

The Sparse Frontier: Sparse Attention Trade-offs in Transformer LLMs

Piotr Nawrot*
University of Edinburgh

Robert Li
Cohere

Renjie Huang
Cohere

Sebastian Ruder†
Meta

Kelly Marchisio
Cohere

Edoardo M. Ponti
University of Edinburgh

Abstract

Sparse attention offers a promising strategy to extend long-context capabilities in Transformer LLMs, yet its viability, its efficiency–accuracy trade-offs, and systematic scaling studies remain unexplored. To address this gap, we perform a careful comparison of *training-free* sparse attention methods at varying model scales, sequence lengths, and sparsity levels on a diverse collection of long-sequence tasks—including novel ones that rely on natural language while remaining controllable and easy to evaluate. Based on our experiments, we report a series of key findings: 1) an isoFLOPS analysis reveals that for very long sequences, larger and highly sparse models are preferable to smaller and dense ones. 2) The level of sparsity attainable while statistically guaranteeing accuracy preservation is higher during decoding than prefilling, and correlates with model size in the former. 3) There is no clear strategy that performs best across tasks and phases, with different units of sparsification or budget adaptivity needed for different scenarios. Even moderate sparsity levels often result in significant performance degradation on at least one task, highlighting that sparse attention is not a universal solution. 4) We introduce and validate novel scaling laws specifically tailored for sparse attention, providing evidence that our findings are likely to hold true beyond our range of experiments. Through these insights, we demonstrate that sparse attention is a key tool to enhance the capabilities of Transformer LLMs for processing longer sequences, but requires careful evaluation of trade-offs for performance-sensitive applications.

1 Introduction

The ability to model long sequences in large language models (LLMs) lies at the heart of long-context processing (Liu et al., 2025a) and inference-time scaling (Snell et al., 2024; Muennighoff et al., 2025). The fundamental bottleneck for this ability is the self-attention mechanism (Bahdanau et al., 2015) in Transformer (Vaswani et al., 2017) LLMs: during inference-time prefilling, the complexity of attention scales quadratically with sequence length—hence, ballooning time-to-first-token and deployment cost (Jiang et al., 2024). During inference-time decoding, dense attention results in a linearly growing cache of key–value (KV) items, whose size is proportional to the sequence length. Hence, runtime is dominated by high bandwidth memory access for loading these KV pairs from memory (Nawrot et al., 2024).

Sparse attention mechanisms aim to address the aforementioned challenges and reduce computational overhead by approximating dense attention outputs with a subset of key–query interactions (Fu, 2024).

*Research conducted during an internship at Cohere. Correspondence email: piotr.nawrot@ed.ac.uk

†Work done prior to joining Meta.

Though such methods promise to accelerate long sequence modelling and alleviate memory load while maintaining dense model performance (Jiang et al., 2024), their viability and robustness remain unclear due to a lack of comprehensive large-scale evaluation of state-of-the-art methods. While Li et al. (2025b), Liu et al. (2025b), and Yuan et al. (2024) provide a preliminary exploration of this question, they are limited to a narrow range of configurations (model sizes, sequence lengths, and sparsity levels), specific use cases (such as multi-turn decoding), and rely on datasets with variable sequence lengths that prevent a systematic analysis of length-dependent effects.

In this work, we conduct the largest-scale empirical analysis of *training-free*² sparse attention methods to date, covering models between 7B and 72B parameters, sequences between 16K to 128K tokens, and sparsity between 0% and 95%. To enable a controlled analysis, we first survey existing approaches, addressing the challenge of comparing rapidly evolving methods whose implementation details often obscure their core design principles. We distil these approaches into four key axes: units of sparsification (blocks/pages or verticals and slashes), importance estimation (fixed or context-aware), budget allocation across layers (uniform or adaptive), and KV cache management (eviction or full cache). Based on this taxonomy, we select six representative patterns spanning these design dimensions and harmonise their implementations, allowing us to rigorously evaluate the distinct effects of each fundamental principle.

For our evaluation, we curate a benchmark suite of 9 long-context tasks designed to systematically probe the influence of key factors on sparse attention performance. These factors include diverse *task types* (ranging from retrieval to multi-hop tracking and information aggregation), varying *naturalness* of sequences (synthetic or natural language), and precisely controlled *sequence lengths*. The importance of these dimensions is underscored by prior work indicating their significant impact on sparse attention effectiveness (Chen et al., 2024; Liu et al., 2024). Alongside established benchmarks (Rajpurkar et al., 2018; Pang et al., 2022; Tseng et al., 2016), we introduce novel, more challenging tasks based on natural language story templates. These complement synthetic benchmarks such as RULER (Hsieh et al., 2024), which measures core skills but whose artificial nature may fail to extrapolate performance on natural data. Our tasks address this gap by instantiating these core skills within a more realistic—and yet fully controllable—natural language setting. All this gives us the opportunity to address fundamental questions that currently remain unresolved:

RQ1: Given a fixed computing budget, should one choose a small dense model, or a large model with sparse attention? We conduct an isoFLOPS analysis finding that the answer depends on sequence length. For short sequences, any increase in either density or size enhances performance; for long sequences, only highly sparse configurations lie on the accuracy—FLOPS Pareto frontier.

RQ2: How far can we push sparsity while statistically guaranteeing that dense attention performance is fully preserved? To assess this, we develop a statistical testing framework reminiscent of distribution-free risk control (Angelopoulos et al., 2022). We find that in general, higher sparsity levels can be achieved during decoding and for larger model sizes. However, this aggregate perspective obscures a crucial caveat: even moderate sparsity levels frequently result in significant performance degradation on at least one task for most configurations.

RQ3: Are there one-size-fits-all methods that are consistently better across a diverse array of long-sequence tasks? We address this question separately for each of the inference phases (prefilling and decoding) and different categories of tasks. We find that no single sparse attention method uniformly excels across all diverse long-sequence tasks, with the ideal unit of sparsification and the choice of whether the budget should be adaptive being task- and phase-specific.

RQ4: Can we establish scaling laws (Brown et al., 2020; Kaplan et al., 2020; Hoffmann et al., 2022) for sparse attention, which can generalise beyond the range of configurations we consider? When holding out model sizes, sequence lengths, or sparsity levels, we can reliably predict their performance through a simple log-linear model, lending credence to the generality of our results.

Overall, our findings support the adoption of sparse attention for enhancing LLM capabilities for processing longer sequences, while emphasising that it requires careful evaluation of trade-offs, particularly for performance-sensitive applications. We release our code at <https://github.com/PiotrNawrot/sparse-frontier>.

²We deliberately concentrate on these approaches because training-based alternatives require either training from scratch (Yuan et al., 2025) or fine-tuning (Nawrot et al., 2024)—both prohibitively expensive in terms of computational resources and requiring access to often confidential training data mixtures.

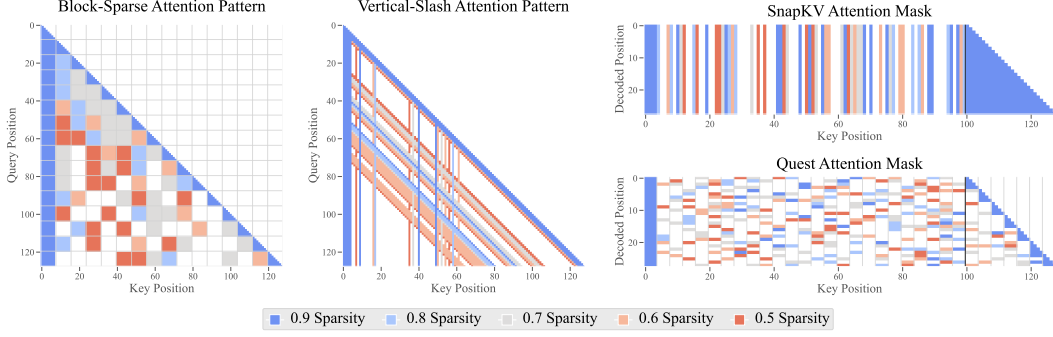


Figure 1: Overview of sparse attention methods for prefilling (left) and generation (right). These methods differ in the units of sparsification (blocks or pages vs. verticals and slashes), importance estimation, and KV cache management strategies. Colours represent query–key interactions preserved at different sparsity levels, while white areas indicate interactions that are not computed.

2 Training-Free Sparse Attention

Given an input sequence of tokens $X \in \mathbb{R}^{n \times d_{\text{model}}}$ with sequence length n and embedding dimension d_{model} , attention first projects the input into query, key, and value representations: $Q = XW^Q$, $K = XW^K$, $V = XW^V$, where $W^Q, W^K, W^V \in \mathbb{R}^{d_{\text{model}} \times d_{\text{head}}}$ are the projection matrices, and d_{head} is the attention head dimensionality. For readability, we omit the multiple attention heads.

Attention computes the output for the i -th token as a weighted sum over each value, with weights representing query–key (QK) interactions defined by the attention matrix $A \in \mathbb{R}^{n \times n}$:

$$A_i = \text{softmax} \left(\frac{Q_i K^\top}{\sqrt{d_{\text{head}}}} \right) \quad (1)$$

The output for each position is computed as $O_i = \sum_{j=1}^n A_{ij} V_j$, resulting in $O \in \mathbb{R}^{n \times d_{\text{head}}}$.

Text generation with Transformer-based models operates in two distinct phases, each with different computational characteristics affecting attention mechanisms:

Prefilling processes the entire input prompt at once, computing hidden representations for all tokens in parallel. This incurs quadratic computational complexity with respect to sequence length due to the computation of the full attention matrix A in Equation (1).

Decoding generates output tokens auto-regressively, one token at a time. Attention complexity is linear in sequence length per generation step because there is only a single query, but runtime is dominated by high bandwidth memory access for loading key–value (KV) pairs from memory.

By making A sparse, thus computing only a subset of query–key token interactions, sparse attention methods can mitigate these computational bottlenecks: reducing computational overhead during prefilling and memory transfer requirements during decoding.

The effectiveness of sparse attention methods in computation reduction and potentially memory savings is quantified using two key metrics: **sparsity**, defined as the ratio between the number of attention units (e.g., query–key interactions) that are *not computed* and the total number of units in dense attention for a given configuration, and **compression ratio**, defined as $\frac{1}{1 - \text{sparsity}}$. For example, sparsity = 0 corresponds to dense attention where all query–key interactions are computed, while sparsity = 0.9 means that 90% of interactions are skipped, resulting in a 10× compression ratio.

We condense the broad variety of existing training-free sparse attention approaches into four main dimensions: unit of sparsification, importance estimation, budget allocation, and KV cache management. We exclude token merging methods (Wang et al., 2024; Nawrot et al., 2024), as they leverage token similarity rather than sparsity. Visualisations of example patterns are shown in Figure 1.

2.1 Unit of Sparsification

Sparse attention methods differ primarily in the structural units of the attention matrix they prune or retain. Common units include *local windows* (contiguous regions around each query), *vertical columns* (tokens globally available to all queries), *slashes* (tokens at fixed offsets from each query), and *blocks* (fixed-size tiles of the attention matrix, such as 64×64 tokens). Larger structured units such as blocks or windows offer improved computational efficiency via better memory locality, whereas smaller units allow finer-grained, more precise selection of important information.

Block-based methods select blocks of units to approximate full attention. For prefilling, Star Attention approximates attention using local blocks and the first prefix block. MInference’s Block-Sparse pattern (Jiang et al., 2024) additionally incorporates a set of dynamically selected blocks for each chunk of query tokens. For decoding, Quest (Tang et al., 2024) and InfLLM (Xiao et al., 2024a) divide the KV cache into contiguous pages and select a subset of them for each decoded token.

Vertical-slash patterns represent another essential class of units. Early sparse attention methods like LM-Infinite (Han et al., 2024) and StreamingLLM (Xiao et al., 2024b) utilised local sliding windows supplemented by prefix tokens shared globally, also known as attention sinks. Extending this approach, Tri-shape (Li et al., 2025b) added full attention for suffix tokens, whereas SnapKV (Li et al., 2024b) introduced dynamically chosen vertical columns. MInference (Jiang et al., 2024) built on this by adding diagonal slashes at arbitrary offsets beyond the local window.³

2.2 Importance Estimation

To identify which specific units to retain, one can use fixed patterns—applied identically across all inputs—or dynamic patterns that adapt to the content being processed. Fixed patterns introduce no computational overhead but cannot adapt to varying input requirements, while dynamic patterns better preserve model quality but require additional computation to identify important connections.

Fixed patterns are identified with offline calibration to work well across all inputs. StreamingLLM (Xiao et al., 2024b), LM-Infinite (Han et al., 2024) and MoA (Fu et al., 2024) determine the number of initial tokens (attention sinks) and the width of a local sliding window.

Content-aware methods typically estimate the importance of QK units (tokens, blocks, or diagonals) to retain only the top-k most relevant ones, maximising attention score recall. They use lightweight heuristics such as approximated attention scores from highest-magnitude dimensions (SparQ, Quest; Ribar et al., 2024; Tang et al., 2024) or block-wise pooled token representations (Jiang et al., 2024). Some approaches subsample queries (SampleAttention; Zhu et al., 2024), recognising that recent query tokens often provide better indicators of KV unit importance, as in MInference’s Vertical-Slash pattern (Jiang et al., 2024) and SnapKV (Li et al., 2024b). During decoding, aggregated attention scores from a running average (H2O; Zhang et al., 2023) or the latest query (TOVA; Oren et al., 2024) guide the selection or eviction of KV units, again prioritising units likely to receive the highest attention weights. Some methods incorporate complementary heuristics alongside attention scores, such as vector norms of keys (Devoto et al., 2024) or values (VATP; Guo et al., 2024), recognising that a token’s contribution depends on both its attention score and value vector magnitude, as evident in Equation 1.

2.3 Budget Allocation

The third key dimension in sparse attention design concerns how to distribute the computational budget across different components of the model like layers and heads given a target sparsity level, which involves fundamental trade-offs between uniform simplicity and adaptive expressivity.

Uniform allocation assumes each head attends to the same number of tokens or blocks, as in Block-Sparse attention (Jiang et al., 2024) and SnapKV (Li et al., 2024b). While computationally simpler, this approach ignores the fact that not all layers and heads contribute equally to model accuracy, and their attention distributions often exhibit diverse sparsity characteristics (Zhang et al., 2024).

Adaptive methods such as PyramidKV (Cai et al., 2024) and PyramidInfer (Yang et al., 2024b) observe that attention scores’ entropy decreases with layer depth, suggesting that early layers require

³Interestingly, to efficiently compute attention along these diagonals, MInference uses 64×64 blocks aligned with these diagonals rather than computing attention for individual query-key pairs.

larger budgets than deeper ones. Similarly, Mixture of Sparse Attention (MoA; Fu et al., 2024) employs an automatic budget-fitting procedure using first-order Taylor approximations to optimally distribute the global attention budget across layers. Complementing this inter-layer approach, Ada-KV (Feng et al., 2024) focuses on flexible allocation within each layer by selecting the top- $(k \times h)$ tokens per layer (where h is the number of heads), redistributing the total cache size so that more critical heads retain additional keys while less critical ones are aggressively pruned.

Adaptive **threshold-based** allocation represents the most flexible approach, relaxing the constraint of a fixed global budget altogether. Methods like Twilight (Lin et al., 2025), FlexPrefill (Lai et al., 2025), Tactic (Zhu et al., 2025), and SampleAttention (Zhu et al., 2024) establish coverage thresholds (e.g., capturing 95% of the attention weight mass) and allow each head to dynamically select as many units as necessary to reach these thresholds. Consequently, heads exhibiting diffuse (high-entropy) attention naturally consume a larger share of the budget, whereas heads with sharply peaked attention consume fewer tokens. These methods also incorporate a minimum budget parameter to make them robust to edge cases of extreme sparsity (Chen et al., 2024).

2.4 KV Cache Management

The final dimension distinguishing sparse attention methods concerns their approach to KV cache management, particularly critical during the memory-bound decoding phase. While prefilling benefits primarily from computational speed-ups, decoding performance is often limited by the memory required to store and access the KV cache of past tokens. Sparse attention methods for decoding address this bottleneck through two main strategies, creating a fundamental trade-off between memory footprint reduction and information fidelity.

One strategy involves **KV cache eviction**, where methods like H2O (Zhang et al., 2023) or SnapKV (Li et al., 2024b) permanently discard selected tokens or blocks from the cache based on estimated importance. This directly reduces the memory footprint and data transfer costs, enabling longer sequences within fixed memory budgets and potentially improving hardware utilisation. However, this approach sacrifices information fidelity, as discarded tokens cannot be recovered if they become relevant later in the generation process. Robustly identifying tokens for eviction is challenging, as the set of important tokens can vary significantly across generation steps, particularly with shifting contexts (Li et al., 2025b).

Alternatively, other methods maintain the **full KV cache** but optimise computation by selectively loading only necessary KV pairs from memory during attention calculation, exemplified by Quest (Tang et al., 2024) and SparQ (Ribar et al., 2024). While they may incur a small memory overhead for auxiliary data structures used for importance estimation (e.g., page summaries), they avoid the information loss inherent in eviction. This often allows for effectively operating at higher sparsity levels compared to eviction-based methods (Li et al., 2025b), although they do not reduce the peak memory requirement for storing the cache itself.

3 Experimental Setup

3.1 Models

We perform experiments on Qwen 2.5 models (Yang et al., 2024a) (7B, 14B, 32B, 72B parameters). This model family was selected to systematically investigate how model size interacts with sequence length, task characteristics, and sparsity patterns. Qwen 2.5 uniquely supports 128k context length and provides multiple model sizes trained with consistent methodology—critical for controlled scaling experiments. We modify only the attention mechanism, preserving the original architectures, and utilise the vLLM inference engine (Kwon et al., 2023) with full bf16 precision. Implementation details for each sparse attention pattern are in Appendix A.1.

3.2 Sparse Attention Methods

We evaluate a series of sparse attention methods, which we choose as a representative set spanning across the key dimensions described in Section 2. We focus exclusively on content-aware methods, as prior work has demonstrated that fixed patterns consistently underperform their content-aware counterparts (Li et al., 2025b): the full list is presented in Table 1.

	Method	Unit	Budget	KV Cache Management
Prefill	Vertical-Slash (Jiang et al., 2024)	verticals and slashes	uniform	N/A
	FlexPrefill (Lai et al., 2025)	verticals and slashes	threshold-based	N/A
	Block-Sparse (Jiang et al., 2024)	blocks	uniform	N/A
Decode	SnapKV (Li et al., 2024b)	tokens	uniform	Eviction
	Ada-SnapKV (Feng et al., 2024)	tokens	adaptive	Eviction
	Quest (Tang et al., 2024)	pages	uniform	Full Cache

Table 1: Full list of content-aware sparse attention methods benchmarked in our experiments. These represent diverse strategies in terms of units, budget allocation, and KV cache management.

3.3 Tasks

We evaluate 9 diverse tasks selected to reflect different characteristics along 3 key dimensions known to influence sparse attention performance: task difficulty—defined by *Dispersion* (how hard it is to locate necessary information) and *Scope* (how much information must be processed) (Goldman et al., 2024)—and data *Naturalness* (natural language vs. synthetic data). This multi-dimensional approach is motivated by recent findings that attention patterns vary significantly across task types: retrieval tasks often exhibit localised attention, while reasoning tasks show more uniform distributions that are challenging for sparse methods (Liu et al., 2025c; Chen et al., 2024; Li et al., 2025b). The naturalness dimension is also crucial, as synthetic tasks with arbitrary symbol sequences yield different token representation distributions compared to natural language (Liu et al., 2024). Our task suite therefore incorporates four core tasks from the **RULER** benchmark (Hsieh et al., 2024)—Retrieval (NIAH), Multi-hop reasoning (VT), Aggregation (CWE), and QA (SQuAD)—to provide controlled environments (mostly synthetic) for specific capabilities. We complement these with natural text tasks from existing benchmarks assessed for minimal contamination risk (Li et al., 2024a), such as **QA** from QuALITY and TOEFL; however these are low-dispersion, low-scope tasks. Thus, we additionally introduce three novel tasks (**Story** Retrieval, Multi-hop, Filtering) that translate RULER’s challenging tasks (with high dispersion or scope) into naturalistic narratives, more representative of real-world use. We deliberately avoid open-ended tasks like summarisation due to unreliable evaluation metrics (Yen et al., 2024; Ye et al., 2024), focusing instead on structured-output tasks requiring factual answers, enabling precise evaluation via Exact Match Accuracy, Intersection-over-Union (IoU), and F1 score (all ranging from 0 to 1). These tasks are summarised in Table 2, with detailed descriptions in Appendix A.2 and examples in Appendix G.

Task Name	Description	Dispersion	Scope	Natural
QA (SQuAD)	Open-ended QA on a specified document among distractors	Low	Low	✓
QA (QuALITY, TOEFL)	Multiple-choice QA on a specified document among distractors	Low	Low	✓
Ruler NIAH	Extract 4 values for specified keys among many distractor key-value pairs	Low	Low	×
Ruler VT	Identify variables that resolve to a specific value via chained assignments	High	Low	×
Ruler CWE	Identify the 10 most frequent words from a list with distractors	Low	High	×
Story Retrieval	Answer 16 factoid-style questions about specific chapters in a long narrative	Low	Low	✓
Story Multi-hop	Identify the item acquired immediately before a target item across chapters	High	Low	✓
Story Filtering	Identify chapters where no item purchases occurred in a long narrative	Low	High	✓

Table 2: Summary of 9 evaluation tasks: QA tasks are based on existing datasets—SQuAD (Rajpurkar et al., 2018), QuALITY (Pang et al., 2022), TOEFL (Tseng et al., 2016)—while NIAH, VT, and CWE are taken from the RULER benchmark Hsieh et al. (2024). The remaining three (Story Retrieval, Multi-hop, and Filtering) are our contribution: we automatically generate multi-chapter narratives to evaluate the same skills as RULER tasks but expressed in naturalistic text. For each task, we indicate whether it has High or Low *dispersion* (information is difficult to locate), High or Low *scope* (large amount of necessary information), and whether it is based on *natural* text or is synthetic.

3.4 Evaluation Settings

Our evaluation covers sequence lengths of 16k, 32k, 64k, and 128k tokens, using 100 samples per configuration. We evaluate all combinations of task, model size, sequence length, and sparse attention pattern at fixed compression ratios spanning $1\times$ to $20\times$, interpolating performance at intermediate points. We ensure input samples are 95–100% of the target maximum token length, providing a consistent basis for evaluating the impact of sequence length on performance. Following Karpinska et al. (2024), we adopt a structured prompt format (see Appendix E), encouraging models to explicitly reason via a chain of thought before providing the final answer.

4 Results

4.1 IsoFLOPS Analysis

RQ1: Given a fixed computational budget, should one choose a small model with denser attention, or a large model with sparser attention?

Results in Figure 2 compare average performance across tasks against FLOPS for different model sizes and compression ratios at sequence lengths 32k and 128k tokens.⁴ We observe a crucial effect of sequence length on the accuracy–FLOPS trade-off.

With shorter contexts (32k tokens; Figures 2a left and 2b left), most model size–sparsity combinations lie on the Pareto frontier. In this setting, decreasing sparsity yields better performance more efficiently than increasing model size. This is explained by the fact that for shorter sequences, non-attention components (embedding, MLP, and output layers) dominate the computational cost (see Appendix D for an explanation). Despite testing compression ratios up to $20\times$, we observe that FLOPS ranges of each model size do not cross at this sequence length; the computational cost of a model twice as large (e.g., 14B vs. 7B) always exceeds the combined cost of MLP and dense attention in the smaller model, regardless of sparsity level.

With longer contexts (128k tokens), the computational dynamics change notably, with the impact of attention sparsity becoming more pronounced. Figures 2a right and 2b right illustrate that mostly only high-sparsity models now lie on the Pareto frontier, revealing an efficiency crossover where larger sparse models surpass smaller dense ones for the same computational budget. For prefilling, models with $5\text{--}15\times$ compression ratios remain optimal, while those with $20\times$ compression fall below the optimal boundary. Decoding shows better resilience to high sparsity, with even $20\times$ compression configurations being preferable to smaller models.

Looking more closely at model-specific sensitivity to sparsity, we observe distinct scaling behaviours across different model sizes. For decoding with the Quest pattern, performance resilience to sparsity correlates strongly with the model scale. Qwen 7B exhibits dramatic performance degradation under high compression, dropping by 40% (0.4 to 0.24) at 32k tokens and 79% (0.29 to 0.06) at 128k tokens when comparing dense to $20\times$ compressed variants. In contrast, larger models (32B, 72B) maintain performance stability with compression, showing gaps consistently below 0.05 even at $20\times$ compression. Prefilling exhibits markedly different behaviour, with performance gaps between dense and highly sparse models remaining relatively uniform (0.1–0.15) across all model sizes.

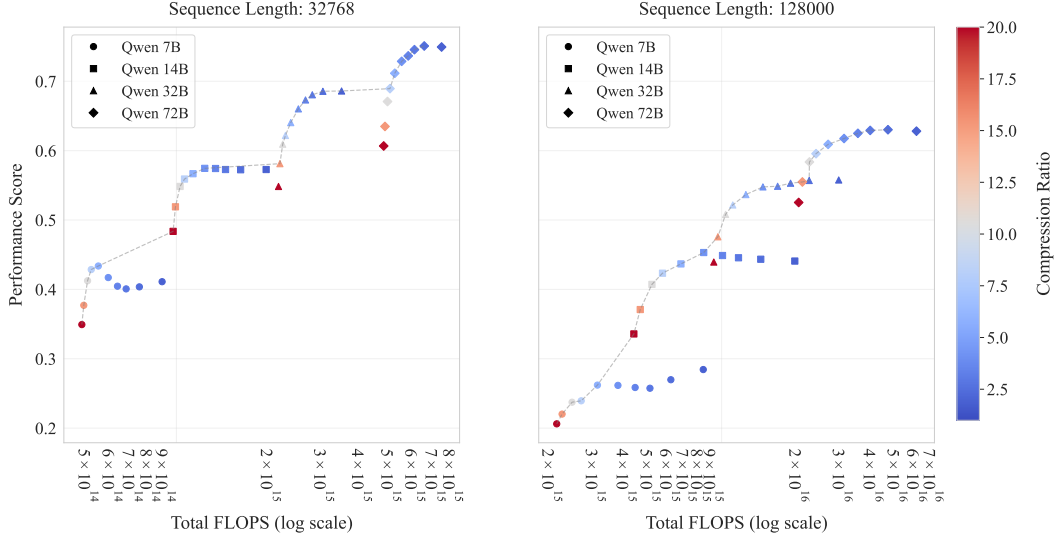
4.2 Maximum Sparsity with Guaranteed Performance

RQ2: What is the highest achievable sparsity that guarantees performance is preserved?

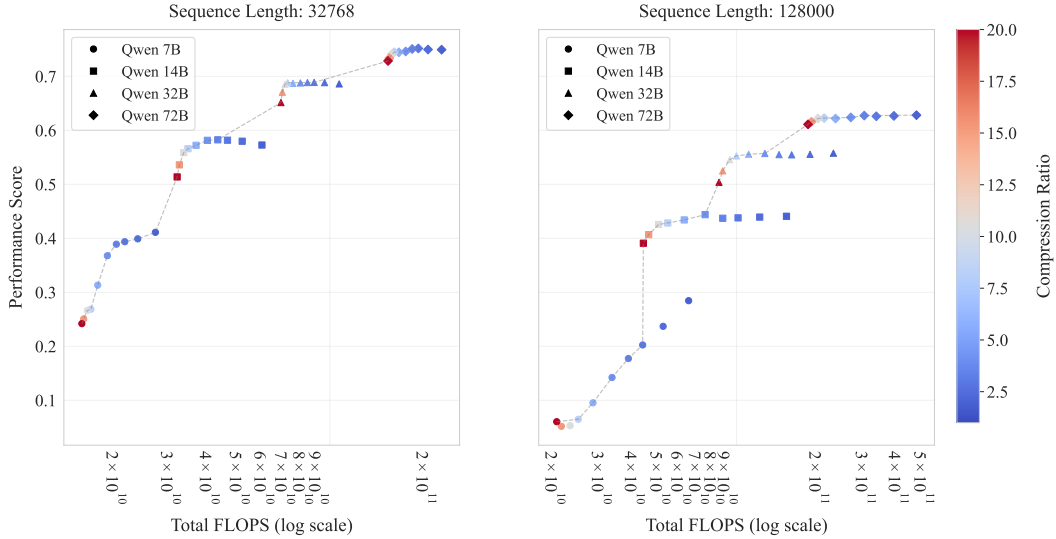
To find the maximum sparsity admissible per task, we perform a one-tailed Welch’s t-test between the performance of a dense model and its counterpart for each sparsity level, across each combination of model size and sequence length. *Maximum sparsity* is the highest sparsity level where the test is *not* significant with $p < 0.05$.⁵

⁴We approach this question using the Vertical-Slash pattern for prefilling and Quest pattern for decoding, as these are, on average, the best-performing patterns for their respective inference phases (see Section 4.3).

⁵We choose one-tailed Welch’s t because we do not assume identical variance between the two populations and are only interested in whether the performance significantly decreases. We excluded model–sparsity–task combinations where performance was random, i.e., lower than a threshold 0.05.



(a) IsoFLOPS analysis for prefilling, using Vertical-Slash pattern.



(b) IsoFLOPS analysis for decoding, using Quest pattern.

Figure 2: Performance comparison for batch size 1 across FLOPS, which are a function of sequence length, model size and sparsity level. We report 4 model sizes (markers) and compression ratios up to $20\times$ (heatmap). Performance scores are aggregated across all 9 tasks. In the plots, we display two sequence lengths—32k (**left**) and 128k (**right**)—and two phases—prefilling (**top**) and decoding (**bottom**). Crucially, there is a phase transition where after a critical sequence length (32–64k tokens for Qwen family models), highly sparse and large models surpass dense and small models in performance for the same FLOPS budget. See Appendix D for details on how we estimate the FLOPS, including indexing costs for sparse attention methods.

Figure 3 reveals distinct patterns for prefilling and decoding phases. For prefilling with Vertical-Slash, we observe no clear correlation between model size, sequence length, and maximum tolerable compression. Across all sequence lengths, models can achieve compression greater than $10\times$ without significant performance degradation when averaged across tasks. The high standard deviation (approximately 7) indicates substantial task-dependent variability in compression tolerance.

For decoding with Quest, model size emerges as a critical factor. In fact, the 7B model exhibits high variability and a clear negative correlation between sequence length and maximum compression—starting at $12\times$ for 16k tokens but declining to $5\times$ at 128k tokens. In contrast, larger models (32B

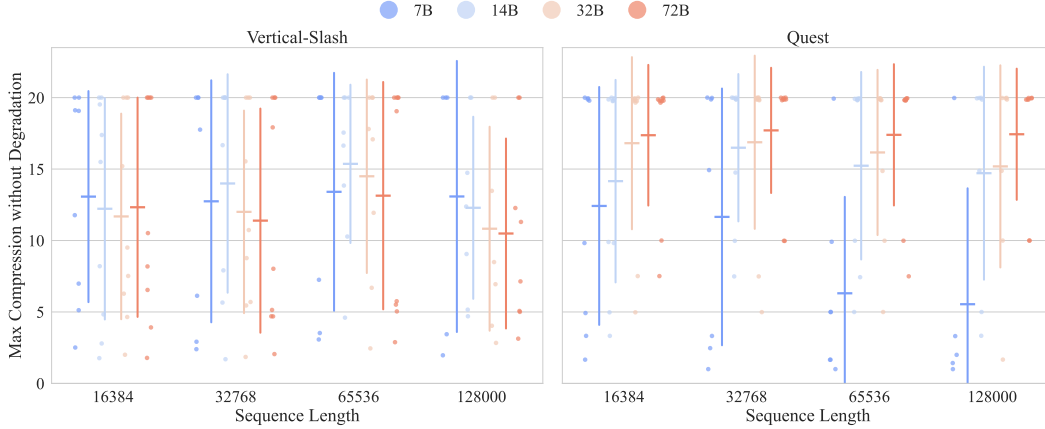


Figure 3: Maximum compression ratio with statistically significant performance retention (y-axis) across different model sizes (colours) and sequence lengths (x-axis). Each point represents a task, with horizontal bars showing the average maximum compression across tasks and vertical bars indicating standard deviation. **Left:** Vertical-Slash pattern for prefilling. **Right:** Quest pattern for decoding. The key conclusion is that decoding tolerates higher compression than prefilling on average, with larger models maintaining performance even at very high compression ratios. However, almost every configuration has at least one task where maximum tolerable compression is below $5\times$ (72B Quest being the only exception).

and 72B) maintain consistently high compression ratios (around $17\times$) regardless of sequence length. For these larger models, the majority of tasks can tolerate the maximum tested compression ratio of $20\times$ without significant degradation.

A crucial observation is that despite these promising averages, almost every configuration has at least one task where the maximum tolerable compression is lower than $5\times$, with 72B using Quest being the only exception (minimum $10\times$). This highlights an important limitation: while sparse attention methods appear highly effective on average, specific inputs in each configuration may still be disproportionately affected by sparsification.

4.3 Results in Individual Tasks and Methods

RQ3: Are there one-size-fits-all methods consistently superior across diverse long-sequence tasks?

In Figure 4, we examine at a finer granularity how task-specific characteristics interact with various sparsification patterns across prefilling and decoding. We observe distinct trends depending on the task’s characteristics along two axes defined in our experimental setup: Scope (how much context needs to be accessed) and Dispersion (how spread out the relevant information is). We first discuss performance trends based on these task characteristics (Sections 4.3.1 and 4.3.2), then analyse the specific impact of non-uniform budget allocation strategies (Section 4.3.3), and finally summarise the findings to determine if any method is universally superior (Section 4.3.4).

4.3.1 Performance on Retrieval Tasks (Low Scope, Low Dispersion)

Retrieval tasks (QuALITY, SQuAD, TOEFL, Ruler NIAH, Story Retrieval) are characterised by Low Scope and Low Dispersion, typically requiring retrieval of specific facts from the context. For these, sparse attention methods show a clear pattern: the severity of degradation correlates directly with the number of queries.⁶ In fact, single-query QA datasets (QuALITY, SQuAD, TOEFL) maintain dense-level accuracy even at $20\times$ compression. Ruler NIAH (4 queries) shows a slight decline for pre-filling methods and a moderate decline for decoding methods, while Story Retrieval (16 queries) experiences more substantial degradation for all methods except Quest.

⁶While more queries increase the required context, Retrieval tasks remain Low Scope compared to Aggregation tasks that necessitate processing the entire input.

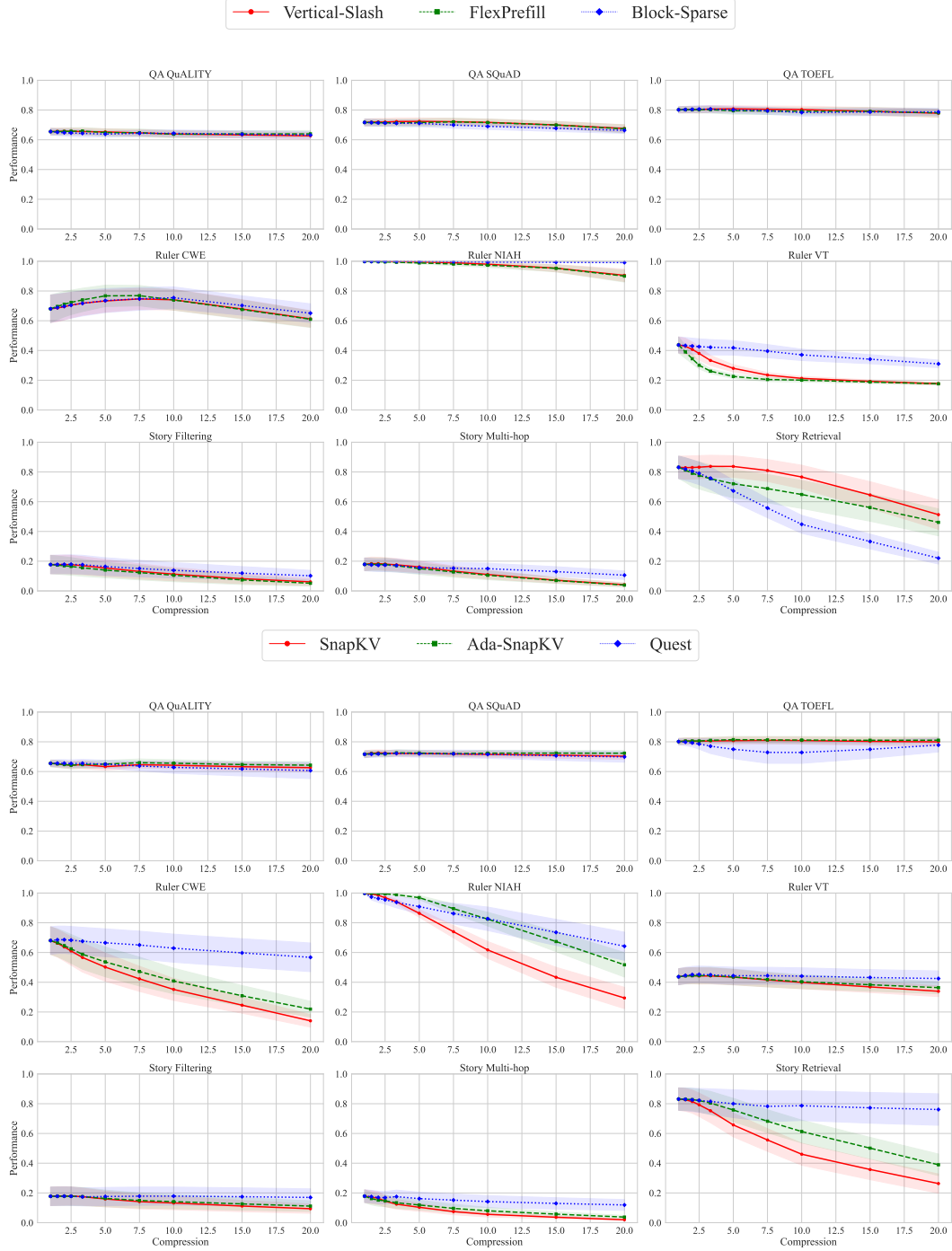


Figure 4: Performance comparison of different sparse attention methods across 9 tasks, aggregated over sequence lengths and models (shaded areas indicate the standard error). **Top:** prefilling methods (Vertical-Slash, FlexPrefill, Block-Sparse). **Bottom:** decoding methods (SnapKV, Ada-SnapKV, Quest). Each subplot shows the relationship between performance and compression for a specific task. The trade-off appears extremely task-dependent. Overall, Vertical-Slash performs best among prefilling methods, while Quest performs best among decoding methods.

Comparing units of sparsification, for methods that globally select individual tokens (Vertical-Slash, FlexPrefill, SnapKV, and Ada-SnapKV), performance drops linearly with compression ratio in the Story Retrieval task with 16 queries—a phenomenon previously studied by Yang et al. (2024c) and Chen et al. (2024). These methods estimate a key token’s importance by averaging attention scores across a subset of query tokens; however, this selection becomes harder with more queries and with fewer retrievable keys.

For block-level units of sparsification, we observe contrasting behaviour between prefilling and decoding. During prefilling, Block-Sparse performs poorly on multi-query retrieval tasks due to two key limitations (which are however necessary for efficiency reasons). First, it must process chunks of query tokens together (potentially containing multiple questions), which can lead to suboptimal importance estimation when queries are fragmented across blocks or multiple queries are collated within a single block. Second, Block-Sparse selects contiguous blocks of key tokens rather than individual tokens, introducing redundancy when retrieving factoid information that may only require specific, non-contiguous tokens.

In contrast, the chunk-level Quest pattern is the only method that maintains close-to-dense performance during decoding for Story Retrieval, even at $20\times$ compression. Unlike Block-Sparse, Quest selects chunks of key tokens for each individual query token, offering significantly higher flexibility. Nevertheless, despite its excellent performance on Story Retrieval, Quest’s performance on Ruler NIAH drops linearly with compression, performing on par or worse with Ada-SnapKV. This degradation can be attributed to the synthetic nature of the Ruler NIAH task, which consists primarily of random symbol sequences lacking underlying syntax or semantics—leading to different distributions of key representations compared with natural language (Liu et al., 2024). This fundamental difference affects the ability of both Ada-SnapKV and Quest to differentiate between unrelated token sets, with Quest being further disadvantaged due to its coarser granularity.

4.3.2 Performance on Tasks with High Scope or High Dispersion

For tasks characterised by High Scope or High Dispersion (Ruler CWE and VT, Story Filtering and Multi-hop), which require accessing broader context or integrating information spread across the sequence, chunk-level methods (Block-Sparse, Quest) consistently match or outperform token-level global-selection methods. Specifically, Block-Sparse uniquely maintains near-baseline performance on Ruler VT at moderate compression ($5\text{--}10\times$), whereas other methods degrade significantly even at modest compression ($3\times$). Similarly, Quest notably retains high performance on Ruler CWE during decoding. We believe that these results stem from fundamental differences in how information must be processed: while attention distributions for retrieval tasks are concentrated on local windows and universally important tokens, those for tasks requiring aggregation or reasoning are more uniform (Liu et al., 2025c; Chen et al., 2024), and hence require more flexible selection mechanisms.

During prefilling, the superior performance of Block-Sparse on tasks like Ruler VT or Story Filtering can be explained by the task structures: resolving independent chains of variables or examining multiple chapters independently is better suited for chunk-based methods compared with global token selection methods, which struggle to assign their limited budget in these tasks. For decoding, Quest outperforms on complex tasks by preserving the complete key-value cache while only optimising memory transfers, unlike SnapKV variants which irreversibly prune tokens before generation begins. Crucially, this severely limits the model’s ability to perform inference-time scaling through strategies like chain-of-thought reasoning as these rely on the ability to explore alternative reasoning paths—and hence possibly different sets of important tokens (Li et al., 2025b, 2024b)—when initial approaches prove incorrect (DeepSeek-AI, 2025).

4.3.3 Impact of Non-Uniform Budget Allocation

Non-uniform budget allocation strategies yield distinct outcomes depending on the inference phase. During prefilling, FlexPrefill’s adaptive, threshold-based selection either matches or underperforms Vertical-Slash’s simpler, uniform allocation across tasks. Ablations (Appendix A.1) revealed FlexPrefill’s extreme sensitivity to the minimum-budget hyperparameter: low values can degrade performance, while high values effectively mimic Vertical-Slash’s fixed-budget scheme at higher compression ratios. This behaviour echoes the “attention sink” phenomenon (Chen et al., 2024), where threshold-based methods might capture a few high-attention tokens, thus being satisfied with their cumulative attention recall criteria, while missing critical information distributed across the long tail of the atten-

tion distribution. Conversely, during decoding, Ada-SnapKV consistently outperforms the uniform SnapKV, particularly for complex multi-query retrieval tasks that inherently require broader token coverage, demonstrating the benefit of adaptive allocation when generation depends on accessing diverse context elements.

4.3.4 Summary and Method Recommendations

Our comprehensive evaluation indicates no single sparse attention method uniformly excels across all diverse long-sequence tasks, although methods offering greater flexibility generally perform better. For prefilling, Vertical-Slash consistently delivers optimal results on retrieval tasks (Low Scope, Low Dispersion), yet slightly trails behind chunk-level methods like Block-Sparse on aggregation and multi-hop reasoning tasks (High Scope or Dispersion). For decoding, Quest emerges as the most robust approach due to its high flexibility in selecting chunks per query and maintaining access to the entire KV cache, particularly adept at handling complex tasks; however, it remains vulnerable to synthetic tasks like Ruler NIAH and incurs additional memory overhead for maintaining page-level summaries required for its efficient selection mechanism. Therefore, the optimal choice of sparse attention method is highly dependent on the specific task characteristics, the required flexibility, and the inference phase (prefilling vs. decoding).

4.4 Sparse Attention Scaling Laws

RQ4: Can we establish scaling laws for sparse attention, which can generalise to larger model sizes, sequence lengths, or compression ratios?

We aim to fit scaling laws that predict downstream accuracy given task, model size, sequence length, and sparse attention compression ratio. We focus on the best-performing sparse attention pattern for each generation phase, namely Vertical-Slash for sparse pre-filling and Quest for sparse decoding.

After surveying previous scaling laws studies in foundation models, an overview of which is available in Appendix B, we opt for a log-linear scaling law as it makes the simple assumption of a power law with task-specific intercepts. Thus, our scaling law takes the form of

$$s = \alpha \log N + \beta \log L + \gamma \log C + \delta_{task} + \epsilon,$$

where s is the logit (inverse sigmoid) of the accuracy, N is model parameter count, L is sequence length, C is compression ratio, δ_{task} is a task-specific intercept and ϵ is a shared intercept. We provide more details on our scaling law formulation and fitting process in Appendix C.

We perform the fitting and validation of our scaling law on the Qwen 2.5 model family. We conduct 3 runs, each one holding out data points with the highest value for one axis as the test set, in order to evaluate the scaling law’s ability to extrapolate along that specific axis. Concretely, we hold out data points with the largest model (72B), the longest sequence (128k) and the highest compression ratio (20 \times) respectively in each run.

In terms of fitting, we choose R^2 as the main metric, as it measures how much of the total variance is explained by our scaling law model. We report R^2 and the inferred parameters in Table 3. From Table 3 it emerges that R^2 is very high (between 0.57 and 0.74). This demonstrates that the fit is very robust and our choice of predictors is largely appropriate for capturing the underlying pattern in the data. What is more, the parameters reveal that task-specific intercepts mirror our task categorisation: their value is negative for most multi-hop and aggregation tasks (Ruler VT and Story Filtering and Multi-Hop). The intercept for Ruler NIAH stands out for its exceptionally high value.

R ²			Parameters											
			ϵ	δ _{QA}		δ _{Ruler}			δ _{Story}	Retr.	α	β	γ	
			SQuAD	TOEFL	CWE	NIAH	VT	Filt.	Mult.					
Prefill	Sequence Length	0.74	-16.92	0.27	0.80	0.76	14.67	-1.60	-3.55	-3.21	1.43	1.31	-1.15	-1.00
	Model Size	0.64	-18.10	0.20	0.73	-0.42	10.51	-1.63	-4.42	-3.74	0.46	1.58	-1.58	-0.83
	Compression Ratio	0.72	-18.08	0.24	0.79	0.40	13.62	-1.54	-3.65	-3.55	1.17	1.51	-1.51	-0.73
Decode	Sequence Length	0.64	-28.82	0.35	0.72	0.54	12.18	-0.85	-2.98	-2.87	2.32	1.72	-0.97	-0.94
	Model Size	0.57	-35.57	0.32	0.60	-0.53	8.45	-0.89	-3.88	-3.44	1.31	2.25	-1.45	-0.81
	Compression Ratio	0.64	-26.89	0.31	0.66	0.26	11.37	-0.90	-3.19	-3.40	1.78	1.81	-1.38	-0.82

Table 3: R^2 as a measure of fitness and inferred parameters for our scaling laws.

To evaluate the quality of the scaling laws’ predictions, for each of the three axes of extrapolation and for each of the nine tasks, we report two main metrics on test examples: 1) the Spearman’s ρ between true and predicted values; and 2) the mean absolute error (MAE) of the predicted accuracy. The evaluation results of our scaling laws are shown in Table 4 in terms of Spearman’s ρ and in Table 7 in Appendix C.3 in terms of MAE.

		Quality	QA SQuAD	TOEFL	CWE	Ruler NIAH	VT	Filtering	Story Multi-hop	Retrieval
Prefill	Sequence Length	.73 *	.79 *	.78 *	.86 *	.87 *	.86 *	.90 *	.34	.82 *
	Model Size	.68 *	.75 *	.60 *	.85 *	.56 *	.68 *	.89 *	.90 *	.85 *
	Compression Ratio	.77 *	.87 *	.63 *	.75 *	.82 *	.40	.79 *	.38	.65 *
Decode	Sequence Length	.75 *	.81 *	.69 *	.93 *	.84 *	.54 *	.67 *	.50 *	.92 *
	Model Size	.77 *	.49 *	.31 *	.92 *	.52 *	.84 *	.84 *	.89 *	.81 *
	Compression Ratio	.88 *	.91 *	.87 *	.97 *	.75 *	.92 *	.96 *	.59	.93 *

Table 4: Spearman’s correlation ρ by task between true and predicted accuracy on each held-out axis (sequence length, model size, compression ratio) and each phase (prefilling and decoding). We mark statistically significant entries where $p < 0.05$ with *.

Based on Table 4, we find that our scaling laws can meaningfully predict downstream accuracy in new configurations, as correlations between true and predicted performance are generally strong. Nonetheless, there remain a few settings where the correlation is weak or not statistically significant. Most notably, this is the case for the task of Story Multi-hop, where our laws fail to extrapolate accuracy when ablating the compression ratio (during both phases) and sequence length (during prefilling).

5 Conclusions

Our study undertakes the most comprehensive comparison of sparse attention methods to date, applied to a range of model scales (up to 72B), sequence lengths (up to 128K), and sparsity levels (up to 95%) across 9 diverse long-sequence tasks. We revealed that, under an isoFLOPS analysis, larger models with high sparsity are more effective than their smaller, dense counterparts, particularly for very long sequences. This finding suggests a shift in strategy where scaling up model size combined with sparse attention mechanisms is crucial for efficiently handling extended sequences.

Another key finding is that the degree of sparsity applicable while statistically ensuring full accuracy retention is on average very high ($10\text{-}15\times$) and even increases for larger models during decoding. However, this aggregate perspective obscures a crucial caveat: our analysis reveals that even moderate sparsity levels (e.g., 5x compression) frequently result in significant performance degradation on at least one task for most configurations. This highlights a critical limitation: average performance gains may obscure significant degradation on certain tasks, even at moderate sparsity levels. This task-dependent sensitivity underscores the need for thorough evaluation across diverse benchmarks, covering the full spectrum of potential deployment scenarios. Consequently, sparse attention is not a panacea and always necessitates careful trade-off evaluation, particularly for performance-sensitive applications. Future research should prioritise dynamic sparsity mechanisms that adapt to input and task demands, ideally incorporating performance guarantees.

Zooming in on the results, we observe a trend towards increased performance with greater flexibility and finer granularity in selecting attention interactions for both prefilling and decoding. During prefilling, the optimal sparsification structure (e.g., blocks or verticals and slashes) varies by task, with uniform allocation across layers performing comparably to dynamic allocation. During decoding, page-level Quest excels by preserving the KV cache structure, avoiding the performance degradation associated with token pruning during generation. While more flexible and fine-grained methods incur higher indexing costs, their ability to adapt to diverse task requirements (as evidenced by the differing optimal strategies for prefilling vs. decoding and across tasks) suggests that pursuing such adaptive sparsity mechanisms is a promising direction for future research.

Finally, our study establishes robust scaling laws for sparse attention, which hold true on held-out data, indicating that the observed trends are likely to generalise beyond the tested configurations. Together,

these insights demonstrate that sparse attention is bound to play a key role in next-generation LLM architectures, ultimately contributing to more scalable, efficient, and adaptable AI models.

Acknowledgements

This work was supported in part by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) and the University of Edinburgh, School of Informatics and School of Philosophy, Psychology & Language Sciences.

References

- Samira Abnar, Harshay Shah, Dan Busbridge, Alaaeldin Mohamed Elnouby Ali, Josh Susskind, and Vimal Thilak. 2025. Parameters vs FLOPS: Scaling laws for optimal sparsity for mixture-of-experts language models. *arXiv:2501.12370*.
- Anastasios N. Angelopoulos, Stephen Bates, Emmanuel J. Candès, Michael I. Jordan, and Lihua Lei. 2022. Learn then test: Calibrating predictive algorithms to achieve risk control. *arXiv:2110.01052*.
- Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in Neural Information Processing Systems*.
- Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and Wen Xiao. 2024. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv:2406.02069*.
- Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. 2024. Magicpig: LSH sampling for efficient LLM generation. *arXiv:2410.16179*.
- Leshem Choshen, Yang Zhang, and Jacob Andreas. 2024. A Hitchhiker’s guide to scaling law estimation. *arXiv:2410.11840*.
- DeepSeek-AI. 2025. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv:2501.12948*.
- Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. 2024. A simple and effective 12 norm-based strategy for kv cache compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18476–18499.
- Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2024. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient LLM inference. *arXiv:2407.11550*.
- Elias Frantar, Carlos Riquelme, Neil Houlsby, Dan Alistarh, and Utku Evci. 2023. Scaling laws for sparsely-connected foundation models. *arXiv:2309.08520*.
- Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. 2024. Moa: Mixture of sparse attention for automatic large language model compression. *arXiv:2406.14909*.
- Yao Fu. 2024. Challenges in deploying long-context transformers: A theoretical peak performance analysis. *arXiv:2405.08944*.
- Omer Goldman, Alon Jacovi, Aviv Slobodkin, Aviya Maimon, Ido Dagan, and Reut Tsarfaty. 2024. Is it really long context if all you need is retrieval? Towards genuinely difficult long context NLP. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16576–16586.

- Zhiyu Guo, Hidetaka Kamigaito, and Taro Watanabe. 2024. Attention score is not all you need for token importance indicator in kv cache reduction: Value also matters. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21158–21166.
- Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 3991–4008.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 30016–30030.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Krman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. Ruler: What’s the real context size of your long-context language models? *arXiv:2404.06654*.
- Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv:2001.08361*.
- Marzena Karpinska, Katherine Thai, Kyle Lo, Tanya Goyal, and Mohit Iyyer. 2024. One thousand and one pairs: A "novel" challenge for long-context language models. *arXiv:2406.16264*.
- Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.
- Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. 2025. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. In *The Thirteenth International Conference on Learning Representations*.
- Margaret Li, Sneha Kudugunta, and Luke Zettlemoyer. 2025a. (mis)fitting: A survey of scaling laws. In *Proceedings of ICLR 2025*.
- Xinze Li, Yixin Cao, Yubo Ma, and Aixin Sun. 2024a. Long context vs. RAG for LLMs: An evaluation and revisits. *arXiv:2501.01880*.
- Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H. Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and Lili Qiu. 2025b. SCBench: A kv cache-centric analysis of long-context methods. In *The Thirteenth International Conference on Learning Representations*.
- Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024b. Snapkv: LLM knows what you are looking for before generation. *arXiv:2404.14469*.
- Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. 2025. Twilight: Adaptive attention sparsity with hierarchical top- p pruning. *arXiv:2502.02770*.
- Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. 2025a. A comprehensive survey on long context language modeling. *arXiv:2503.17407*.

- Xiang Liu, Zhenheng Tang, Hong Chen, Peijie Dong, Zeyu Li, Xiuze Zhou, Bo Li, Xuming Hu, and Xiaowen Chu. 2025b. Can LLMs maintain fundamental abilities under kv cache compression? *arXiv:2502.01941*.
- Xiang Liu, Zhenheng Tang, Hong Chen, Peijie Dong, Zeyu Li, Xiuze Zhou, Bo Li, Xuming Hu, and Xiaowen Chu. 2025c. Can LLMs maintain fundamental abilities under kv cache compression? *arXiv:2502.01941*.
- Xiaoran Liu, Ruixiao Li, Qipeng Guo, Zhigeng Liu, Yuerong Song, Kai Lv, Hang Yan, Linlin Li, Qun Liu, and Xipeng Qiu. 2024. Reattention: Training-free infinite context with finite attention scope. *arXiv:2407.15176*.
- Yuqi Luo, Chenyang Song, Xu Han, Yingfa Chen, Chaojun Xiao, Zhiyuan Liu, and Maosong Sun. 2025. Sparsing law: Towards large language models with greater activation sparsity. *arXiv:2411.02335*.
- Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori Hashimoto. 2025. s1: Simple test-time scaling. *arXiv:2501.19393*.
- Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. 2024. Dynamic memory compression: Retrofitting LLMs for accelerated inference. In *Proceedings of the 41st International Conference on Machine Learning*.
- Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. 2024. Transformers are multi-state RNNs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18724–18741.
- Richard Yuanzhe Pang, Alicia Parrish, Nitish Joshi, Nikita Nangia, Jason Phang, Angelica Chen, Vishakh Padmakumar, Johnny Ma, Jana Thompson, He He, et al. 2022. 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*, pages 5336–5358.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don’t know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 784–789.
- Luka Ribar, Ivan Chelombiev, Luke Hudliss-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2024. Sparq attention: Bandwidth-efficient LLM inference. In *International Conference on Machine Learning*, pages 42558–42583. PMLR.
- Yangjun Ruan, Chris J. Maddison, and Tatsunori Hashimoto. 2024. Observational scaling laws and the predictability of language model performance. *arXiv:2405.10938*.
- Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv:2408.03314*.
- Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context LLM inference. In *International Conference on Machine Learning*, pages 47901–47911. PMLR.
- Bo-Hsiang Tseng, Sheng-Syun Shen, Hung-Yi Lee, and Lin-Shan Lee. 2016. Towards machine comprehension of spoken content: Initial toefl listening comprehension test by machine. *arXiv:1608.06378*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.
- Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang. 2024. Model tells you where to merge: Adaptive kv cache merging for LLMs on long-context tasks. *arXiv:2407.08454*.

- Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024a. InfLLM: Training-free long-context extrapolation for LLMs with an efficient context memory. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024b. Efficient streaming language models with attention sinks. *arXiv:2309.17453*.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2.5 technical report. *arXiv:2412.15115*.
- Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024b. Pyramidinfer: Pyramid kv cache compression for high-throughput LLM inference. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3258–3270.
- June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. 2024c. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv:2402.18096*.
- Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V Chawla, and Xiangliang Zhang. 2024. Justice or prejudice? Quantifying biases in LLM-as-a-judge. In *Neurips Safe Generative AI Workshop 2024*.
- Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. 2024. Helmet: How to evaluate long-context language models effectively and thoroughly. *arXiv:2410.02694*.
- Jiayi Yuan, Hongyi Liu, Shaochen Zhong, Yu-Neng Chuang, Songchen Li, Guanchu Wang, Duy Le, Hongye Jin, Vipin Chaudhary, Zhaozhuo Xu, Zirui Liu, and Xia Hu. 2024. Kv cache compression, but what must we give in return? A comprehensive benchmark of long context capable approaches. In *The 2024 Conference on Empirical Methods in Natural Language Processing*.
- Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Y. X. Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. 2025. Native sparse attention: Hardware-aligned and natively trainable sparse attention. *arXiv:2502.11089*.
- Zhenrui Yue, Honglei Zhuang, Aijun Bai, Kai Hui, Rolf Jagerman, Hansi Zeng, Zhen Qin, Dong Wang, Xuanhui Wang, and Michael Bendersky. 2025. Inference scaling for long-context retrieval augmented generation. *arXiv:2410.04343*.
- Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John C.S. Lui, and Haibo Chen. 2024. Unifying kv cache compression for large language models with leankv. *arXiv:2412.03131*.
- Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Proceedings of the 37th International Conference on Neural Information Processing Systems, NIPS '23*, Red Hook, NY, USA. Curran Associates Inc.
- Kan Zhu, Tian Tang, Qinyu Xu, Yile Gu, Zhichen Zeng, Rohan Kadekodi, Liangyu Zhao, Ang Li, Arvind Krishnamurthy, and Baris Kasikci. 2025. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context LLMs. *arXiv:2502.12216*.
- Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Huanqi Cao, Xiao Chuanfu, Xingcheng Zhang, Dahua Lin, and Chao Yang. 2024. Sampleattention: Near-lossless acceleration of long context LLM inference with adaptive structured sparse attention. *arXiv:2406.15486*.

A Experimental details

A.1 Implementation Details

This section provides supplementary details on the sparse attention patterns evaluated, focusing on hyperparameter tuning and specific configurations used to achieve target compression ratios. We tuned hyperparameters for each pattern using ablation studies on the Qwen-7B model with a 16K sequence length across all tasks, varying compression ratios from 1x to 10x. Our main experiments evaluated performance at fixed compression ratios (1.5x, 2.0x, 2.5x, 3.33x, 5x, 7.5x, 10x, 15x, 20x), using linear interpolation for intermediate values where necessary. Table 6 summarizes the final parameters we used for each pattern, sequence length, and compression ratio.

A.1.1 Block-Sparse Attention

We implement block-sparse attention by dividing the attention matrix into fixed-size blocks. Based on our ablation studies (Figure 5), we selected a block size of 16x16, as smaller blocks consistently yielded better performance. To achieve a target compression ratio, we select the top- k key blocks for each query block, where k is determined via binary search. We always preserve attention sinks (the first key block) and local context (diagonal key blocks corresponding to the query block).

A.1.2 Vertical-Slash Pattern

We implement the Vertical-Slash pattern (Jiang et al., 2024) by allocating a uniform budget to global (vertical columns) and local (slash diagonals) attention components. We select the most important verticals and slashes by approximating attention scores using a limited window of recent query tokens. Our ablation studies (Figure 11) revealed task-dependent optimal approximation window sizes: 512 tokens for retrieval-heavy tasks (Ruler NIAH, Story Retrieval) and 256 tokens for other tasks. This observation correlates with the typical query lengths for these tasks (see Table 5). We consistently preserve the first 4 (prefix) and the most recent 64 (local) tokens. To achieve target compression ratios, we compute the required number of verticals and slashes based on collected attention statistics for each sequence length.

Table 5: Statistics of token lengths of question and instruction for each task across 100 samples, informing the choice of approximation window size for Vertical-Slash and FlexPrefill.

Task	Mean Tokens	Min Tokens	Max Tokens
QA QuALITY	243.63	196	423
QA SQuAD	217.08	210	235
QA ToeflQA	237.67	202	270
RULER CWE	227.00	227	227
RULER NIAH	337.74	330	350
RULER VT	230.00	230	230
Story Filtering	184.00	184	184
Story Multi-hop	192.97	192	195
Story Retrieval	457.54	452	462

A.1.3 FlexPrefill

We implement FlexPrefill (Lai et al., 2025), which enhances Vertical-Slash by introducing dynamic budget allocation per layer and head, controlled by a coverage parameter α and a minimum budget (`min_budget`). We set $\tau = 0$ in our experiments, hence disabling Query-Aware attention. This choice stemmed from two key considerations: first, our preliminary tests indicated no significant performance gains from enabling it, aligning with the findings reported in the original work; second, this setting isolates the dynamic budget allocation mechanism, allowing us to specifically evaluate its impact compared to the fixed allocation used in the Vertical-Slash pattern. We employ the same task-dependent approximation windows (256 / 512 tokens) and critical token preservation strategy (first 4 prefix, most recent 64 local) as in our Vertical-Slash implementation. Our ablations (Figure 8) indicated that setting `min_budget` to 512 significantly improved performance, suggesting the importance of maintaining a minimum level of connectivity during prefilling. We achieved target compression ratios by selecting the appropriate α based on attention statistics while keeping

`min_budget` fixed at 512. For high compression ratios where dynamic allocation proved less effective, we set $\alpha = 0$, effectively reverting to a uniform allocation of Vertical-Slash.

A.1.4 SnapKV

We implement SnapKV (Li et al., 2024b) by compressing the Key-Value (KV) cache after the prefilling stage and applying a uniform token budget across all heads for the subsequent decoding phase. We predict token importance for decoding by computing attention scores using a window of recent query tokens (approximation window). Our ablations showed an optimal approximation window size of 256 tokens (Figure 9), with no significant task dependency observed, unlike Vertical-Slash and FlexPrefill. We smooth the calculated token importance scores using 1D average pooling with a kernel size of 21 (chosen based on Figure 10). We always preserve the first 4 and the most recent 128 tokens. We control sparsity by setting the ‘`token_capacity`’ (token limit per head) to match the target compression ratio.

A.1.5 Ada-SnapKV

We implement Ada-SnapKV (Feng et al., 2024), which extends SnapKV by incorporating dynamic token budget allocation per head. One difference in our implementation between Ada-SnapKV and SnapKV is that we use max-aggregation (instead of averaging) across query positions and heads for score calculation; this empirically proved more effective for adaptive allocation but had no effect for uniform (SnapKV) allocation. We utilize the same smoothing kernel size (21) and critical token preservation strategy (first 4 prefix, most recent 128 local) as in our SnapKV implementation. Our ablations (Figure 7) indicated that providing each head with a minimum budget of 20% of its capacity was optimal. Performance was found to be less sensitive to minimum budget (performing well within the 10-50% range) compared to FlexPrefill’s sensitivity, but degraded sharply when approaching 100% (uniform allocation), underscoring the benefits of dynamic allocation during decoding. We control sparsity by setting ‘`token_capacity`’, identically to SnapKV.

A.1.6 Quest

We implement Quest (Tang et al., 2024), which applies dynamic sparse attention during the decoding phase at the page level. Based on our ablations (Figure 6), we used a page size of 16 tokens. We represent pages by their minimum and maximum key values to enable efficient similarity computation with queries. At each decoding step, we select the most relevant pages based on query-page similarity scores, always including the page containing the current token. We control sparsity by setting the ‘`token_budget`’ (number of tokens selected per step) to achieve the target compression ratio.

Table 6: Pattern parameters for different sequence lengths and compression ratios

Pattern	Parameter	Sequence Length	Values for Different Compression Ratios
Vertical & Slash	Verticals/Slashes	16384	164, 240, 315, 400, 448, 576, 768, 1024, 1536, 2304
		32768	290, 384, 448, 576, 704, 1024, 1536, 2304, 3584, 4608
		65536	400, 448, 544, 640, 960, 1280, 2304, 4096, 6144, 8192
		128000	480, 768, 1024, 1536, 2048, 3584, 5632, 10240, 13312, 18432
FlexPrefill	$(\alpha, \text{min_budget})$	16384	(0, 164), (0, 240), (0, 315), (0, 400), (0.55, 512), (0.71, 512), (0.88, 512)
		32768	(0, 290), (0, 384), (0.45, 512), (0.6, 512), (0.7, 512), (0.8, 512), (0.92, 512)
		65536	(0, 400), (0.45, 512), (0.55, 512), (0.7, 512), (0.77, 512), (0.85, 512), (0.94, 512)
Block Sparse	top_chunks	16384	26, 35, 53, 71, 108, 188, 300
		32768	52, 69, 105, 141, 216, 376, 600
		65536	104, 139, 210, 283, 432, 752, 1200
SnapKV/AdaSnapKV	token_capacity	16384	819, 1092, 1638, 2183, 3276, 4915, 6553, 8192, 9830, 11468
		32768	1638, 2185, 3276, 4367, 6553, 9830, 13107, 16384, 19660, 22937
		65536	3276, 4371, 6553, 8735, 13107, 19660, 26214, 32768, 39321, 45875
		128000	6400, 8544, 12800, 17056, 25600, 38400, 51200, 64000, 76800, 89600
Quest	token_budget	16384	816, 1088, 1632, 2176, 3280, 4912, 6560, 8192, 9824, 11472
		32768	1632, 2192, 3280, 4368, 6560, 9824, 13104, 16384, 19664, 22944
		65536	3280, 4368, 6560, 8736, 13104, 19664, 26208, 32768, 39328, 45872
		128000	6400, 8544, 12800, 17056, 25600, 38400, 51200, 64000, 76800, 89600

A.2 Task Details

This section provides further details on the nine evaluation tasks used in our experiments. Tasks are grouped into Question Answering, synthetic tasks from RULER (Hsieh et al., 2024), and our Story

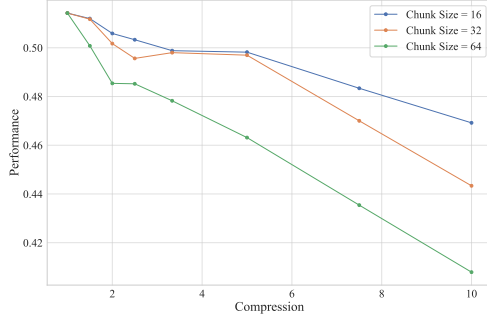


Figure 5: Block-Sparse block size.

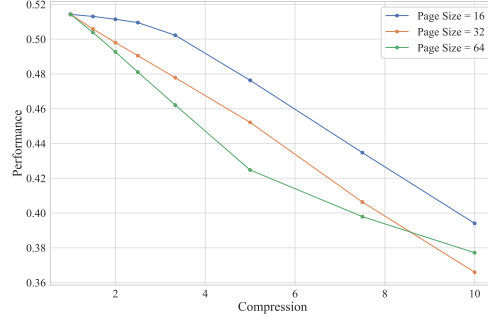


Figure 6: Quest page size.

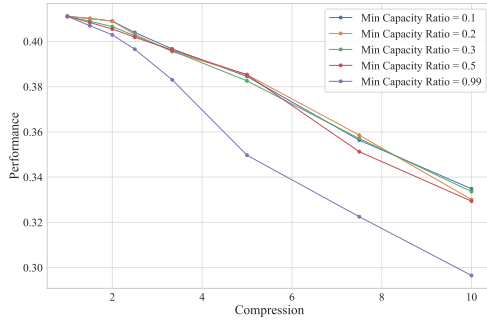


Figure 7: Ada-SnapKV min budget.

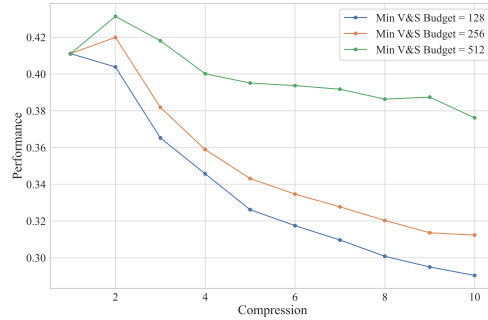


Figure 8: FlexPrefill min budget.

tasks. We specify key hyperparameters, evaluation metrics, and characterise each task along the axes of Scope (Low vs. High) and Dispersion (Low vs. High), as defined in Table 2. Scope refers to the amount of information required, while Dispersion indicates how difficult it is to locate the relevant information within the context.

A.2.1 Question Answering (QA)

We use SQuAD (Rajpurkar et al., 2018) from RULER and two other QA datasets selected for minimal data contamination (Li et al., 2024a): QuALITY (Pang et al., 2022) and ToeflQA (Tseng et al., 2016).

- **Setup:** Each example contains one answer-bearing document and distractor documents to reach the target sequence length. Documents are shuffled and numbered; the question refers to a specific document ID.
- **Preprocessing:** We remove duplicate question-context pairs and filter examples where the original context exceeds 8k tokens (ensuring space for at least one distractor at 16k sequence length).
- **Evaluation:** Exact Match Accuracy (QuALITY, ToeflQA multiple-choice), token-level F1 (SQuAD open-ended).
- **Characteristics:** Natural text. Requires identifying and processing a specific document, thus characterised by **Low Dispersion** and **Low Scope**. See Appendix G.1.

A.2.2 Synthetic – RULER Tasks

We use three synthetic tasks from the RULER benchmark (Hsieh et al., 2024).

- **Needle-in-a-Haystack (NIAH):** Extract values for 4 target keys from a document containing relevant and distractor key-value pairs (random hyphenated strings). Evaluated using Exact Match Accuracy. Requires finding specific items, characteristic of **Low Dispersion** and **Low Scope**. See Appendix G.2.

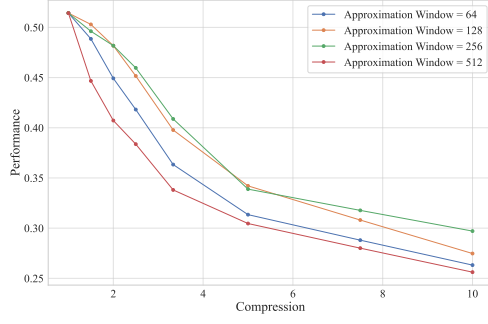


Figure 9: SnapKV/Ada-SnapKV approximation window.

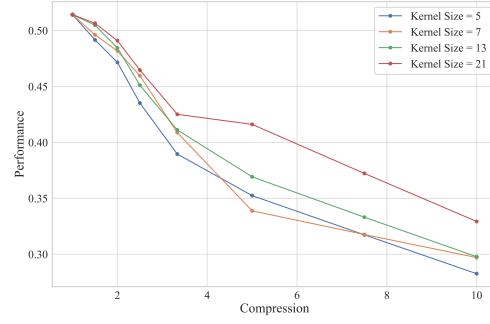


Figure 10: SnapKV/Ada-SnapKV kernel size.

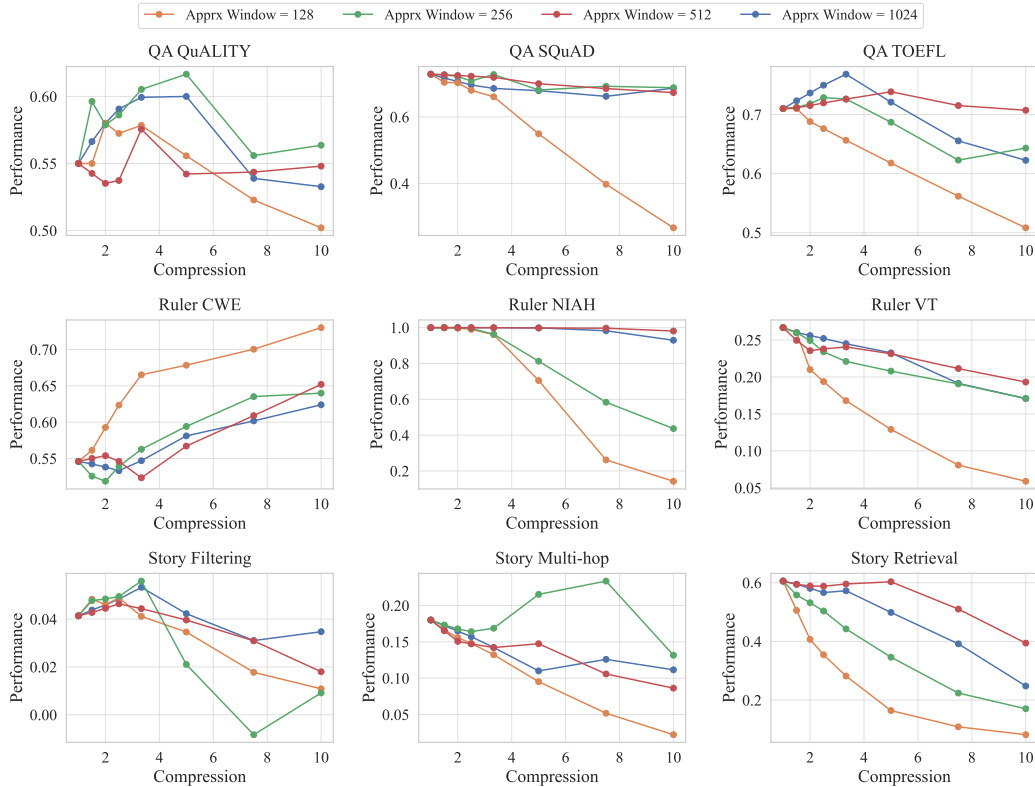


Figure 11: Vertical-Slash approximation window ablation per task.

- **Common Word Extraction (CWE):** Identify the 10 most frequent words (appearing 30 times each) among distractor words (appearing 3 times each), sampled from a vocabulary of $\sim 9,000$ English words⁷. Evaluated using Intersection-over-Union (IoU). Requires processing the entire context to count frequencies, demanding **Low Dispersion** (words are presented directly as a list, not obscured within complex structures) but **High Scope** (all words must be processed to determine frequencies). See Appendix G.3.
- **Variable Tracking (VT):** Resolve variable assignments (direct or chained) to identify all variables matching a target value. Context includes repeated filler text (“*The grass is green...*”). Evaluated using IoU. Requires tracking dependencies across the context,

⁷<https://github.com/mrmaxguns/wonderwordsmodule>

demanding **High Dispersion** (information location depends on chains) and **Low Scope** (only specific chains matter). See Appendix G.4.

A.2.3 Semi-Synthetic – Story Tasks

These tasks use procedurally generated multi-chapter narratives that scale with sequence length. Each chapter follows a schema involving travel, dialogue, and item transactions. See Appendix F for an example narrative.

- **Story Retrieval:** Answer 16 factoid questions (e.g., location visited, item acquired) about specific chapters, with chapter IDs provided in the questions. Evaluated using Exact Match Accuracy. Requires accessing specific chapters, characteristic of **Low Dispersion** and **Low Scope**. See Appendix G.5.
- **Story Filtering:** Identify the three specific chapters where no item purchases occurred. The prompt explicitly asks for these three chapter IDs, and the narrative is constructed such that exactly three chapters meet this condition. Evaluated using IoU. Requires checking all chapters, demanding **Low Dispersion** (information is chapter-based) but **High Scope** (all chapters must be checked). We found this task to be challenging even for the largest models evaluated. See Appendix G.6.
- **Story Multi-hop:** Given a target item, identify the item acquired immediately before it, requiring reasoning across the transaction history in multiple chapters. In our setup, an item is acquired in every chapter; this simplifies the task to locating the chapter where the target item was acquired and retrieving the item name from the immediately preceding chapter. We found this simplified version to be highly challenging, even for the largest models evaluated, thus we did not explore more complex variants (e.g., selective item acquisition requiring longer lookbacks). Evaluated using Exact Match Accuracy. Requires tracking history across the narrative, demanding **High Dispersion** (relevant transactions can be far apart) and **Low Scope** (only specific transaction pairs matter). See Appendix G.7.

B Overview of Scaling Laws Studies

Scaling laws research aims to model the relationship between model performance and various factors such as model size, training data, and inference budget. As Li et al. (2025a) comprehensively discuss, these studies typically involve regressing performance metrics onto those factors using data from systematic experiments. We briefly outline key aspects of this field to situate our contribution within the broader scaling laws landscape.

Performance metrics The choice of target metric depends on the controlled factors. For studies varying model size or data quantity, language modelling metrics like perplexity are common. For supervised models or capability-specific studies (e.g., RAG performance in Yue et al. (2025)), task accuracy is typically used. In our work, we measure performance using downstream accuracy on long-context benchmarks (QA, RULER, Story), which directly assess the capabilities we aim to model.

Controlled factors Most scaling laws research focuses on training compute, which scales with model size and parameter count. Recent work has expanded to study model sparsity (MoE or activation sparsity), quantisation precision, and inference budgets. Our work contributes to this last category by controlling attention sparsity (via compression ratio), which directly impacts inference compute and memory requirements, alongside model size and sequence length.

Experimental approach Scaling law data can be gathered either through controlled experiments (Kaplan et al., 2020; Hoffmann et al., 2022) or by analysing existing models across different families (Choshen et al., 2024; Ruan et al., 2024). The former approach is costly but precise, while the latter leverages public data but may be affected by undisclosed differences in training procedures. We conduct controlled experiments with the Qwen 2.5 family across a range of sparse attention algorithms, sparsity levels (compression ratios), sequence lengths, and tasks.

Fitting methodology Common approaches to fitting scaling laws include:

- Observing performance trends to determine the appropriate functional form (e.g., power law, exponential) (Luo et al., 2025). We adopt a log-linear form, assuming a power law relationship for numerical factors.
- Fitting parameters using optimisation algorithms (typically L-BFGS) with robust loss functions like Huber loss (Abnar et al., 2025). We use L-BFGS with Huber loss.
- Managing outliers through careful loss function parameterisation (Frantar et al., 2023) or selective data filtering (Li et al., 2025a). Our use of Huber loss helps mitigate outlier influence.
- Validating on held-out examples, particularly those at the extremes of the parameter range (Yue et al., 2025). We validate by holding out the maximum values along each axis (model size, sequence length, compression ratio).

Our approach aligns with these established practices, as detailed in the main paper and Appendix C.

C Sparse Attention Scaling Laws: Formulation and Fitting Details

This section provides further details on our approach to formulating and fitting scaling laws for sparse attention, complementing the discussion in the main paper (Section 4.4). We aim to predict downstream task accuracy based on model size, sequence length, compression ratio, and task characteristics, focusing on the Qwen 2.5 model family and the best-performing sparse patterns (Vertical-Slash for prefilling, Quest for decoding).

C.1 Scaling law formulation

Inspired by prior work, particularly Yue et al. (2025), we develop a log-linear scaling law formulation suitable for modelling inference performance on long-context tasks:

Target We transform task accuracy $a \in [0, 1]$ to its logit $s = \sigma^{-1}(a) = \log(a) - \log(1 - a)$ to create an unbounded regression target, suitable for linear modelling.

Model size We model the effect of parameter count N as $\alpha \log N$. Based on general observations that larger models perform better, α is expected to be positive.

Sequence length We include a term $\beta \log L$, where L is sequence length. As longer contexts often pose greater challenges, β is expected to be negative.

Compression ratio We incorporate compression ratio $C = \text{FLOPS}_{\text{dense}}/\text{FLOPS}_{\text{sparse}}$ as $\gamma \log C$. Since higher compression (more sparsity) typically degrades performance, γ is expected to be negative.

Task-specific effects We include a task-specific intercept parameter δ_{task} to account for inherent differences in task difficulty and sensitivity to the other factors.

Shared Intercept We include a shared intercept ϵ representing the baseline performance level.

The complete scaling law formula, as presented in the main paper, is:

$$s = \alpha \log N + \beta \log L + \gamma \log C + \delta_{\text{task}} + \epsilon \quad (2)$$

C.2 Fitting procedure

We optimise the scaling law parameters $(\alpha, \beta, \gamma, \{\delta_{\text{task}}\}, \epsilon)$ using the L-BFGS algorithm. To improve robustness against potential outliers in the performance data, we employ a Huber loss function with $\delta = 1$. As noted in the main paper, we perform three separate fitting runs for extrapolation analysis, each time holding out data points corresponding to the maximum value of one axis (model size, sequence length, or compression ratio). We evaluated a Mean Squared Error loss as well,

Table 7: Mean Absolute Error (MAE) between predicted and true accuracy by task when extrapolating along different axes (holding out max value). Lower is better.

		QuALITY	QA SQuAD	TOEFL	CWE	Ruler NIAH	VT	Filtering	Story Multi-hop	Retrieval
Prefill	Model Size	0.16	0.11	0.06	0.11	0.02	0.22	0.24	0.08	0.07
	Sequence Length	0.12	0.08	0.10	0.23	0.16	0.06	0.04	0.06	0.20
	Compression Ratio	0.13	0.10	0.12	0.18	0.24	0.10	0.02	0.05	0.41
Decode	Model Size	0.19	0.15	0.08	0.07	0.04	0.29	0.19	0.19	0.02
	Sequence Length	0.16	0.12	0.17	0.14	0.19	0.08	0.04	0.07	0.24
	Compression Ratio	0.10	0.10	0.08	0.12	0.29	0.10	0.07	0.05	0.15

finding similar results and consistent patterns regarding which tasks and extrapolation axes were more challenging to model accurately.

C.3 Scaling law accuracy (MAE)

As discussed in the main paper, we evaluate the extrapolation capabilities of our fitted scaling laws using Spearman’s ρ (Table 4) and Mean Absolute Error (MAE) on the held-out test sets. Table 7 reports the MAE between the predicted accuracy (transformed back from logit space) and the true accuracy for each task across the different extrapolation axes (holding out max model size, sequence length, or compression ratio).

Lower MAE indicates better predictive accuracy. While the overall fit is strong (high R^2 , see Table 3), the MAE values highlight specific task–axis combinations where extrapolation is more challenging. For instance, extrapolating the accuracy when holding out data for the highest compression ratio yields a relatively high MAE in Story Retrieval during prefilling (0.41) and Ruler NIAH during decoding (0.29). However, no single task or extrapolation axis proves universally difficult across both prefilling and decoding. An interesting observation is the tendency for task–axis pairs with high MAE in prefilling to also exhibit high MAE in decoding (e.g., Story Retrieval on sequence length extrapolation), suggesting that the inherent sensitivity of certain tasks to specific factors persists regardless of whether sparsity is applied during prefilling or decoding.

D FLOPS breakdown

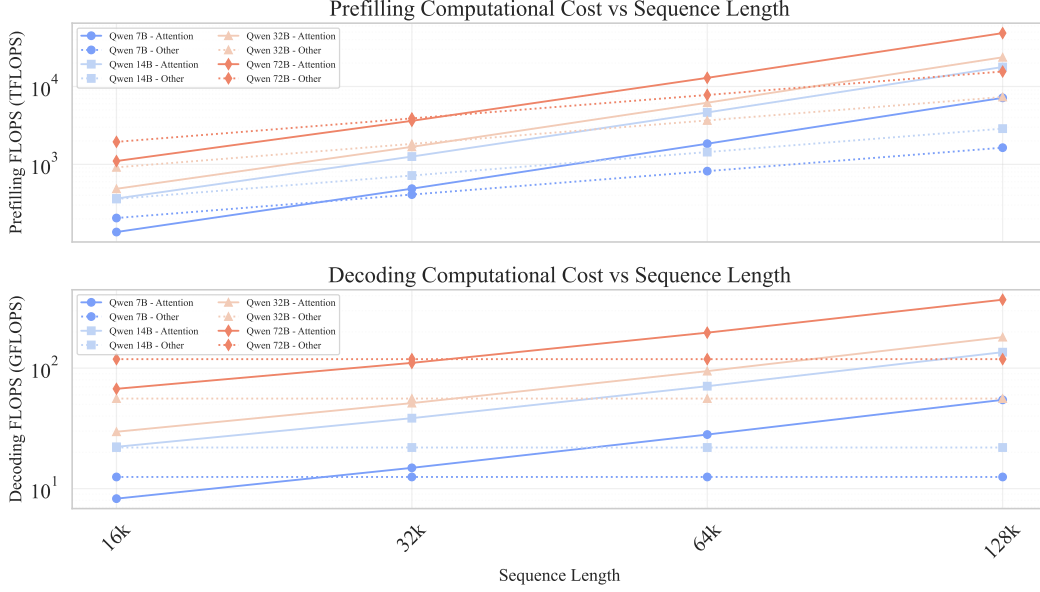


Figure 12: Computational cost of Qwen models (7B to 72B) across sequence lengths from 16K to 128K. Top: Prefilling FLOPS (TFLOPS) showing both attention (solid lines) and non-attention components (dashed lines). Bottom: Decoding FLOPS (GFLOPS) with similar decomposition. Where the lines cross shows the sequence length at which attention FLOPS equal non-attention FLOPS.

We estimate the computational cost of models from Qwen 2.5 family using a component-wise approach:

$$\text{FLOPS}_{\text{total}} = \text{FLOPS}_{\text{embedding}} + \text{FLOPS}_{\text{attention}} + \text{FLOPS}_{\text{mlp}} + \text{FLOPS}_{\text{logits}} \quad (3)$$

$$\text{FLOPS}_{\text{embedding}} = 2 \cdot L \cdot d \quad (4)$$

$$\text{FLOPS}_{\text{attention}} = N \cdot (2Ld^2 + 2L^2d \cdot \rho + 3hL^2 \cdot \rho + 2L^2d \cdot \rho) \quad (5)$$

$$\text{FLOPS}_{\text{mlp}} = N \cdot (4Ld \cdot d_{\text{mlp}} + 2L \cdot d_{\text{mlp}}) \quad (6)$$

$$\text{FLOPS}_{\text{logits}} = 2 \cdot L \cdot d \cdot |V| \quad (7)$$

Where L is sequence length, d is hidden dimension, h is number of heads, N is number of layers, d_{mlp} is MLP intermediate dimension, $|V|$ is vocabulary size, and ρ represents attention density (1-sparsity). For decoding the typical generation length (in our case 200 tokens on average) is orders of magnitude shorter than prefilling lengths. Therefore, we estimate the decoding FLOPS by calculating the cost of processing one new token while attending to the entire context of length L .

For sparse attention methods, we also account for the computational cost of importance estimation—the process of determining which subset of the attention matrix to compute. For Vertical-Slash pattern, this includes dot products between keys and a subset of queries, softmax operations, aggregation over queries and diagonals, top-k selection, and index building. The indexing cost is approximately:

$$\text{FLOPS}_{\text{VS indexing}} = N \cdot h \cdot [2dLq + 3Lq + 2Lq + 2L \log_2(L) + \frac{L}{64}(k_v + k_s)] \quad (8)$$

Where q is the number of last queries used for importance estimation, and k_v and k_s are the number of vertical and slash patterns selected.

For Quest during decoding, the indexing cost involves computing importance scores for each page in the KV cache:

$$\text{FLOPS}_{\text{Quest indexing}} = N \cdot h \cdot [2d \cdot \frac{L}{p} + 3d \cdot \frac{L}{p} + \frac{L}{p} \log_2(\frac{L}{p})] \quad (9)$$

Where p is the page size (16 in our case). These indexing costs are included in our isoFLOPS analysis to ensure fair comparison between dense and sparse attention methods.

Figure 12 illustrates the FLOPS required for both prefilling and decoding operations, revealing two distinct computational regimes in transformer models: a linear-dominated regime at shorter sequences (below 16K) where non-attention components (embedding, MLP, and output layers) determine the computational cost, and a quadratic-dominated regime at longer sequences (beyond 16K-48K, depending on model size) where attention computation with its $O(L^2)$ complexity becomes the primary factor driving the total computational requirements.

A critical insight emerges in the quadratic-dominated regime: at sufficient sequence lengths, larger models with sparse attention can require fewer FLOPS than smaller dense models. This efficiency crossover occurs because sparsity substantially reduces the attention cost which scales faster than the linear components. This effect becomes more pronounced as sequence length increases, creating an increasingly favorable computational trade-off for sparse large models over dense small ones.

E Prompt Template

Input format:

You are provided with a task introduction, context, and a question.

{task_intro}

Below is your question. I will state it both before and after the context.

```
<question>
{question}
</question>
```

```
<context>
{context}
</context>
```

```
<question_repeated>
{question}
</question_repeated>
```

Instructions:

1. First, provide a brief explanation of your reasoning process. Explain how you identified the relevant information from the context and how you determined your answer.
2. Then, provide your final answer following this exact format:

```
<answer>
{answer_format}
</answer>
```

Your response must follow this structure exactly:

```
<explanation>
Your explanation here...
</explanation>
<answer>
Your answer here...
</answer>
```

Important:

{extra_instructions}

- Keep your explanations clear, coherent, concise, and to the point.
- Do not include any additional text, explanations, or reasoning in the answer section.

F Example Story Narrative

Chapter 1:

Beneath gentle breezes, Arion ventured into Athens, curious about its secrets. Long journeys had led Arion to Athens, a step closer to understanding. Soon enough, a tense negotiation seized everyone's attention. Cleo appeared as if expecting Arion, engaging them without delay. Carefully, they navigated the topic of old feuds, wary of awakening dormant animosities that still simmered. In a calm moment, they compared notes on the traders who passed through Athens, each leaving their subtle mark. In hushed tones, they spoke of local customs and distant rumors, sharing hints of hidden pathways. Following subtle bargaining with Cleo, Arion claimed ownership of lavish crystal lamp. With a light gesture, Arion acknowledged Cleo once more before departing. Nothing would be the same as Arion left Athens, thoughts turning inward. In quiet corners, ambitions simmered, waiting for a spark.

Chapter 2:

At dawn, Arion reached the gates of Hippo Regius, where merchants and travelers converged. This place might hold a clue Arion had long sought. Hardly had Arion arrived before a violent storm stirred uneasy whispers. Thanos approached Arion, eyes bright with opportunity. They lingered over tales of old alliances and forgotten disputes, weaving past into present. They debated the meaning of recent events, each seeking patterns in the chaos. Their reflections turned to the interplay of supply and demand, seeing how fortunes might turn in an instant. After reaching terms with Thanos, Arion took possession of ceremonial gold seal. Arion turned from Thanos, ready to move on. In parting, Arion acknowledged that the journey still had far to run. Hidden corners of the city promised knowledge or peril.

Chapter 3:

The threshold of Emerita Augusta welcomed Arion, who felt the weight of untold stories. Arion came here hoping to learn something new, or perhaps gain an advantage. Within hours, a violent storm disrupted the familiar routines. There, Arion encountered Niko, who seemed eager to exchange words or goods. Their words lingered on rumors of distant lands, where fortunes or ruin awaited bold seekers. They debated the meaning of recent events, each seeking patterns in the chaos. Their dialogue danced around subtle clues, each suggestion hinting at treasures undiscovered. The transaction concluded with Arion acquiring delicate porcelain sword from Niko. With a light gesture, Arion acknowledged Niko once more before departing. Eventually, Arion moved on, carrying new impressions forward. The distant hum of voices hinted at unseen deals.

Chapter 4:

Under fading daylight, Arion set foot in Berenice, eager to learn what it offered. A quiet determination brought Arion to Berenice, ever searching for meaning. A sudden market crash cast its shadow over Berenice, changing plans and minds. Roxana approached Arion, eyes bright with opportunity. Together, they reflected on the nature of trust and deceit, aware that fate often twists. They compared accounts of strange visitors bearing knowledge or confusion, each arrival a new riddle in Berenice. A short exchange revealed uncharted corners of Berenice, where knowledge or secrets might dwell. mystic bronze lamp changed hands as Arion completed the purchase from Roxana. Arion handed over lavish crystal lamp to Roxana as the deal closed. With a light gesture, Arion acknowledged Roxana once more before departing. As Arion prepared to depart, the path ahead remained uncertain but compelling. Somewhere, a whisper promised answers for those who dared.

Chapter 5:

Under fading daylight, Arion set foot in Syracuse, eager to learn what it offered. In pursuit of truth, Arion looked to Syracuse for subtle revelations. Not long after arriving, an opulent banquet shook the local order. Phaedra appeared as if expecting Arion, engaging them without delay. Their words traced over delicate negotiations that had once sealed lasting truces in Syracuse. Carefully, they navigated the topic of old feuds, wary of awakening dormant animosities that still simmered. They delved into the subtle art of earning trust in a place where trust was scarce and hard-won. With measured consideration, Arion purchased engraved emerald goblet from Phaedra, examining it closely. In quiet understanding, Arion left Phaedra, their paths diverging. In parting, Arion acknowledged that the journey still had far to run. A subtle tension lingered, as though fate held its breath.

G Example Task Inputs

G.1 Question Answering (QA)

Input format:

I will provide you with multiple documents and ask you a question about one specific document.

Below is your question. I will state it both before and after the context.

<question>

Question about document 39:

Who works to get workers higher compensation?

</question>

<context>

Document 1:

[...text omitted...]

Document 39:

Jobs with high demand and low supply pay more. Professional and labor organizations can raise wages by limiting worker supply and using collective bargaining or political influence.

Document 47:

[...text omitted...]

</context>

<question_repeated>

Question about document 39:

Who works to get workers higher compensation?

</question_repeated>

Instructions:

1. Provide a brief explanation of your reasoning process.

2. Then, give your final answer in this format:

<answer>

Your answer here...

</answer>

Your response must follow this structure:

<explanation>

Your explanation here...

</explanation>

<answer>

Your answer here...

</answer>

Important:

- Do not use complete sentences in the answer.
- For dates: Include ONLY the COMPLETE date if specifically asked.
- For locations: Use the shortest unambiguous form (e.g., 'New York' not 'New York City').
- For comparisons: State ONLY the answer that matches the criteria
- Keep your explanations clear, coherent, concise, and to the point.
- Do not include any additional text, explanations, or reasoning in the answer section.

Example answer:

<explanation>

I found the relevant sentence in document 39, which states that professional and labor organizations help increase wages using bargaining and political means.

</explanation>

<answer>

Professional and labor organizations

</answer>

G.2 RULER - Needle-in-a-Haystack (NIAH)

Input format:

I will provide you with a document containing multiple key-value pairs.
Your task is to extract specific values associated with given keys.

Below are your questions. I will state them both before and after the context.

<questions>

Extract the values for the following keys:

key-A, key-B, key-C, key-D

</questions>

<context>

The value for key-A is: value-A.

The value for key-X is: value-X.

The value for key-B is: value-B.

The value for key-Y is: value-Y.

The value for key-C is: value-C.

The value for key-Z is: value-Z.

The value for key-D is: value-D.

</context>

<questions_repeated>

Extract the values for the following keys:

key-A, key-B, key-C, key-D

</questions_repeated>

Instructions:

1. First, provide a brief explanation of your reasoning process. Explain how you identified the relevant information from the context and how you determined your answer.

2. Then, provide your final answer following this exact format:

<answer>

1. The answer for <key1> is <value1>.

2. The answer for <key2> is <value2>.

etc.

</answer>

Your response must follow this structure exactly:

<explanation>

Your explanation here...

</explanation>

<answer>

Your answer here...

</answer>

Important:

- Provide answers in the exact order of the requested keys
- Each answer must follow the format: "<number>. The answer for <key> is <value>."
- Ensure exact key matches - do not modify or paraphrase the keys
- Values must match exactly as they appear in the document
- Keep your explanations clear, coherent, concise, and to the point.
- Do not include any additional text, explanations, or reasoning in the answer section.

Example answer:

<explanation>

I scanned the context for exact matches of the requested keys. For each key, I extracted the value as stated directly after the pattern "The value for key-X is: ...".

</explanation>

<answer>

1. The answer for key-A is value-A.

2. The answer for key-B is value-B.

3. The answer for key-C is value-C.

4. The answer for key-D is value-D.

</answer>

G.3 RULER - Common Word Extraction (CWE)

Input format:

You will be given a numbered list of words. Your task is to identify the most frequently occurring words. You should solve this task by carefully reading and analyzing the word list. Do not attempt to write code or use programming tools to count frequencies. This is a test of your ability to track word frequencies directly.

Below is your question. I will state it both before and after the context.

<question>

The list contains exactly 10 words that appear 30 times each.

All other words appear 3 times each.

Your task is to identify the 10 words that appear 30 times each.

</question>

<context>

1. alpha

2. beta

3. gamma

4. delta

5. alpha

6. epsilon

...

[...list continues with randomized repeated words...]

...

N. gamma

</context>

<question_repeated>

The list contains exactly 10 words that appear 30 times each.

All other words appear 3 times each.

Your task is to identify the 10 words that appear 30 times each.

</question_repeated>

Instructions:

1. First, provide a brief explanation of your reasoning process.

Explain how you identified the relevant information from the context and how you determined your answer.

2. Then, provide your final answer following this exact format:

<answer>

1. word_one

2. word_two

...

10. word_ten

</answer>

Your response must follow this structure exactly:

<explanation>

Your explanation here...

</explanation>

<answer>

Your answer here...

</answer>

Important:

- List exactly 10 words, one per line, numbered from 1 to 10.

- Keep your explanations clear, coherent, concise, and to the point.

- Do not include any additional text, explanations, or reasoning in the answer section.

Example answer:

<explanation>

I scanned the word list and tracked the frequency of each word.

The following 10 words appeared 30 times each, which I confirmed by careful counting.

</explanation>

<answer>

1. diligent

2. ash

3. pour

4. chateau

5. marble

6. laparoscope

7. grub

8. vinyl

9. mobility

10. kettledrum

</answer>

G.4 RULER - Variable Tracking (VT)

Input format:

I will provide you with a text containing variable assignments. The text contains two types of assignments:

1. Numeric assignments that set a variable to a number (e.g., "VAR ABC = 12345")
 2. Copy assignments that set a variable equal to another variable (e.g., "VAR XYZ = VAR ABC")
- Variables are sequences of uppercase letters. The assignments can appear in any order in the text.

Below is your question. I will state it both before and after the context.

<question>

Which variables resolve to the value 41015? A variable resolves to 41015 if it is either directly assigned 41015, or assigned to another variable that resolves to 41015.

</question>

<context>

VAR A = VAR B
VAR B = 41015
VAR C = VAR D
VAR D = VAR B
VAR E = 12345
VAR F = VAR G
VAR G = VAR H
VAR H = VAR B

</context>

<question_repeated>

Which variables resolve to the value 41015? A variable resolves to 41015 if it is either directly assigned 41015, or assigned to another variable that resolves to 41015.

</question_repeated>

Instructions:

1. First, provide a brief explanation of your reasoning process. Explain how you identified the relevant information from the context and how you determined your answer.
2. Then, provide your final answer following this exact format:

<answer>

VARIABLE_ONE VARIABLE_TWO etc.

</answer>

Your response must follow this structure exactly:

<explanation>

Your explanation here...

</explanation>

<answer>

Your answer here...

</answer>

Important:

- List ONLY the variable names that resolve to the target value.
- Variables can be listed in any order.
- Do not include "VAR" prefix in your answer. Do not include punctuation.
- Keep your explanations clear, coherent, concise, and to the point.
- Do not include any additional text, explanations, or reasoning in the answer section.

Example answer:

<explanation>

I traced each variable assignment to see if it leads to the value 41015. B is directly assigned 41015. A, D, and H point to B. C and G point to D and H, respectively. So A B C D G H resolve to 41015.

</explanation>

<answer>

A B C D G H

</answer>

G.5 Story Retrieval

Input format:

You are given a narrative composed of multiple chapters. Throughout these chapters, the protagonist travels between different locations, meets various characters, and engages in trading activities. All items mentioned in the narrative are unique, and their ownership can change through trades. Your task is to carefully read the narrative and answer the questions based on the provided information.

Below are your questions. I will state them both before and after the context.

<questions>

1. In Chapter 3, which character did the protagonist interact with?
2. In Chapter 5, which specific item was acquired by the protagonist?
3. In Chapter 7, which specific location did the protagonist visit?

</questions>

<context>

Chapter 1:

[...text omitted...]

Chapter 3:

Arion entered Babylon and met Thanos. After exchanging stories, Arion acquired a silver idol.

Chapter 5:

In Berenice Troglodytica, Arion encountered Xanthe and traded for a golden vase.

Chapter 7:

Delphi welcomed Arion with quiet mystery. A meeting with Vitalis ended with a jade idol.

</context>

<questions_repeated>

1. In Chapter 3, which character did the protagonist interact with?
2. In Chapter 5, which specific item was acquired by the protagonist?
3. In Chapter 7, which specific location did the protagonist visit?

</questions_repeated>

Instructions:

1. First, provide a brief explanation of your reasoning process. Explain how you identified the relevant information from the context and how you determined your answer.
2. Then, provide your final answer following this exact format:

<answer>

1. ANSWER_ONE
 2. ANSWER_TWO
- etc.

</answer>

Your response must follow this structure exactly:

<explanation>

Your explanation here...

</explanation>

<answer>

Your answer here...

</answer>

Important:

- For answers, use one line per answer with the number prefix
- Do not include articles like 'the' or 'a' in answers
- Answers should be specific names/items/locations mentioned in the text
- Keep your explanations clear, coherent, concise, and to the point.
- Do not include any additional text, explanations, or reasoning in the answer section.

Example answer:

<explanation>

I located Chapter 3 in the context and identified Thanos as the mentioned character.

In Chapter 5, Arion acquired a golden vase from Xanthe.

Chapter 7 stated that Arion visited Delphi, so I used that as the answer.

</explanation>

<answer>

1. Thanos
2. Golden Vase
3. Delphi

</answer>

G.6 Story Filtering

Input format:

You are given a narrative composed of multiple chapters. Throughout these chapters, the protagonist travels between different locations, meets various characters, and engages in trading activities. All items mentioned in the narrative are unique, and their ownership can change through trades. Your task is to carefully read the narrative and answer the questions based on the provided information.

Below is your question. I will state it both before and after the context.

<question>

Identify all chapters where the protagonist did not buy any item.

Note: There are exactly 2 chapters without any purchases.

</question>

<context>

Chapter 1:

[... Arion visits Athens and purchases a crystal lamp ...]

Chapter 2:

[... Arion travels to Hippo Regius and buys a gold seal ...]

Chapter 3:

[... Arion enters Babylon and engages in an ongoing event but do not buy anything ...]

Chapter 4:

[... Arion arrives in Pergamon and has conversations, but no purchases are mentioned ...]

Chapter 5:

[... Arion goes to Delphi and buys a jade idol ...]

</context>

<question_repeated>

Identify all chapters where the protagonist did not buy any item.

Note: There are exactly 2 chapters without any purchases.

</question_repeated>

Instructions:

1. First, provide a brief explanation of your reasoning process. Explain how you identified the relevant information from the context and how you determined your answer.
2. Then, provide your final answer following this exact format:

<answer>

chapter_id_1, chapter_id_2, ...

</answer>

Your response must follow this structure exactly:

<explanation>

Your explanation here...

</explanation>

<answer>

Your answer here...

</answer>

Important:

- In the answer section, provide only the chapter IDs separated by commas.
- Keep your explanations clear, coherent, concise, and to the point.
- Do not include any additional text, explanations, or reasoning in the answer section.

Example answer:

<explanation>

I scanned each chapter to check whether a purchase by the protagonist was explicitly described. In Chapter 3 and 4, no item acquisition are mentioned. Other chapters include phrases like "Arion purchased" or "Arion acquired", indicating a transaction.

</explanation>

<answer>

3, 4

</answer>

G.7 Story Multi-hop

Input format:

You are given a narrative composed of multiple chapters. Throughout these chapters, the protagonist travels between different locations, meets various characters, and engages in trading activities. All items mentioned in the narrative are unique, and their ownership can change through trades. Your task is to carefully read the narrative and answer the questions based on the provided information.

Below is your question. I will state it both before and after the context.

<question>
What was the last item that the protagonist acquired before acquiring timeworn amber sword?
</question>

<context>
Chapter 1:
[... narrative text omitted for brevity ...]

Chapter 17:
The transaction concluded with Arion acquiring pristine bronze seal from Damon.

Chapter 18:
After reaching terms with Marcus, Arion took possession of timeworn amber sword.
</context>

<question_repeated>
What was the last item that the protagonist acquired before acquiring timeworn amber sword?
</question_repeated>

Instructions:
1. First, provide a brief explanation of your reasoning process. Explain how you identified the relevant information from the context and how you determined your answer.
2. Then, provide your final answer following this exact format:
<answer>
ITEM_NAME
</answer>

Your response must follow this structure exactly:

<explanation>
Your explanation here...
</explanation>
<answer>
Your answer here...
</answer>

Important:
- Provide only the item name in the answer section.
- Do not include articles like 'the' or 'a' in your answer.
- The item name must be exactly as mentioned in the text.
- Keep your explanations clear, coherent, concise, and to the point.
- Do not include any additional text, explanations, or reasoning in the answer section.

Example answer:

<explanation>
I located the chapter where the protagonist acquired the timeworn amber sword. Then, I scanned earlier chapters to find the most recent prior acquisition, which occurred in Chapter 17 with the item pristine bronze seal.
</explanation>
<answer>
pristine bronze seal
</answer>