $[0, 1]$.

2. **Top-κ Concentration.** We define concentration as the cumulative mass of the top κ fraction of blocks:

$$C(\mathbf{A}_h) = \frac{\sum_{j \in \mathcal{S}_\kappa} \mathbf{A}_h[j]}{\sum_{i=1}^{N} \mathbf{A}_h[i]}, \tag{3}$$

   where $\mathcal{S}_\kappa$ contains indices of the largest $\lfloor \kappa \cdot N \rfloor$ entries in $\mathbf{A}_h$. A high $C(\mathbf{A}_h)$ indicates that attention mass is already sharply focused, serving as a veto signal against unnecessary global expansion.

3. **Mid-Zone Coverage.** To monitor insufficient allocation to the middle portion of the context, we compute the attention mass over the central interval (by default, 25%–75% of blocks):

$$M(\mathbf{A}_h) = \frac{\sum_{i=\lfloor 0.25N \rfloor}^{\lfloor 0.75N \rfloor} \mathbf{A}_h[i]}{\sum_{j=1}^{N} \mathbf{A}_h[j]}. \tag{4}$$

   Low $M(\mathbf{A}_h)$ indicates that mid-range context receives little attention mass, which is undesirable for retrieval-intensive workloads.

3.4 Task-conditional Routing

Based on the task prior and profiling signals, we route each head to one of three mutually exclusive policies: No-Expand, Global-TopK, and Coverage-Sampling. This avoids compounding overhead and reduces unintended interactions between policies.

1. **No-Expand.** For structure-dominant inputs, we prioritize noise suppression over recall and disable any dynamic expansion. The attention mask is strictly constrained to the base topology:

$$\mathcal{M} = \mathcal{M}_{VS}. \tag{5}$$

2. **Global-TopK.** For reasoning-intensive inputs, we allow limited global augmentation only when necessary. We trigger Global-TopK if the head exhibits both diffuse attention (high entropy) and weak focus (low concentration):

$$\alpha_{\text{global}} = \mathbb{I}(H(\mathbf{A}_h) > \tau_{\text{ent}}) \wedge \mathbb{I}(C(\mathbf{A}_h) < \tau_{\text{conc}}). \tag{6}$$

   If $\alpha_{\text{global}} = 1$, we append a small set of globally selected blocks:

$$\mathcal{M} = \mathcal{M}_{VS} \cup \text{TopK}_{\text{Global}}(\mathbf{A}_h), \tag{7}$$

   otherwise we keep $\mathcal{M} = \mathcal{M}_{VS}$. The concentration condition acts as a veto to reduce noise from over-expansion.

3. **Coverage-Sampling.** For retrieval-intensive inputs, the main risk is insufficient long-range coverage. We trigger Coverage-Sampling when mid-zone coverage is low and concentration is not high:

$$\alpha_{\text{cov}} = \mathbb{I}(M(\mathbf{A}_h) < \tau_{\text{mid}}) \wedge \mathbb{I}(C(\mathbf{A}_h) \leq \tau_{\text{mod}}). \tag{8}$$

   If triggered, we inject additional landmarks in the mid-zone using an adaptive stride S_h:

$$S_h = \max\left(S_{\min}, \left\lfloor S_{\text{base}} \cdot \frac{M(\mathbf{A}_h)}{\tau_{\text{mid}}} \right\rfloor\right). \tag{9}$$

   The final mask incorporates interval-sampled landmarks:

$$\mathcal{M} = \mathcal{M}_{VS} \cup \text{Sample}_{\text{Interval}}(\text{Mid-Zone}, S_h). \tag{10}$$

Cluster-aware extension We also study a cluster-aware refinement for the two expansion-enabled branches, Global-TopK and Coverage-Sampling, where high-response key centers are expanded into locally coherent causal candidate blocks. Rather than replacing the original compensation rule, this variant adopts a dual-branch allocation, assigning 70% of the flat block budget to cluster-aware candidates and retaining 30% from the original score-cover or sampling branch. To keep the main algorithm and evaluation protocol focused, we present this extension as an appendix-level analysis in Appendix A and Appendix B.

Overall, our methodology uses task-conditioned dispatch to constrain the space of sparse attention behaviors, and relies on head-wise physical profiling to trigger No-Expand, Global-TopK, or Coverage-Sampling only when clear necessity signals are observed. This adaptive design tailors sparse index selection to the attention patterns of individual heads, allowing us to efficiently construct the sparse mask $\mathcal{M}$ for prefill computation. By focusing additional computation on a subset of heads and high-mass interactions implied by $\mathbf{A}_h$, we balance computational efficiency with attention quality in practice. Empirically, we observe that not all heads are activated for augmentation or sampling, which further concentrates compute on the most relevant token interactions under the predefined selection rules.

4 Experiments

In this section, we provide an overview of the models, datasets and results of our experiments to evaluate AdaCal's perfomance and capabilities of various long-context language moedels.

Table 1: Performance comparison of different methods on various models and tasks on RULER. The **bold**/underlined numbers indicate the first/second highest value in each column.

Models	Methods	4k	8k	16k	32k	64k	128k	Avg
LLaMA	Full-attn	95.67	93.75	93.51	87.50	84.86	79.57	89.14
	Streaming LLM	**95.91**	93.99	75.96	45.43	24.76	31.49	61.26
	MInference	**95.91**	**94.23**	93.75	86.54	83.90	62.26	86.10
	FlexPrefill	95.43	**94.23**	**94.95**	89.91	82.93	78.12	89.26
	Ours	95.67	93.99	94.23	**91.11**	**85.34**	**80.05**	**90.06**
Yi	Full-attn	93.03	83.89	81.73	70.19	67.07	64.18	76.68
	Streaming LLM	**93.27**	81.73	63.70	51.92	41.59	36.06	61.38
	MInference	92.55	84.61	80.53	**71.40**	63.70	60.10	75.48
	FlexPrefill	92.55	**87.02**	83.89	70.19	60.58	59.61	75.64
	Ours	92.79	82.69	**83.90**	70.91	**65.15**	**62.74**	**76.36**
Qwen	Full-attn	91.83	85.82	81.01	80.53	52.40	17.55	68.19
	Streaming LLM	91.59	86.78	58.65	40.14	23.32	11.78	52.04
	MInference	**92.07**	86.54	**82.69**	78.13	48.56	11.54	66.59
	FlexPrefill	90.87	87.50	81.25	80.29	**54.33**	17.31	68.59
	Ours	**92.07**	**88.46**	**82.69**	**80.77**	54.09	**18.03**	**69.35**

Table 2: Performance comparison on InfiniteBench across different model families.

Model	Method	En. Sum	En. QA	En. MC	En. Dia	Zh. QA	Code Debug	Math Find	Retr. PassKey	Retr. Number	Retr. KV	Avg
LLaMA	Full-attn	32.32	25.96	69.43	20.50	32.66	18.02	25.14	99.32	99.66	62.60	48.56
	Streaming LLM	30.47	10.10	40.17	9.50	22.56	**23.10**	16.57	2.71	5.93	0.00	16.11
	MInference	30.91	22.73	65.07	14.00	27.74	21.32	25.43	59.83	76.44	21.00	36.45
	FlexPrefill	31.55	25.49	69.43	**19.00**	**35.22**	20.81	**30.86**	98.64	**99.66**	44.60	47.53
	Ours	**33.08**	**26.31**	**70.13**	18.50	33.20	19.29	27.14	**99.15**	**99.66**	**59.20**	**48.57**
Yi	Full-attn	8.00	11.19	62.45	1.00	16.83	23.35	25.43	100.00	100.00	15.00	36.32
	Streaming LLM	6.83	10.46	43.67	**3.00**	15.48	20.30	27.71	12.71	49.66	3.40	19.32
	MInference	7.63	10.98	59.39	1.50	17.29	**25.38**	**26.00**	**100.00**	**100.00**	14.60	36.28
	FlexPrefill	7.77	**12.45**	**65.94**	1.50	16.98	19.29	25.14	**100.00**	95.93	14.20	35.92
	Ours	**8.97**	11.20	62.01	1.00	**17.64**	24.62	25.14	**100.00**	**100.00**	**15.00**	**36.56**
Qwen	Full-attn	19.15	6.26	45.41	13.00	11.60	19.29	28.00	62.37	73.22	0.00	27.83
	Streaming LLM	**18.76**	6.37	44.10	11.50	11.50	**23.35**	22.29	33.39	24.07	0.00	19.53
	MInference	11.85	5.04	26.64	7.50	12.00	15.23	28.29	66.10	55.25	0.00	22.79
	FlexPrefill	17.64	**6.64**	43.67	**15.00**	**12.36**	20.81	**31.14**	**92.20**	59.83	0.00	**29.93**
	Ours	18.35	6.37	**44.98**	13.50	11.25	19.54	30.00	65.93	**73.56**	0.00	28.35

4.1 Settings

Models To evaluate the universality and robustness of AdaCal, we conduct experiments on four representative state-of-the-art LLMs that support long-context windows. These include: 1) **Meta-Llama-3.1-8B-Instruct-128k**Dubey et al. [2024], which supports a context window of up to 128k tokens; 2) **Yi-9B-200K** Young et al. [2024], known for its strong performance in long-context retrieval and reasoning; 3) **Qwen2-7B-Instruct-128k** Team et al. [2024], which also supports a 128k context window.By selecting diverse models, we ensure that our findings are generalized across different architectures and pre-training distributions.

Datasets We assess the long-context capabilities on two challenging benchmarks designed to test varying levels of complexity. **RULER** Hsieh et al. [2024], a synthetic benchmark covering retrieval, variable tracking, common words extraction, and question answering across 4K–128K contexts; and **InfiniteBench** Zhang et al. [2024], a realistic long-context benchmark with 10 tasks spanning retrieval,

Table 3: Performance of different ablation methods using Meta-LlaMA-3.1-8B-Instruct-128k on RULER.

Methods	4k	8k	16k	32k	64k	128k	Avg
Only vs	95.67	93.99	**94.79**	86.54	85.34	77.88	89.04
Only Glo.	**95.91**	93.99	92.55	88.46	81.97	71.15	87.34
Only Sam.	95.67	93.75	94.47	87.74	85.10	78.13	89.14
Ours	95.67	**94.47**	94.23	**91.11**	**85.34**	**80.05**	**90.15**

mathematical reasoning, code debugging, and long-document summarization.This allows us to verify AdaCal's effectiveness in real-world scenarios where noise and dependencies are unstructured.

Baselines We compare AdaCal against the following state-of-the-art attention mechanisms. 1) **Full Attention** Dao [2023] Serves as the upper bound for accuracy and the lower bound for speed; 2) **StreamingLLM** Xiao et al. [2023]: A static sparse method that retains initial attention sinks and local sliding windows; 3) **MInference** Jiang et al. [2024] A dynamic method that utilizes off-line computed static sparse patterns. 4) **FlexPrefill** Lai et al. [2025]: A dynamic sparse attention mechanism that uses Jensen-Shannon divergence to switch between query-aware and vertical-slash patterns. This serves as our strongest baseline for dynamic routing efficiency.

Implementation Details All experiments are conducted on a single NVIDIA H800 (80GB) GPU. To ensure rigorous reproducibility and computational efficiency, we build our system upon the engineering infrastructure of FlexPrefill. Specifically, we inherit its optimized Triton-based sparse kernels and memory management protocols to guarantee a fair baseline. However, distinct from FlexPrefill's statistical estimation approach, our mechanism introduces a novel profile-guided mechanism. We set the block size $b = 128$, the number of blocks is $N = L/b$ and the cumulative attention threshold $\gamma = 0.95$ for all settings. Crucially, to minimize profiling overhead, we compute the physical fingerprints exclusively on the sparse blocks pre-selected by the base Vertical-Slash pattern, rather than scanning the full attention matrix. Empirically, we use $\tau_{\text{ent}} = 0.22$ and the default thresholds $\tau_{\text{conc}} = 0.9$ and $\tau_{\text{mid}} = 0.2$. The complete configuration is provided in E. A sensitivity analysis of the routing thresholds is provided in Appendix F. Finally, we use greedy decoding for all evaluations. Finally, we use greedy decoding for all evaluations.

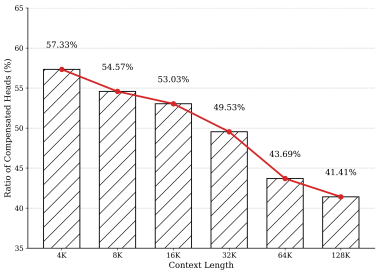


Figure 4: Adaptive Activation Dynamics.

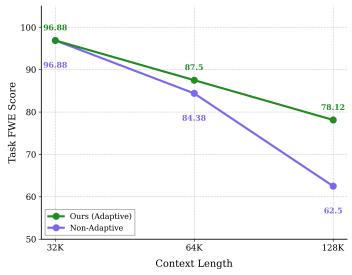


Figure 5: Adaptive vs. Non-Adaptive Allocation under Iso-Budget.

4.2 Main Results

In this section, we present a comparative analysis of our proposed Profile-Guided Adaptive Sparse Attention mechanism against various attention mechanisms, including Full Attention, StreamingLLM, MInference, and FlexPrefill. This comparison demonstrates our approach's superior efficiency compared to existing methods.

RULER To demonstrate the efficacy of our approach for long-context large language models, we conduct comprehensive evaluations on the RULER benchmark. As shown in Table 1, as context length increases, static methods like StreamingLLM struggle to maintain efficacy, while dynamic baselines like FlexPrefill exhibit performance volatility due to heuristic-based estimation. In contrast, our method consistently preserves model performance across varying context lengths. This

advantage becomes particularly pronounced beyond the 16k token threshold, where our mechanism not only outperforms other sparse methods but also surpasses the Full Attention oracle in specific scenarios,demonstrating its effectiveness on long-context tasks.

Infinite Bench The evaluation results on InfiniteBench, presented in Table 2, further validates our mechanism's potential on realistic tasks. Remarkably, our method achieves higher average scores than the Full Attention oracle across all three models. This advantage is particularly pronounced on the challenging Retr.KV task, where our method secures a score of 59.20, significantly outperforming the strong baseline FlexPrefill.

Complexity and efficiency. AdaCal shares the same block-sparse complexity regime as existing sparse attention methods and remains substantially more efficient than full attention for long contexts. By reducing the active block ratio r, it computes attention only over selected blocks, leading to an approximate time complexity of $O(H \cdot r \cdot L^2 \cdot d)$ without constructing the full $N \times N$ attention table. Its memory cost is comparable to other sparse frameworks, and the added physical feature extraction introduces negligible overhead on top of the optimized FlexPrefill implementation. See C for more details.Additional comparisons of wall-clock latency, throughput, and peak GPU memory are reported in Tables 8, 9, and 10, respectively.

4.3 Ablations

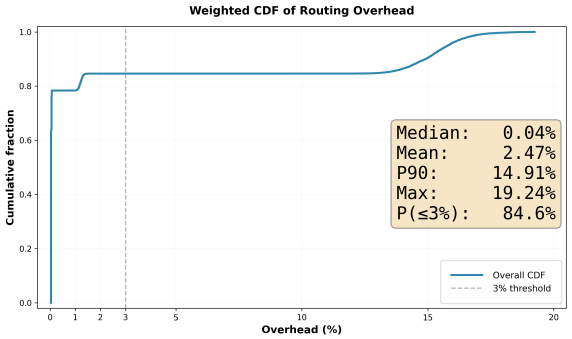


Figure 6: Weighted CDF of adaptive-selection overhead in block selecting. Overhead is defined as $t_{\text{adaptive}}/t_{\text{total}}$, where t_{adaptive} includes entropy, concentration, and routing logic, and t_{total} is the total function time. Most runs incur small overhead ($P(\leq 3\%) = 84.6\%$), with a tail up to 19.24%.

Contributions of each manifold. To isolate the role of each manifold, we evaluate three ablations: Only VS applies the No-Expand path to all tasks, Only Glo. universally enables the Uncertainty-Aware trigger, and Only Sam. uses the Coverage-Aware trigger for every task. As shown in Table 3, Only VS performs well at 16K but degrades at 128K due to limited receptive fields; Only Glo. performs worst at 128K, indicating that unfiltered global expansion introduces detrimental noise; and Only Sam. improves coverage but still falls behind the full mechanism. By synergizing these paths, our method achieves the highest average score, empirically validating the effectiveness of the Constraint-then-Calibrate mechanism in balancing sparsity and recall.

Adaptive Budgeting and Noise Mitigation. Figure 4 shows that the Compensate activation ratio decreases from 57.33% at 4K to 41.41% at 128K. This length-dependent selectivity supports our noise-aware design: as contexts grow, the entropy-based trigger restricts global expansion to fewer critical heads, reducing distractor accumulation in ultra-long inputs.

Necessity of Adaptive Selection. We compare AdaCal with a fixed-order baseline under the same Iso-Budget setting. As shown in Figure 5, the fixed allocation strategy collapses at 128K, while our profile-guided routing maintains strong performance. This confirms that simply increasing the active budget is insufficient; adaptive head selection is necessary to mitigate noise and recover relevant information.

Routing overhead distribution. We profile the runtime cost of our routing logic inside selecting blocks. We measure t_{total} (total function time) and $t_{\text{adaptive}} = t_{\text{entropy}} + t_{\text{conc}} + t_{\text{judge}}$, and report

Overhead $= (t_{\text{adaptive}}/t_{\text{total}}) \times 100\%$. Figure 6 shows that the overhead is typically small: 84.6% of runs stay within 3%, while a small tail reaches 19.24%. This indicates that our profiling-based selection is lightweight for most inputs and incurs higher cost only occasionally.

5 Conclusion

In this paper, we introduced AdaCal, a profile-guided adaptive sparse attention mechanism for efficient long-context prefill. AdaCal combines coarse task dispatch with head-wise physical profiling, using entropy, concentration, and mid-zone coverage from a lightweight probe topology to trigger three task-conditional policies (No-Expand, Global-TopK, Coverage-Sampling). Experiments on three long-context LLMs with RULER and InfiniteBench show improved robustness over strong static and dynamic baselines up to 128K while remaining efficient. Future work will extend the profiling-driven design to decoding-time optimizations and multimodal long-sequence modeling. Overall, AdaCal provides a practical, adaptive solution to balance efficiency and accuracy in long-context processing. We believe it offers a promising direction for building scalable and responsive large language models.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (Nos. 62576159 and 61402220), the Natural Science Foundation of Hunan Province, China (Nos. 2025JJ50384 and 2022JJ30495), and the Postgraduate Scientific Research Innovation Project of Hunan Province (No. CX20251443).

References

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

Avi Caciularu, Matthew E Peters, Jacob Goldberger, Ido Dagan, and Arman Cohan. Peek across: Improving multi-document modeling via cross-document question-answering. *arXiv preprint arXiv:2305.15387*, 2023.

Yapei Chang, Kyle Lo, Tanya Goyal, and Mohit Iyyer. Booookscore: A systematic exploration of book-length summarization in the era of llms. *arXiv preprint arXiv:2310.00785*, 2023.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models. *arXiv preprint arXiv:2309.12307*, 2023.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

Yichuan Deng, Zhao Song, and Chiwun Yang. Attention is naturally sparse with gaussian distributed input. *CoRR*, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv e-prints*, pages arXiv–2407, 2024.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering. *arXiv preprint arXiv:1907.09190*, 2019.

Yao Fu. Challenges in deploying long-context transformers: A theoretical peak performance analysis. *arXiv preprint arXiv:2405.08944*, 2024.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

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.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Wojciech Kryściński, Nazneen Rajani, Divyansh Agarwal, Caiming Xiong, and Dragomir Radev. Booksum: A collection of datasets for long-form narrative summarization. In *Findings of the association for computational linguistics: EMNLP 2022*, pages 6536–6558, 2022.

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.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.

Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. Fast attention over long sequences with dynamic sparse flash attention. *Advances in Neural Information Processing Systems*, 36:59808–59831, 2023.

Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. Quality: Question answering with long input texts, yes! In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5336–5358, 2022.

Siddhant Porwal, Laxmi Bewoor, and Vivek Deshpande. Transformer based implementation for automatic book summarization. *arXiv preprint arXiv:2301.07057*, 2023.

Qwen Team et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2(3), 2024.

Chong Wang, Jian Zhang, Yebo Feng, Tianlin Li, Weisong Sun, Yang Liu, and Xin Peng. Teaching code llms to use autocompletion tools in repository-level code generation. *ACM Transactions on Software Engineering and Methodology*, 34(7):1–27, 2025.

Yanlin Wang, Yanli Wang, Daya Guo, Jiachi Chen, Ruikai Zhang, Yuchi Ma, and Zibin Zheng. Rlcoder: Reinforcement learning for repository-level code completion. *arXiv preprint arXiv:2407.19487*, 2024.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Guoyin Wang, Heng Li, Jiangcheng Zhu, Jianqun Chen, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020.

Peiyuan Zhang, Yongqi Chen, Haofeng Huang, Will Lin, Zhengzhong Liu, Ion Stoica, Eric P Xing, and Hao Zhang. Faster video diffusion with trainable sparse attention. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. ∞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, 2024.

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

A Cluster-aware top-k

To enhance the coverage of locally coherent evidence regions in long-context sparse attention, we further introduce a cluster-aware top-k selection mechanism. Unlike directly performing global top-k selection over all causal block pairs, this mechanism first aggregates block-level attention scores along the query dimension to obtain a global saliency estimate for each key block. Then, multiple high-response centers are selected along the key axis, and each center is expanded with a fixed local radius, yielding a set of cluster-induced key columns. Based on these key columns, we enumerate all query-key block pairs that satisfy the causal constraint and construct a cluster-induced causal candidate pool. Finally, the model does not forcibly retain all block pairs associated with the centers and their neighboring regions; instead, it performs top-k selection within this candidate pool according to block-level attention weights. This design enables the sparse pattern to prioritize multi-center, high-response evidence regions while avoiding the conversion of local neighborhood expansion into a rigid group-binding selection rule.

This process is mainly controlled by two parameters. First, key_budget determines the maximum number of key blocks retained from the key-column saliency profile for multi-center selection and local neighborhood expansion. When an integer budget is not explicitly specified, we adopt a ratio-based formulation:

$$\texttt{key_budget} = \max(1, \lfloor B \cdot \rho_k \rfloor) .$$

where B denotes the number of key blocks and ρ_k corresponds to cluster_topk_key_budget_ratio, which is set to 0.0625 by default. Therefore, key_budget controls the scale of the candidate region along the key dimension, rather than the final number of retained flat blocks.

Second, cluster_quota_ratio controls the allocation of the total flat block budget between the cluster-induced candidate pool and the original score-cover retained set. Given the total budget flat_budget, we define

$$\begin{aligned} \text{cluster_quota} &= \lfloor \text{flat_budget} \cdot \text{cluster_quota_ratio} \rfloor , \\ \text{original_quota} &= \text{flat_budget} - \text{cluster_quota}. \end{aligned}$$

By default, cluster_quota_ratio=0.7, meaning that approximately 70% of the budget is used to select high-scoring block pairs from the cluster-induced candidates according to block-level attention weights, while the remaining 30% is supplemented by the original score-cover branch. This dual-branch budget allocation enables the method to enhance the coverage of multi-center salient evidence regions while preserving the robustness of the original global selection strategy.

To analyze the effects of these two parameters, we further conduct a parameter sensitivity study4. Specifically, key_budget_ratio mainly affects the coverage range and structure of the cluster candidate pool, whereas cluster_quota_ratio mainly determines the degree to which the final sparse pattern relies on the cluster branch. By comparing the performance under different parameter combinations, we examine the trade-off between recalling salient evidence and suppressing noisy blocks, and evaluate the stability of the default configuration across different long-context tasks.

The cluster-aware mechanism studied in this appendix is not limited to the Global-TopK branch, but serves as a general refinement for compensated sparse block construction in AdaCal. For Global-TopK, it enriches score-based compensation with cluster-induced candidate pools; for Coverage-Sampling, it can guide mid-zone compensation toward locally coherent high-response key regions. Rather than replacing the original compensation rule, the method adopts a dual-branch budget allocation: by default, 70% of the flat block budget is assigned to cluster-aware candidates, while the remaining 30% is retained from the original branch. This design improves the coverage of multi-center salient regions while preserving the robustness of the original sparse selection strategy. Since this extension has not yet been fully evaluated across all tasks, models, and context lengths, we keep it as an appendix-level analysis rather than incorporating it into the main AdaCal algorithm.

CWE and FWE are two representative tasks from the RULER benchmark for evaluating long-context retrieval behavior. As shown in Table 4, FWE is relatively insensitive to the tested parameter configurations, maintaining stable performance across different values of key_budget_ratio and cluster_quota_ratio. By contrast, CWE shows more pronounced variations, making it a more discriminative task for hyperparameter selection. We therefore choose the default configuration mainly according to the CWE results. Specifically, we set key_budget_ratio to 0.0625 $(1/16)$ and cluster_quota_ratio to 0.7, which yields the best CWE score while preserving competitive and stable performance on FWE.

Table 4: Sensitivity analysis of cluster-aware top-k selection parameters.

Parameters		Result	
key_budget_ratio	cluster_quota_ratio	CWE	FWE
0.03125 (1/32)	0.7	46.88	96.88
0.0625 (1/16)	0.7	50.00	96.88
0.0625 (1/16)	0.8	50.00	96.88
0.125 (1/8)	0.5	43.75	96.88
0.125 (1/8)	0.7	43.75	96.88
0.25 (1/4)	0.5	40.62	96.88
0.25 (1/4)	0.7	46.88	100.00

Table 5: Performance of different way in selecting compensated blocks using Meta-LlaMA-3.1-8B-Instruct-128k on RULER.

Methods	4k	8k	16k	32k	64k	128k	Avg
key_budget	95.67	94.23	94.23	89.66	85.57	79.32	89.78
cluster-aware top-k	95.67	93.99	94.71	90.14	86.06	79.32	90.06

B Top-k vs Cluster-aware top-k

During our experiments, we observe that important blocks in long-context attention are often not purely isolated, but tend to concentrate around locally coherent key regions. This motivates the use of cluster-aware selection for compensated block construction. Instead of independently selecting high-scoring block pairs as in vanilla top-k, cluster-aware top-k first identifies multiple high-response key regions and expands their local neighborhoods into an enriched candidate pool, while still selecting the final compensated blocks according to block-level attention weights. This design enables the sparse pattern to prioritize multi-center, high-response evidence regions while avoiding the conversion of local neighborhood expansion into a rigid group-binding selection rule.

We conduct experiments on the RULER benchmark using LLaMA on NVIDIA A800 GPUs, and the results in Table 5 show that cluster-aware top-k outperforms vanilla top-k under nearly all tested context lengths, demonstrating the effectiveness of incorporating local cluster structure into compensated block selection.

C Time, Space complexity and Speed

C.1 Computational Complexity

Time6 and Space7: In terms of time and space, both FlexPrefill and AdaCal share a common backbone of $O(hbnd) + O(\|P\|b^2 d)$ and $O(hbn)$. Under the setting where $\|P\|$ is much smaller than the number of dense block pairs, the two are of the same order and both outperform the traditional Full Attention, which requires $O(hn^2 d)$ in time and $O(hn^2)$ in explicit logits space.

We let $|P|$ denote the total number of causal block pairs (i, j) actually evaluated across attention heads during prefill after block sparsification, i.e., the number of query–key block tiles processed by the sparse block-wise attention kernel.

Speed: AdaCal is built upon the FlexPrefill engineering framework, reusing its highly optimized Triton kernels for core operations such as block attention score computation. This ensures that our baseline performance is aligned with current SOTA sparse attention methods. As shown in Fig. 6, the physical feature extraction is lightweight (median overhead of 0.04%), and could be further accelerated by implementing specialized Triton kernels in the future.

C.2 Practical Prefill Efficiency

Evaluation protocol. We evaluate prefill efficiency on a single NVIDIA H200 GPU using Llama-3.1-8B-Instruct across context lengths from 8K to 128K. For each input, we perform two warm-up runs

Table 6: Comparison of computational complexity. Both FlexPrefill and AdaCal share the same asymptotic backbone complexity and outperform the full attention baseline in sparse scenarios.

Method	**Attention Time Complexity (Dominant term, single-layer prefill)**	**Comparison with Full Attention**
Full Attention	$O(hn^2d)$	Baseline (n: seq. length, h: num heads, d: head dim)
FlexPrefill	$O(hbnd) + O(\|P\|b^2d)$	When $\|P\|$ is much smaller than dense block pairs ht^2, it outperforms $O(hn^2d)$
AdaCal	$O(hbnd) + O(\|P\|b^2d)$	When $\|P\|$ is much smaller than dense block pairs ht^2, it outperforms $O(hn^2d)$

Table 7: Comparison of peak memory space for attention-related operations. While Flash-style implementations avoid the explicit $\Theta(hn^2)$ logits, sparse methods (FlexPrefill, AdaCal) introduce an $O(hbn)$ workspace for block processing, trading off the quadratic sequence dependency for a linear one.

Method	**Peak Attention Space (Dominant Term)**	**Comparison with Full Attention**
Full Attention	$O(hn^2)$	Explicit logits grow quadratically with n^2; Flash path avoids this but has no $O(hbn)$ routing tensor overhead.
FlexPrefill	$O(hbn)$ + $O(hnd)$ +$O(\|P\|)$ + optional $O(t^2)$	Avoids the $O(hn^2)$ logits matrix.
AdaCal	$O(hbn) + O(hnd) + O(\|P\|)$ + optional $O(t^2)$	Avoids the $O(hn^2)$ logits matrix.

followed by ten measured runs. Tables 8, 9, and 10 report prefill latency, aggregate prefill throughput, and process-level peak GPU memory, respectively. Each cell also includes the corresponding RULER score to jointly characterize efficiency and task performance.

Results. Compared with FlexPrefill, AdaCal reduces prefill latency by 7.3%–11.2% across the evaluated context lengths, while achieving higher throughput. At 128K, latency decreases from $29{,}435.0$ ms to $27{,}010.8$ ms, and throughput increases from 4.43 to 4.83 KTok/s. The corresponding RULER score improves from 78.12 to 80.05.

Peak GPU memory is unchanged at the reported precision from 8K to 64K and increases from 47.07 to 47.83 GiB at 128K, corresponding to an increase of approximately 1.6%. These results demonstrate lower prefill latency and higher throughput than FlexPrefill with comparable peak GPU memory in the evaluated setting. Efficiency comparisons across all methods should also account for their task accuracy, as reported in the main results.

D GPU memory with/without SVD

On a single 40 GB GPU, we observed that enabling SVD-with-reconstruction allows the long-context run to finish, while disabling it leads to CUDA OOM. To understand this discrepancy, we profile per-layer GPU memory using two allocator metrics.

We measure peak allocated memory $M_{\text{alloc}}^{\text{peak}}$ (active tensor footprint) and peak reserved memory $M_{\text{res}}^{\text{peak}}$ (PyTorch CUDA caching allocator pool size), computed as torch.cuda.max_memory_allocated()/1024^2 and torch.cuda.max_memory_reserved()/1024^2 (MiB), respectively. We reset peak statistics at the beginning of each layer's prefill attention call.

Table 11 Table 12shows that SVD-with-reconstruction does not reduce tensor footprint: $M_{\text{alloc}}^{\text{peak}}$ increases by $\sim$126 MiB per layer due to additional temporary buffers. However, it reduces $M_{\text{res}}^{\text{peak}}$ by 1.8–3.5 GiB, indicating that SVD introduces additional intermediate tensors and synchronization points, which reshapes the allocation/freeing schedule and improves reuse of large memory blocks; as a result, the caching allocator requires a smaller reserved pool to satisfy peak demands, reducing OOM risk.

Table 8: Latency (ms) ↓ / RULER Score ↑. Each cell is formatted as prefill latency (ms) / RULER score.

Mode	Method	8K	16K	32K	64K	128K
Offline	StreamingLLM	294.3 / 93.99	680.9 / 75.96	1,503.3 / 45.43	3,163.3 / 24.76	6,444.2 / 31.49
Offline	MInference	1,069.2 / 94.23	1,433.1 / 93.75	2,190.2 / 86.54	3,793.6 / 83.90	6,734.6 / 62.26
Online	FlexPrefill	1,153.9 / **94.23**	2,431.4 / **94.95**	5,139.0 / 89.91	12,009.4 / 82.93	29,435.0 / 78.12
Online	AdaCal (Ours)	**1,069.7** / 93.99	**2,157.9** / 94.23	**4,567.1 / 91.11**	**11,048.0 / 85.34**	**27,010.8 / 80.05**

Table 9: Throughput (kTok/s) ↑ / RULER Score ↑. Each cell is formatted as aggregate prefill throughput / RULER score.

Mode	Method	8K	16K	32K	64K	128K
Offline	StreamingLLM	26.21 / 93.99	23.26 / 75.96	21.27 / 45.43	20.37 / 24.76	20.24 / 31.49
Offline	MInference	7.21 / 94.23	11.04 / 93.75	14.60 / 86.54	16.99 / 83.90	19.37 / 62.26
Online	FlexPrefill	6.68 / **94.23**	6.51 / **94.95**	6.22 / 89.91	5.37 / 82.93	4.43 / 78.12
Online	AdaCal (Ours)	**7.21** / 93.99	**7.33** / 94.23	**7.00 / 91.11**	**5.83 / 85.34**	**4.83 / 80.05**

Table 11 and Table 12 show that SVD with reconstruction does not reduce the tensor footprint: $M_{\text{alloc}}^{\text{peak}}$ increases by approximately $126\,\text{MiB}$ per layer due to additional temporary buffers. However, it reduces $M_{\text{res}}^{\text{peak}}$ by 1.8–$3.5\,\text{GiB}$, indicating that SVD introduces additional intermediate tensors and synchronization points. This reshapes the allocation/freeing schedule and improves the reuse of large memory blocks; as a result, the caching allocator requires a smaller reserved pool to satisfy peak demands, thereby reducing the risk of OOM.

E Threshold Configuration for Profiling and Routing

Table 13 summarizes the profiling thresholds used in our experiments.

Our thresholds are empirical values based on attention statistics. Identical parameters achieve significant gains on Llama-3.1 and Yi, proving robust cross-architecture generalization.

Due to Qwen's RoPE scaling, it exhibits entropy shifts in attention distributions across varying context lengths. We introduced length-adaptive scheduling to ensure optimal sparse precision across the entire length range. Specifically, we apply a deterministic length-adaptive schedule for $L \leq 16\text{K}$ to stabilize short-context behavior, and revert to the same fixed thresholds for $L > 16\text{K}$. All constants are fixed across tasks and runs.

F Sensitivity to Routing Thresholds

Evaluation protocol. We conduct a one-at-a-time threshold sweep on Llama across all tasks and context lengths in RULER. In each experiment, only one threshold is varied, while all other thresholds are fixed at their default values.

We report the following measures:

- **RULER Span and Branch Span:** the max–min score ranges across the tested threshold values for all RULER tasks and the affected routing branch, respectively.
- **Active-Block Ratio:** the proportion of retained attention blocks.
- **Expansion-Head Ratio:** the proportion of heads triggering block expansion.

Overall sensitivity. As shown in Table 14, the overall RULER spans range from 0.04 to 0.32 points in the one-at-a-time sweeps, indicating limited sensitivity of aggregate performance within the tested ranges. The routing statistics nevertheless respond to threshold changes. In particular, varying τ_{conc}^{G} changes the expansion-head ratio from 37.25% to 65.36% and the active-block ratio from 15.96% to

Table 10: Peak Memory (GiB) ↓ / RULER Score ↑. Each cell is formatted as process-level peak GPU memory / RULER score.

Mode	Method	8K	16K	32K	64K	128K
Offline	StreamingLLM	19.30 / 93.99	21.74 / 75.96	30.25 / 45.43	30.03 / 24.76	47.02 / 31.49
Offline	MInference	19.21 / 94.23	21.55 / 93.75	29.88 / 86.54	31.16 / 83.90	49.02 / 62.26
Online	FlexPrefill	19.22 / 94.23	21.55 / 94.95	29.91 / 89.91	29.46 / 82.93	47.07 / 78.12
Online	AdaCal (Ours)	19.22 / 93.99	21.55 / 94.23	29.91 / 91.11	29.46 / 85.34	47.83 / 80.05

Table 11: Peak allocated memory (MB) at selected layers.

Layer	SVD(64)	noSVD
0	17652.66	17525.62
1	18033.62	17906.59
2	18160.06	18033.57
30	21716.28	21589.42

21.73%. Thus, relatively stable aggregate performance can coexist with changes in routing behavior and block allocation.

Effect of the global concentration threshold. Among the tested thresholds, τ^{G}_{conc} has a more pronounced effect on branch-level performance. As shown in Table 15, the default value of 0.92 achieves the highest Global Branch Avg. score of 75.26, compared with 73.96 at 0.87 and 73.18 at 0.97. The lower threshold triggers fewer expansions, whereas the higher threshold activates more attention blocks without improving accuracy. These results support 0.92 as a reasonable operating point that balances branch performance and block usage within the tested range.

G Attention Heatmaps

To further motivate the task-conditional design of AdaCal, we visualize full-attention heatmaps7 across representative tasks. Notably, these heatmaps are obtained before applying any sparse attention mechanism. We observe that different tasks already induce qualitatively different attention distributions under full attention, indicating that the desired attention topology is task-dependent even when no sparsification constraint is imposed. This supports the use of a coarse, experience-driven dispatcher in AdaCal: rather than applying a universal sparse expansion rule, the dispatcher first groups inputs according to task characteristics and constrains the subsequent sparse attention policy accordingly. Within this task-conditioned regime, head-wise profiling and calibration can then be applied more safely and effectively.

H Pseudo-code of AdaCal

This section provides the detailed pseudo-code of AdaCal. We decompose the overall procedure into one main algorithm and three auxiliary subroutines, as shown in Algorithm 1–Algorithm 4. The main algorithm, Algorithm 1, first applies a coarse task-conditional dispatch to constrain the allowable sparse attention behavior, and then uses a lightweight Vertical-Slash probe to obtain block-level attention summaries. The task dispatch procedure is detailed in Algorithm 2. Based on the probe summaries, Algorithm 3 computes head-wise diagnostic signals, including normalized entropy, top-κ concentration, and mid-zone coverage. Finally, Algorithm 4 determines whether to keep the base topology or augment it through Global-TopK or Coverage-Sampling, and the resulting sparse index set is used for the final sparse attention computation. This design highlights the central principle of AdaCal: constrain sparse attention decisions with task priors first, and calibrate additional computation only when the head-wise physical profiles indicate clear necessity.

Table 12: Peak reserved memory (MB) at selected layers.

Layer	SVD(64)	noSVD
0	18384.0	18386.0
1	21054.0	21056.0
2	21054.0	21056.0
30	23724.0	25514.0

Table 13: Profiling threshold configuration used in our experiments, where L denotes the context length.

Model / regime	Thresholds
LLaMA, Yi (all L)	$\tau_{\text{ent}} = 0.22,\ \tau_{\text{conc}} = 0.90,\ \tau_{\text{mid}} = 0.20$
Qwen ($L > 16$K)	$\tau_{\text{ent}} = 0.22,\ \tau_{\text{conc}} = 0.90,\ \tau_{\text{mid}} = 0.20$
Qwen ($L \leq 16$K)	Length-adaptive $\tau_{\text{conc}}(L)$ and $\tau_{\text{mid}}(L)$
Base values	$\tau_{\text{conc}}^{\text{base}} = 0.90,\ \tau_{\text{mid}}^{\text{base}} = 0.20$
Scaling	$\alpha_{\text{conc}} = -0.05,\ \alpha_{\text{mid}} = 0.025$
Bounds	$\tau_{\text{conc}} \in [0.70, 0.98],\ \tau_{\text{mid}} \in [0.10, 0.35]$
Adaptive stride	$S_{\text{base}} = 4,\ S_{\min} = 2$

I Limitations

The current version of AdaCal mainly aims to validate the effectiveness of task-aware and profile-guided sparse attention, and two limitations remain. First, although the physical profiling stage introduces only lightweight diagnostic computations, the subsequent compensation process has not yet been fully implemented with a dedicated Triton kernel. Specifically, after computing entropy, concentration, and mid-zone coverage, the expansion decisions are currently realized through high-level indexing and mask operations, which may introduce avoidable overhead. Therefore, the current implementation may underestimate the potential efficiency of a kernel-fused version of AdaCal.

Second, the task dispatch module relies on coarse, experience-driven task grouping. This design is simple, training-free, and effective for validating our main hypothesis, but it may be suboptimal for unseen tasks or ambiguous task types whose sparse attention requirements do not clearly match the predefined categories. Future work will focus on Triton-level fusion of the compensation procedure and more adaptive task routing.

Table 14: One-at-a-time sensitivity analysis of AdaCal's routing thresholds on Llama. All other thresholds are fixed at their default values when varying the threshold in each row.

Threshold	Tested Values	RULER Span	Branch Span	Active-Block Ratio	Expansion-Head Ratio
Global Entropy τ_{ent}	$0.17/0.22/0.27$	**0.04**	0.26	18.67–19.00%	47.95–48.18%
Global Conc. τ^{G}_{conc}	$0.87/0.92/0.97$	**0.32**	2.08	15.96–21.73%	37.25–65.36%
Coverage Midpoint τ_{mid}	$0.15/0.20/0.25$	**0.24**	0.29	61.18–62.35%	81.42–89.13%
Coverage Conc. τ^{C}_{conc}	$0.85/0.90/0.95$	**0.12**	0.14	61.13–62.14%	54.84–56.09%

Table 15: Effect of the global concentration threshold τ^{G}_{conc} with $\tau_{\text{ent}} = 0.22$ fixed. The default configuration is highlighted in bold.

τ^{G}_{conc} ($\tau_{\text{ent}} = 0.22$ fixed)	Global Branch Avg.	Active-Block Ratio	Expansion-Head Ratio
0.87	73.96	15.98%	37.25%
0.92 (default)	**75.26**	**18.98%**	**48.17%**
0.97	73.18	21.73%	65.36%

Algorithm 1 AdaCal Sparse Attention

Require: $Q, K, V \in \mathbb{R}^{S \times d_h}$, task type t, block size b, thresholds $\tau_{\text{ent}}, \tau_{\text{conc}}, \tau_{\text{mid}}, \tau_{\text{mod}}$

Determine the task-constrained sparse policy
policy $\leftarrow$ TASK DISPATCH(t)

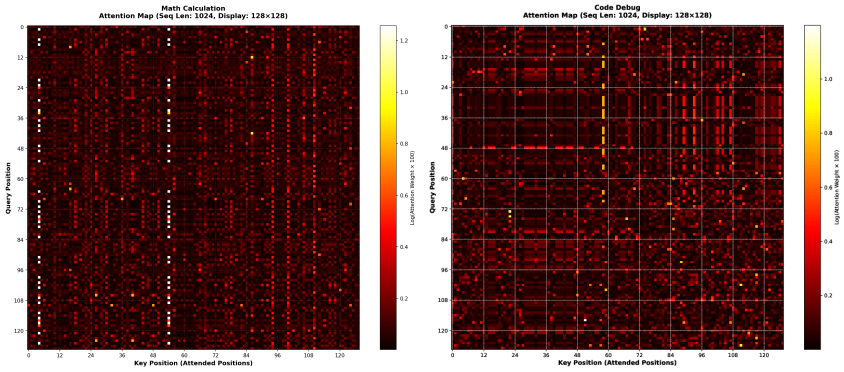
Build the base Vertical-Slash probe mask
$\mathcal{M}_{\text{VS}} \leftarrow$ VERTICAL SLASH MASK(Q, K, b)

Compute block-level attention summary under the probe
$\hat{A} \leftarrow \text{softmax}\left(QK^{\top}/\sqrt{d} + \mathcal{M}_{\text{VS}}\right)$
$A \leftarrow$ BLOCK AGGREGATE($\hat{A}, b$)

Compute physical profiles from the probe attention
$(H, C, M) \leftarrow$ PHYSICAL PROFILER(A)

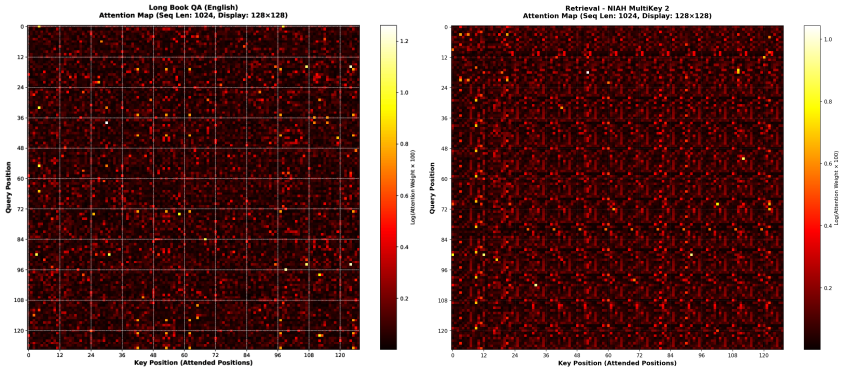
# Decide the sparse index set $\mathcal{S}$ based on policy and profiles
if policy == no_expand **then**
    $\mathcal{S} \leftarrow$ NO EXPAND INDEX($\mathcal{M}_{\text{VS}}$)
else if policy == global_topk **then**
    $\mathcal{S} \leftarrow$ GLOBAL TOPK INDEX($\mathcal{M}_{\text{VS}}, A, H, C, \tau_{\text{ent}}, \tau_{\text{conc}}$)
else if policy == coverage_sampling **then**
    $\mathcal{S} \leftarrow$ COVERAGE SAMPLING INDEX($\mathcal{M}_{\text{VS}}, A, M, C, \tau_{\text{mid}}, \tau_{\text{mod}}$)
end if

Compute the final sparse attention output
$y \leftarrow \mathcal{A}(Q, K, V, \mathcal{S})$
return y



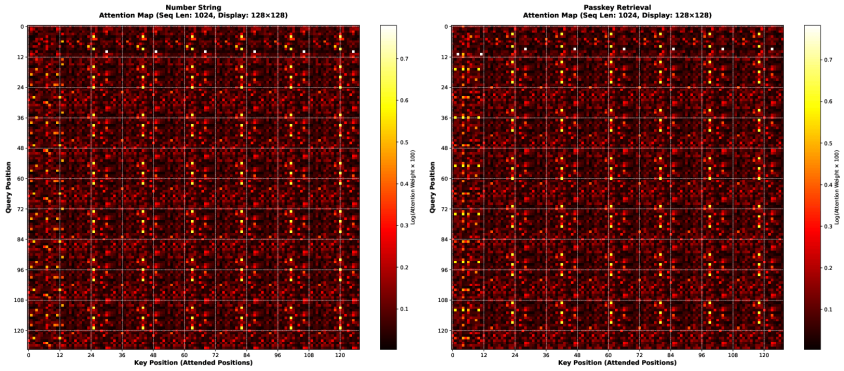
(1) `math_cal`.

(2) `code_debug`.



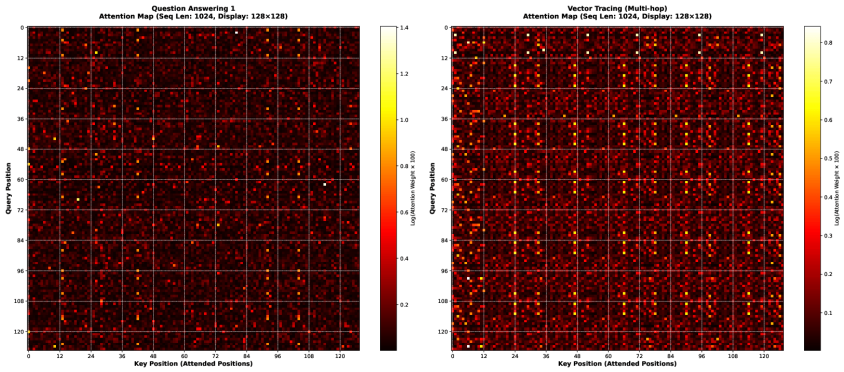
(3) `longbook_qa_eng`.

(4) `niah_multikey_2`.



(5) `number_string`.

(6) `passkey`.



(7) `qa_1`.

(8) `vt`.

Figure 7: Attention heatmaps of various tasks.

Algorithm 2 Task Dispatch

Require: task type t

# Assign a coarse task prior to constrain sparse attention
if $t \in \mathcal{T}_{\text{struct}}$ **then**
    policy $\leftarrow$ no_expand
else if $t \in \mathcal{T}_{\text{reason}}$ **then**
    policy $\leftarrow$ global_topk
else if $t \in \mathcal{T}_{\text{retrieval}}$ **then**
    policy $\leftarrow$ coverage_sampling
else
    policy $\leftarrow$ no_expand
end if
return policy

Algorithm 3 Physical Profiler

Require: block-level attention mass $A \in \mathbb{R}^N$, top fraction κ, small constant ϵ

Normalize block-level attention mass
$p_i \leftarrow A[i] / \sum_{j=1}^{N} A[j]$

Compute normalized entropy
$H \leftarrow -\frac{1}{\ln N} \sum_{i=1}^{N} p_i \ln(p_i + \epsilon)$

Compute top-κ concentration
$\mathcal{S}_\kappa \leftarrow \textsc{TopK}(A, \lfloor \kappa N \rfloor)$
$C \leftarrow \sum_{j \in \mathcal{S}_\kappa} A[j] / \sum_{i=1}^{N} A[i]$

Compute mid-zone coverage
$l \leftarrow \lfloor 0.25N \rfloor, \quad u \leftarrow \lfloor 0.75N \rfloor$
$M \leftarrow \sum_{i=l}^{u} A[i] / \sum_{j=1}^{N} A[j]$
return H, C, M

Algorithm 4 Task-Conditional Index Search

Require: $\mathcal{M}_{\text{VS}}$, block attention mass A, policy, profiles H, C, M, thresholds $\tau_{\text{ent}}, \tau_{\text{conc}}, \tau_{\text{mid}}, \tau_{\text{mod}}$, global budget K_g, base stride s_{base}, minimum stride $s_{\min}$

# No-Expand: keep the base topology
if policy == no_expand **then**
    $\mathcal{S} \leftarrow \mathcal{M}_{\text{VS}}$

Global-TopK: expand only when attention is diffuse and weakly focused
else if policy == global_topk **then**
    $\alpha_{\text{global}} \leftarrow \mathbb{I}(H > \tau_{\text{ent}}) \wedge \mathbb{I}(C < \tau_{\text{conc}})$
    if $\alpha_{\text{global}} == 1$ **then**
        $\mathcal{S}_g \leftarrow \textsc{TopKGlobal}(A, K_g)$
        $\mathcal{S} \leftarrow \mathcal{M}_{\text{VS}} \cup \mathcal{S}_g$
    else
        $\mathcal{S} \leftarrow \mathcal{M}_{\text{VS}}$
    end if

Coverage-Sampling: recover mid-zone coverage for retrieval-intensive inputs
else if policy == coverage_sampling **then**
    $\alpha_{\text{cov}} \leftarrow \mathbb{I}(M < \tau_{\text{mid}}) \wedge \mathbb{I}(C \leq \tau_{\text{mod}})$
    if $\alpha_{\text{cov}} == 1$ **then**
        $s \leftarrow \max(s_{\min}, \lfloor s_{\text{base}} \cdot M / \tau_{\text{mid}} \rfloor)$
        $\mathcal{S}_m \leftarrow \textsc{SampleInterval}(\text{Mid-Zone}, s)$
        $\mathcal{S} \leftarrow \mathcal{M}_{\text{VS}} \cup \mathcal{S}_m$
    else
        $\mathcal{S} \leftarrow \mathcal{M}_{\text{VS}}$
    end if
end if
return $\mathcal{S}$

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 elucidate our contributions 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: See Appendix H.

   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: Clearly. The proofs and empirical study of this paper are presented in Sections 3 and 4.

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: See supplemental materials. we provide the codes for experiments.

   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: [Yes]

Justification: See supplemental materials.

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: See section 4.1.

   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: See section 4.2 and 4.3.

   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: See section 4.1 and Appendix B.

   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: Yes, we reviewed the Code od Ethics.

   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: [N/A]

    Justification: There is no societal impact of the work performed.

    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: The paper presents a theoretical analysis and does not release any data or models that carry a risk of misuse.

    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: See section 4.1.

    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: See supplemental materials.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: [No]

Justification: LLM is used only for writing.

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.