POEM: OUTPUT-RELATIVE BLOCK-SPARSE ATTENTION FOR LONG-CONTEXT PREFILLING

Anonymous authors
Paper under double-blind review

ABSTRACT

As large language models (LLMs) evolve to support increasingly long contexts, accelerating prefilling without compromising output accuracy becomes a critical challenge. Block-sparse attention addresses this by retaining only the most important key-value blocks, but existing importance proxies often discard blocks and cause output errors. These proxies generally estimate a block's contribution to the dense attention output (i.e., attention-weighted value), implicitly assuming that removing the block loses exactly this contribution. In this work, we show that this assumption does not hold because attention renormalization amplifies the retained blocks' contributions. As a result, the output error depends not only on the removed attention mass but also on the distance between the removed block's output and the dense output. We call the distance *output deviation* and empirically find that it varies by over $20\times$ even among blocks with comparable attention mass, so existing proxies often discard blocks whose large output deviation leads to large output errors. To address this gap, we propose *Prefilling via Output Error Minimization (POEM)*, a training-free method that combines each block's contribution and output deviation into an output-relative score, reducing the output error of each head. To connect the head-level optimization to the model's predictions, we derive a bound on the prediction divergence that weights each head's output error by its sensitivity, which guides POEM to calibrate the retention threshold of each head. On long-context benchmarks, POEM maintains accuracy parity with dense attention while achieving up to a $6.5\times$ attention speedup over FlashAttention-2.

1 INTRODUCTION

Large language models (LLMs) (Brown et al., 2020; Touvron et al., 2023) have achieved striking advances in recent years, demonstrating strong performance across a wide range of tasks. These advances have enabled their deployment in increasingly ambitious applications, from scientific discovery (Novikov et al., 2025; Yamada et al., 2025) to autonomous agents (Wang et al., 2023; Zhang et al., 2026). Such applications often demand processing information across extremely long sequences such as extensive scientific corpora and long-horizon task trajectories (Bai et al., 2025). However, as context lengths continue to grow, the quadratic cost of self-attention (Vaswani et al., 2017) makes prefilling prohibitively expensive, rendering attention the dominant computational bottleneck in LLM inference (Fu, 2024). To alleviate this bottleneck, recent studies have developed efficient attention mechanisms (Yuan et al., 2025; Hatamizadeh et al., 2026; Sun et al., 2025), ranging from sparsifying token interactions to reformulating attention with subquadratic alternatives.

Among these mechanisms, block-sparse attention is particularly attractive, as it retains exact attention over the relevant regions while entirely skipping the rest. Block-sparse attention (Child et al., 2019) operates by partitioning the key–value sequence into fixed-size blocks, retaining only important blocks for each query, and computing exact attention over the retained subset. A key challenge is to accurately identify the most important blocks without computing the full attention matrix (Roy et al., 2021). Recent works on block-sparse attention have explored a wide range of strategies. Some methods incorporate block selection into model training (Yuan et al., 2025; Lu et al., 2026) or train auxiliary gates to predict block importance (Gao et al., 2024b). However, these methods incur additional training costs, complicating their application to existing pretrained LLMs. To avoid this overhead, training-free methods instead estimate block importance using lightweight proxies (Lai et al., 2025). These methods select blocks by measuring their contribution to dense attention. Most

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

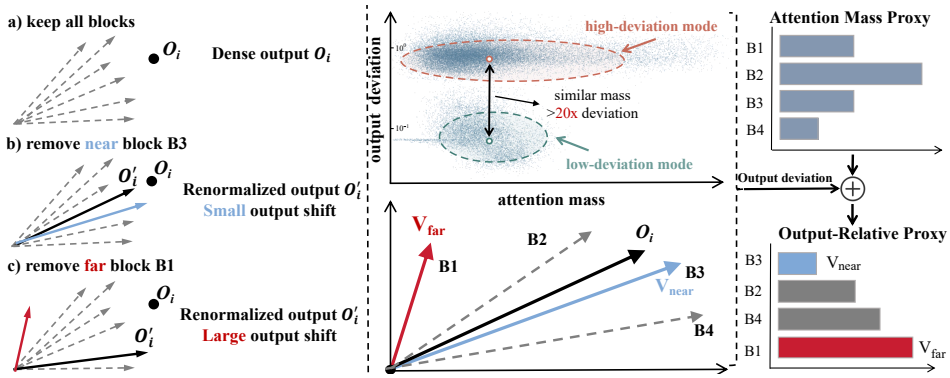

Figure 1: Through experiments, we observe that blocks with similar attention mass can differ by over 20× in output deviation. Removing a high-deviation block (B1) shifts the attention output far more than removing a low-deviation one (B3) of comparable mass. We therefore combine attention mass with output deviation into our output-relative proxy. V denotes the exact output of a block.

of them estimate attention mass (Jiang et al., 2024; Lai et al., 2025; Xu et al., 2025; Wang et al., 2026a), while STEM (Niu et al., 2026) further incorporates value to account for differences in value contributions. Nevertheless, a block's contribution to dense attention is not equivalent to the output shift by its removal. When a block is dropped, softmax renormalization amplifies the contributions of the retained blocks, compensating for part of the removed blocks' contribution.

So what determines whether a removed block's contribution is easily compensated? As Figure 1 (left) illustrates, if a block's output lies close to the dense output, renormalization amplifies the retained blocks' contributions while largely preserving the output direction, resulting in only a small output shift. However, if the block's output points away from the dense output, the same amplification cannot compensate for its removal, and the output shifts substantially. We call the distance between a block's output and the dense output the block's *output deviation*. Empirically, we observe that output deviation varies by more than 20× across blocks of comparable attention mass, forming distinct low-deviation and high-deviation modes (center). Therefore, existing proxies that rely on block contribution alone cannot distinguish safely removable blocks from critical ones.

To address this gap, we propose *Prefilling via Output Error Minimization (POEM)*, a training-free block-sparse attention method. **(1) Output-relative block selection:** Motivated by the renormalization analysis above, we derive a head-level output-relative score that combines each block's attention mass with the deviation of its output from the dense head output. A two-stage pipeline makes this score practical: mass estimates identify a candidate pool at low cost, and the output-relative score then selects the final blocks within it. To connect this head-level optimization to model predictions, we derive a theoretical bound on predictive divergence that weights each head's output error by its sensitivity. This bound motivates **(2) Offline threshold calibration:** POEM calibrates a separate retention threshold for each head group according to its sensitivity. Because calibrating every head group independently would be prohibitively expensive, POEM ranks heads by sensitivity and jointly calibrates similarly ranked heads via an efficient batched bisection procedure.

Experiments on LongBench (Bai et al., 2024) and RULER (Hsieh et al., 2024) using Llama-3.1-8B (Grattafiori et al., 2024), Qwen3-8B, and Qwen3-14B (Yang et al., 2025) show that POEM maintains accuracy comparable to that of full attention at high sparsity. On RULER at 128K context, POEM achieves a 2.4-point gain over our mass-only baseline, demonstrating the effectiveness of the output-relative score. POEM also achieves the lowest attention latency among the compared methods, achieving up to a 6.5× speedup over FlashAttention-2 (Dao, 2024) at 128K context.

Our contributions can be summarized as:

- We propose POEM, a training-free block-sparse attention method that combines attention mass with output deviation for output-relative block selection, optimizing output error within heads.
- To bridge head-level optimization and model-level prediction, we derive a bound on output divergence in terms of head error weighted by sensitivity, motivating our offline threshold calibration.
- Across three LLMs on LongBench and RULER, POEM almost matches full-attention accuracy at high sparsity and achieves up to a 6.5× attention speedup at 128K prefill.

2 Related Work

Training-free sparse attention. Training-free methods use the sparsity already present in pretrained attention. For decoding, Quest (Tang et al., 2024) selects KV entries by query-aware importance, and Twilight (Lin et al., 2026) adapts the sparsity budget. Prefilling must select blocks for every query position, so the selector itself has to be cheap yet accurate. MInference (Jiang et al., 2024) assigns sparse patterns to heads offline, whereas FlexPrefill (Lai et al., 2025) adapts patterns and budgets to each input. To score blocks, XAttention (Xu et al., 2025) sums antidiagonal entries, SALE (Ji et al., 2025) uses low-bit estimates, Sketch&Walk (Le et al., 2026) propagates randomized sketches across layers, UniSparse (Liu et al., 2025b) compresses and routes at multiple granularities, and Prism (Wang et al., 2026a) keeps high-frequency positional signals with spectral-aware scores. Other work lowers cost elsewhere: SpargeAttention (Zhang et al., 2025) co-designs the selector with the kernel, Pbs-Attn (Wang et al., 2026b) packs important tokens into fewer blocks, and ProxyAttn (Wang et al., 2026c) lets a few representative heads score for the rest. These selectors rank blocks mainly by attention mass, and STEM (Niu et al., 2026) adds value norms to estimate how much each block contributes to dense attention. POEM instead aims to minimize the output error caused by removal, combining attention mass and output deviation together to select the blocks.

Training-based sparse attention. Training-based methods learn sparsity patterns during pretraining or adaptation. Earlier, Routing Transformer (Roy et al., 2021) clustered queries and keys, and Landmark Attention (Mohtashami & Jaggi, 2023) retrieved blocks via learned landmark tokens. More recent designs include Native Sparse Attention (Yuan et al., 2025), which combines compression, selection, and local attention; MoBA (Lu et al., 2026), which routes each query to its top-k blocks. InfLLM-V2 (Zhao et al., 2026), which switches between dense and sparse attention depending on sequence length, and HiLS (Hu et al., 2026), which learns hierarchical retrieval from the language-modeling loss. SeerAttention (Gao et al., 2024b) instead trains block-level gates on a frozen backbone for prefilling, and SeerAttention-R (Gao et al., 2025) extends them to decoding. The selectors also differ in supervision: DeepSeek Sparse Attention (Liu et al., 2025a) uses a token-level indexer with attention scores, whereas SAS (Li et al., 2026) ranks context with the language-modeling loss. All of these methods train either the model or an auxiliary selector, whereas POEM is training-free and applies directly to pretrained models at no additional training cost.

3 Preliminaries

Attention and block-sparse formulation. Let $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N \times d_h}$ be the query, key, and value matrices with row vectors $\mathbf{q}$, $\mathbf{k}$, $\mathbf{v}$, where N is the sequence length and d_h the head dimension. Under causal masking, the attention weight and output are defined for each query position as follows:

$$A_{it} = \frac{\exp(\mathbf{q}_i \mathbf{k}_t^\top / \sqrt{d_h})}{\sum_{u=1}^{i} \exp(\mathbf{q}_i \mathbf{k}_u^\top / \sqrt{d_h})}, \qquad \mathbf{o}_i = \sum_{t=1}^{i} A_{it} \mathbf{v}_t. \tag{1}$$

Block-sparse attention (Xu et al., 2025) partitions KV sequences into B blocks $\{\mathcal{B}_b\}_{b=1}^{B}$ and restricts each query i to a retained block set $\mathcal{S}_i \subseteq \{1, \dots, B\}$. For block b, define the block attention mass $m_{ib} = \sum_{t \in \mathcal{B}_b} A_{it}$ and the block output $\bar{\mathbf{v}}_{ib} = m_{ib}^{-1} \sum_{t \in \mathcal{B}_b} A_{it} \mathbf{v}_t$, so that $\mathbf{o}_i = \sum_b m_{ib} \bar{\mathbf{v}}_{ib}$. Given the discarded mass $\delta_i = \sum_{b \notin \mathcal{S}_i} m_{ib}$, the renormalised sparse output and the projected output of a layer ℓ with H heads are respectively given by the following expressions:

$$\widetilde{\mathbf{o}}_i = \frac{\sum_{b \in \mathcal{S}_i} m_{ib} \bar{\mathbf{v}}_{ib}}{1 - \delta_i}, \qquad \mathbf{o}_\ell^{\text{out}} = \sum_{h=1}^{H} \mathbf{o}_{\ell,h} \, \mathbf{W}_{O,\ell}^{(h)}, \tag{2}$$

where $\mathbf{W}_{O,\ell}^{(h)} \in \mathbb{R}^{d_h \times d}$ projects the output $\mathbf{o}_{\ell,h}$ of head h to the model dimension d.

4 Method

In this section, we first present the output-relative block score (Section 4.1). We then introduce a hierarchical error decomposition (Section 4.2) that connects predictive divergence to head output error, followed by the offline headwise threshold calibration (Section 4.3). Finally, we summarize the resulting POEM pipeline with two stages (Section 4.4) and show it in Figure 2.

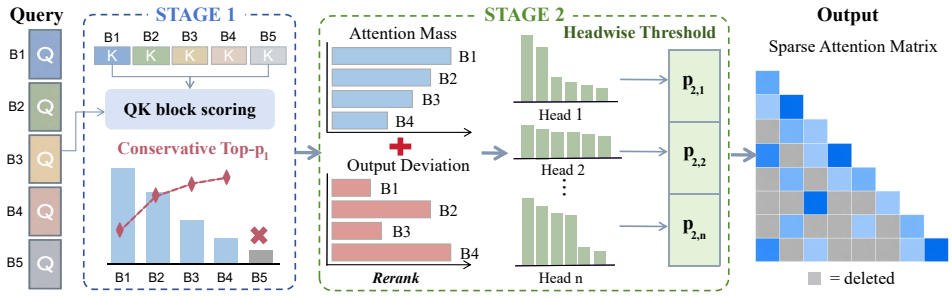

Figure 2: Overview of POEM. Stage 1 preselects candidate blocks using coarse attention mass. Stage 2 reranks candidates with output-relative scores and selects blocks using calibrated headwise thresholds. p_1 is the Stage 1 threshold, and $p_{2,h}$ is the calibrated threshold for head h.

4.1 Output-relative block selection

A central design question for training-free block-sparse attention is how to reliably select significant blocks under a fixed budget while preserving the dense output. Existing methods such as MInference (Jiang et al., 2024), FlexPrefill (Lai et al., 2025), and XAttention (Xu et al., 2025) score each block solely by the attention mass. STEM (Niu et al., 2026) furthermore augments the QK score with the largest value norm inside the block, which makes the proxy value-aware. However, all of existing methods overlook the effect of renormalization, leaving out output deviation, except POEM. POEM's proxy scores each block by combining attention mass with output deviation, making the proxy both value-aware and output-relative, as summarized in Table 1. This subsection then introduces POEM's output-relative proxy and describes how it is ultimately used in the complete POEM pipeline.

Table 1: Comparison of proxies. $*$ means antidiagonal scoring, and $v^{\max}$ denotes the maximum value norm within the block.

Method	Proxy Score	Value-Aware	Output-Relative
MInference	$\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d}}$	✗	✗
FlexPrefill	$\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d}}$	✗	✗
XAttention	$\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d}}{}^*$	✗	✗
STEM	$\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d}}{}^* + \beta\,\mathrm{ReLU}(\log v^{\max})$	✓	✗
POEM	$\frac{\mathbf{q}\mathbf{k}^\top}{\sqrt{d}}{}^* + \beta \log \left\lVert \mathbf{v} + \frac{\mathbf{q}}{\sqrt{d}}\mathbf{C} - \hat{\mathbf{o}} \right\rVert$	✓	✓

From output error to block importance. Our objective is to select a subset of blocks $\mathcal{S}_i$ that minimizes the output error between the sparse output $\widetilde{\mathbf{o}}_i$ and the dense output $\mathbf{o}_i$. As defined in Section 3, we express this objective as the following subset selection problem:

$$\min_{\mathcal{S}_i} \|\mathbf{o}_i - \widetilde{\mathbf{o}}_i\|_2 = \min_{\mathcal{S}_i} \frac{\left\|\sum_{b\notin\mathcal{S}_i} m_{ib}(\bar{\mathbf{v}}_{ib} - \mathbf{o}_i)\right\|_2}{1 - \sum_{b\notin\mathcal{S}_i} m_{ib}}. \tag{3}$$

The numerator couples all discarded blocks inside a single norm, so the objective cannot be decomposed into independent per-block scores. To measure each block's contribution to the output error, we apply the triangle inequality and obtain:

$$\|\mathbf{o}_i - \widetilde{\mathbf{o}}_i\|_2 \le \frac{\sum_{b\notin\mathcal{S}_i} m_{ib}\,\|\bar{\mathbf{v}}_{ib} - \mathbf{o}_i\|_2}{1 - \sum_{b\notin\mathcal{S}_i} m_{ib}}. \tag{4}$$

Let p denote the overall mass threshold used to determine the retained block set. Under a fixed threshold p, $1 - \sum_{b\notin\mathcal{S}_i} m_{ib}$ remains almost constant around p. Thus, minimizing the bound reduces to minimizing $\sum_{b\notin\mathcal{S}_i} m_{ib}\|\bar{\mathbf{v}}_{ib} - \mathbf{o}_i\|_2$. We therefore adopt $i_{ib} = m_{ib}\|\bar{\mathbf{v}}_{ib} - \mathbf{o}_i\|_2$ as the importance proxy of block b, which combines attention mass and output deviation multiplicatively.

An approximation to the block score. To obtain a proxy compatible with existing top-p kernels, we apply a logarithmic transformation to I_{ib} and express it as the sum of two terms. To avoid the expensive computation of the exact block output, we approximate it with a first-order Taylor expansion, yielding the selection score s_{ib} used for practical candidate ranking:

$$s_{ib} = \frac{\bar{\mathbf{q}}_i \bar{\mathbf{k}}_b^\top}{\sqrt{d_h}} + \log \|\bar{\mathbf{v}}_{ib} - \mathbf{o}_i\|_2, \qquad \bar{\mathbf{v}}_{ib} \approx \bar{\mathbf{v}}_b + \frac{\bar{\mathbf{q}}_i}{\sqrt{d_h}}\mathbf{C}_b, \tag{5}$$

where $\bar{\mathbf{v}}_b$ denotes the pooled value vector in block b, $n_b = |\mathcal{B}_b|$, and $\mathbf{C}_b = \frac{1}{n_b}\sum_{t\in\mathcal{B}_b}(\mathbf{k}_t - \bar{\mathbf{k}}_b)^\top(\mathbf{v}_t - \bar{\mathbf{v}}_b)$ is the key–value covariance of block b. Appendix A provides the theoretical derivations, Appendix B develops the selection score, and Appendix C provides the experimental configuration.

Two-stage selection. Computing the magnitude $\bar{\mathbf{q}}_i\mathbf{C}_b/\sqrt{d_h}$ over all query–key block pairs requires $\mathcal{O}(n^2)$ computation, which is unaffordable. To avoid unnecessary computation on negligible blocks, we restrict this calculation to a small subset of potentially important blocks, which we call the candidate pool. Therefore, we design a selector with two stages. Stage 1 selects the candidates, and Stage 2 further selects significant blocks using output deviation. Given an overall mass target p, Stage 1 applies a conservative threshold $p_1 \geq p$ to retain candidates while forming the coarse reference $\widehat{\mathbf{o}}_i = \sum_{b\leq i} a_{ib}\bar{\mathbf{v}}_b$ as a prior for output $\mathbf{o}_i$, where a_{ib} is the pooled attention weight of query i over its visible blocks. Stage 2 sets $p_2 = p/p_1$ as its threshold, reranks candidates by the output-relative score, and applies p_2 to their renormalized attention weights, retaining nominal mass p under the assumption that discarded blocks remain negligible. Appendix B provides the output reference construction and complete selection procedure. Appendix D verifies the Stage 1 candidate-pool coverage, while Appendix E analyzes the auxiliary memory overhead across sequence lengths.

Output-relative score. To adjust the strength of the output-deviation term and better approximate the actual block importance I_{ib}, we introduce a scalar β on the log term to balance attention mass and output deviation. This gives POEM's selection proxy for practical candidate reranking:

$$s_{ib} = \frac{\bar{\mathbf{q}}_i\bar{\mathbf{k}}_b^\top}{\sqrt{d_h}} + \beta\log\left\|\bar{\mathbf{v}}_b + \frac{\bar{\mathbf{q}}_i}{\sqrt{d_h}}\mathbf{C}_b - \widehat{\mathbf{o}}_i\right\|_2. \tag{6}$$

The first term estimates block-level attention strength, while the second estimates the approximate block output's deviation from $\widehat{\mathbf{o}}_i$. Following XAttention (Xu et al., 2025), we estimate the first term by antidiagonal scoring. These two magnitudes combine to form our output-relative scorer.

Table 2 compares the reconstruction errors of the three selection scores at the same density. POEM consistently outperforms the mass-only baseline and STEM's value-aware proxy at all three evaluated layers and at the final logits, supporting the superiority of the output-relative proxy for block selection. Appendix C provides the detailed evaluation configuration, including the matched block density setting and definitions of the layer and logit reconstruction errors. Figure 4 (d) examines the end-to-end impact of the output-relative proxy.

Table 2: Layer and logit output errors on LongBench, using Qwen3-8B.

Method	Layer 5 ($\times10^{-4}$)	Layer 15 ($\times10^{-2}$)	Layer 35	Final logits
Mass only	3.27	1.24	5.88	2.86
STEM	3.27	1.23	5.88	2.85
POEM	**3.25**	**1.22**	**5.79**	**2.74**

4.2 Hierarchical Error Decomposition

To keep the sparse output close to the dense output at the model level, we hierarchically decompose the prediction error and relate it to head-level reconstruction error. Our overall objective is to select attention blocks that minimize the discrepancy between dense and sparse prediction distributions under a fixed block budget, which can be formalized as the following constrained objective:

$$\min_{\mathcal{S}:\,|\mathcal{S}|=k} d_{\mathrm{KL}}\big(\mathbf{p}^d \,\|\, \mathbf{p}^s(\mathcal{S})\big). \tag{7}$$

Here, $\mathcal{S}$ is the selected block set of size k, $\mathbf{p}^d$ and $\mathbf{p}^s(\mathcal{S})$ are the prediction distributions under dense and sparse attention, $d_{\mathrm{KL}}(\cdot\,\|\,\cdot)$ is the Kullback–Leibler (KL) divergence. The objective is defined on the predictions, whereas block selection is made separately for each head, so we bridge this gap by decomposing the prediction error first into layer-level and then into head-level output errors.

Step 1: Prediction to layer output. The objective in Eq. 7 is defined on the final predictions, but what block selection actually changes is the attention output of a single layer, so we begin by tracing how a perturbation of that output propagates to the predictions. Let $\mathbf{O}_\ell$ and $\widetilde{\mathbf{O}}_\ell$ denote the attention outputs of layer ℓ under dense and sparse attention, $\Delta\mathbf{O}_\ell := \widetilde{\mathbf{O}}_\ell - \mathbf{O}_\ell$ for the layer-output perturbation and $\boldsymbol{\epsilon}_\ell := \mathrm{vec}(\Delta\mathbf{O}_\ell)$ for its vectorization. A first-order expansion of the downstream network gives the change in the final logits $\mathbf{z}$ as $\Delta\mathbf{z} = \mathbf{J}_{\ell\to z}\,\boldsymbol{\epsilon}_\ell + \mathcal{O}\big(\|\boldsymbol{\epsilon}_\ell\|_2^2\big)$, where $\mathbf{J}_{\ell\to z}$ is the Jacobian from the vectorized layer output to the logits. The perturbed logits shift the prediction

distribution from $\mathbf{p}^d = \mathrm{softmax}(\mathbf{z})$ to $\mathbf{p}^s = \mathrm{softmax}(\mathbf{z} + \Delta\mathbf{z})$, and because the KL divergence attains its minimum at $\Delta\mathbf{z} = 0$, its expansion in $\Delta\mathbf{z}$ has no constant or first-order term,

$$d_{\mathrm{KL}}(\mathbf{p}^d \,\|\, \mathbf{p}^s) = \tfrac{1}{2}\,\Delta\mathbf{z}^\top \mathbf{F}\,\Delta\mathbf{z} + \mathcal{O}\big(\|\Delta\mathbf{z}\|_2^3\big)\,, \qquad \mathbf{F} := \mathrm{Diag}(\mathbf{p}^d) - \mathbf{p}^d(\mathbf{p}^d)^\top, \tag{8}$$

where $\mathbf{F}$ is the softmax Fisher matrix, which captures how strongly each direction of logit change perturbs the prediction distribution. We further bound the quadratic form by the spectral norms of $\mathbf{F}$ and $\mathbf{J}_{\ell\to z}$, yielding the scalar upper bound

$$d_{\mathrm{KL}}(\mathbf{p}^d \,\|\, \mathbf{p}^s) \le c_\ell\,\|\Delta\mathbf{O}_\ell\|_F^2 + \mathcal{O}\big(\|\Delta\mathbf{O}_\ell\|_F^3\big)\,, \qquad c_\ell := \tfrac{1}{2}\,\|\mathbf{F}\|_2\,\|\mathbf{J}_{\ell\to z}\|_2^2, \tag{9}$$

where the constant c_ℓ measures how sensitive the predictions are to this layer's output. The full derivation is provided in the theoretical appendix. The layer output still aggregates the contributions of all attention heads, so Step 2 splits this layer-level error further into per-head output errors.

Step 2: Layer to head. Block selection operates per attention head, so the layer-level bound must be carried one step further, down to the heads that compose the layer output. Let $\mathbf{o}_{\ell,h}$ and $\widetilde{\mathbf{o}}_{\ell,h}$ denote head h's dense and sparse outputs before projection, and let $\Delta\mathbf{o}_{\ell,h} := \widetilde{\mathbf{o}}_{\ell,h} - \mathbf{o}_{\ell,h}$ be its output error. Since Eq. 2 writes the layer output as a sum of projected head outputs, the perturbation decomposes as $\Delta\mathbf{O}_\ell = \sum_h \Delta\mathbf{o}_{\ell,h}\mathbf{W}_O^{(h)}$, and applying the triangle and Cauchy–Schwarz inequalities gives

$$\|\Delta\mathbf{O}_\ell\|_F^2 \le H \sum_h \Big\|\mathbf{W}_O^{(h)}\Big\|_2^2\,\|\Delta\mathbf{o}_{\ell,h}\|_F^2\,, \tag{10}$$

where H is the number of heads and $\mathbf{W}_O^{(h)}$ is the row block of the output projection for head h. This bound isolates each head's contribution through its projection norm and output error. Combining this inequality with Eq. 9 bounds the predictive divergence by a weighted sum of head-level errors. The two factors motivate the two components of POEM. Output-relative block selection (Section 4.1) targets the per-head error $\|\Delta\mathbf{o}_{\ell,h}\|_F$. Headwise threshold calibration (Section 4.3) accounts for the unequal head sensitivities represented by the projection weights $\|\mathbf{W}_O^{(h)}\|_2^2$ in the resulting predictive-divergence bound across all attention heads within the layer.

4.3 Offline threshold calibration

Error-ranked batch bisection. Offline calibration replaces the shared $p_2 = p/p_1$ with a head-wise threshold $p_{2,h}$, where $p_{2,h}$ is the Stage 2 threshold of head h. Following GQA's KV grouping (Ainslie et al., 2023) and KV-cache compression methods (Cai et al., 2025; Rehg, 2024), we jointly calibrate groups of heads that share KV projections. To avoid calibrating groups individually, Algorithm 2 in Appendix F sorts groups by their estimated output-distribution error and tests them in batches, starting from all groups. Each trial lowers the batch thresholds. If the score on the search set does not decrease, the reduction is accepted for the entire batch. Otherwise, the batch is bisected and the half with lower error is retried first in subsequent calibration. The calibration ends when all groups have been processed or the predefined maximum number of steps is reached.

4.4 POEM: Output-Relative Block-Sparse Attention

We integrate the output-relative selection score and the offline threshold calibration together into POEM, a unified training-free framework. Figure 2 summarizes the execution order, and Algorithm 1 gives the detailed inference procedure. **First**, POEM selects candidates by lightweight attention mass alone to quickly construct the candidate pool and compute the output reference $\widehat{\mathbf{o}}_i$ for subsequent output-relative ranking. **Second**, POEM computes the output-relative score in Eq. (6) within the candidate pool, which combines attention mass with output deviation to account for each block's importance in reconstructing the output. For a head h, POEM reranks the blocks in the candidate pool by the output-relative score and applies the calibrated threshold $p_{2,h}$ to the reordered sequence, where $p_{2,h}$ is the calibrated threshold.

Algorithm 1 POEM: Output-Relative Block-Sparse Attention

Input: $\mathbf{Q}, \mathbf{K}, \mathbf{V}$.
Parameters: p_1, β, $\{p_{2,h}\}_h$, pooling stride B.
Output: Sparse outputs $\widetilde{\mathbf{o}}_{i,h}$.
for each head h (using head-local $\mathbf{Q}, \mathbf{K}, \mathbf{V}$) **do**
  Cache block covariances $\mathbf{C}_b$.
  for each query block i **do**
    # Stage 1
    $\mathbf{a}_i \leftarrow \mathrm{Pool}(\mathbf{Q}, \mathbf{K}, \mathrm{stride} = B)$. ▷ # Antidiagonal
    $\mathcal{C}_i \leftarrow \mathrm{TopP}(\mathbf{a}_i, p_1)$.
    $\widehat{\mathbf{o}}_i \leftarrow \sum_{b \le i} a_{ib}\bar{\mathbf{v}}_b$.
    # Stage 2
    Compute s_{ib} over $b \in \mathcal{C}_i$ using Eq. 6.
    $Rerank(s_{ib}, \mathcal{C}_i)$.
    $\mathcal{S}_i \leftarrow \mathrm{ThresholdP}(\mathcal{C}_i, p_{2,h})$. ▷ # Mass threshold
    $\widetilde{\mathbf{o}}_{i,h} \leftarrow \mathrm{Attention}(\mathbf{Q}_i, \mathbf{K}, \mathbf{V}; \mathcal{S}_i)$.
  end for
end for

5 EXPERIMENTS

Baselines. We compare POEM with dense FlashAttention-2 (Dao, 2024) and four training-free sparse prefilling baselines under the same FlashAttention-2 backend: **MInference** (Jiang et al., 2024) (official sparse-head configuration), **FlexPrefill** (Lai et al., 2025) ($B = 128$, $\gamma = 0.9$, $\tau = 0.1$), **XAttention** (Xu et al., 2025) ($\tau = 0.9$), and **STEM** (Niu et al., 2026) ($B = 128$, $\mu = 0.7$, $\beta = 0.2$, $k_{\text{start}} = 0.2N_{\text{blk}}$ for 8–16K inputs and $0.1N_{\text{blk}}$ beyond 16K, with a minimum budget of 54 blocks). We use each baseline's released or recommended configuration without task-specific retuning. SpargeAttention (Zhang et al., 2025) is excluded because its online PV skipping falls outside this shared-backend block-selection comparison and its controlled evaluation scope.

Implementation details. We implement all experiments in PyTorch on an NVIDIA A100 GPU. After block selection, FlashAttention-2 computes attention over the retained blocks. Following XAttention and STEM, we retain local and sink blocks for stability: four local and four sink blocks for Llama-3.1-8B (Grattafiori et al., 2024), and one of each for Qwen3-8B/14B (Yang et al., 2025). In Stage 1, we estimate block attention mass from low-precision query and key vectors to select the candidate pool. In Stage 2, we compute output-relative scores within the candidate pool. We set the output-relative deviation coefficient to $\beta = 0.3$ for Qwen3-14B and Llama-3.1-8B, and to $\beta = 0.2$ for Qwen3-8B. For Qwen-family models, we use YaRN (Peng et al., 2024) to extend their context windows to 128K, enabling long-context evaluation. We set the overall mass target to $p = 0.9$ and use the Stage 1 threshold $p_1 = 0.94$, yielding $p_2 = p/p_1$ for Stage 2.

Table 3: LongBench and RULER results on three models. Best in bold, second best underlined.

Model	Method	LongBench							RULER						
		S-Doc	M-Doc	Summ.	FSL	Synth.	Code	**Avg.**	4K	8K	16K	32K	64K	128K	**Avg.**
Llama-3.1-8B	Dense	28.57	25.75	25.45	63.61	65.83	33.06	40.38	95.26	94.29	93.58	87.99	84.85	77.87	88.97
	MInference	28.38	25.44	25.28	63.31	65.12	31.16	<u>39.78</u>	95.43	94.25	93.44	87.56	83.21	71.51	87.57
	FlexPrefill	24.25	22.61	25.14	62.24	19.56	35.01	31.47	93.67	93.63	93.84	87.90	84.83	72.43	<u>87.71</u>
	XAttention	27.51	23.55	25.51	63.82	55.83	35.77	38.66	95.09	94.39	93.27	88.95	82.45	70.09	87.38
	STEM	27.20	24.33	25.41	62.56	64.32	32.74	39.43	95.27	94.50	93.12	88.21	82.94	70.33	87.40
	POEM	26.82	23.98	25.44	64.04	65.71	35.00	**40.16**	95.01	94.32	93.20	88.79	82.74	72.38	**87.74**
Qwen3-8B	Dense	14.48	13.12	22.14	60.59	66.51	12.54	31.56	95.18	92.51	89.81	86.78	79.55	75.29	86.52
	MInference	13.97	12.73	21.45	58.06	65.75	12.02	30.66	94.83	92.55	89.41	85.68	76.12	70.89	<u>84.91</u>
	FlexPrefill	12.87	12.11	21.47	61.44	42.87	14.64	27.57	90.94	87.85	87.45	85.12	78.94	73.98	84.05
	XAttention	13.36	12.77	21.38	58.72	63.54	14.42	30.70	94.56	90.80	87.92	84.54	78.02	71.70	84.59
	STEM	14.13	13.00	21.39	57.79	65.62	12.46	<u>30.73</u>	94.67	90.43	87.61	85.09	77.42	71.24	84.41
	POEM	13.65	13.12	21.62	58.56	62.97	15.08	**30.83**	94.27	91.32	88.11	85.30	77.92	72.71	**84.94**
Qwen3-14B	Dense	15.55	12.62	21.92	62.41	66.83	53.30	38.77	95.61	93.83	93.27	93.16	90.45	84.01	91.72
	MInference	15.50	12.51	21.76	62.04	65.58	51.40	38.13	95.34	93.28	91.16	90.11	87.44	80.46	<u>89.63</u>
	FlexPrefill	15.09	13.05	21.77	61.15	48.12	50.95	35.02	93.43	92.74	92.30	91.13	87.41	80.64	89.60
	XAttention	15.37	13.11	21.82	60.94	65.42	52.67	38.22	95.19	93.02	91.23	89.89	87.36	80.33	89.50
	STEM	15.35	12.82	21.71	60.86	66.92	51.77	<u>38.24</u>	95.78	92.85	91.08	89.92	87.22	80.01	89.48
	POEM	15.36	13.21	21.83	62.12	65.92	53.61	**38.68**	95.23	93.12	91.60	89.95	87.64	80.51	**89.68**

Comparison on LongBench and RULER. Table 3 compares POEM with dense attention and four training-free sparse baselines, including MInference, FlexPrefill, XAttention, and STEM, on LongBench and RULER, covering models from different families and at different scales: Llama-3.1-8B and Qwen3-8B/14B. On LongBench, POEM ranks first among all sparse methods on every model, narrowing the gap to dense attention to as little as 0.09 points on Qwen3-14B. On RULER, POEM achieves the best average score across all three evaluated models. Across the two benchmarks, POEM is the only sparse method that consistently matches or outperforms every baseline on all model configurations. We attribute this robustness to the output-relative block scoring and head-wise threshold calibration, which together better reconstruct the dense attention output.

Effect of output-relative reranking. To examine whether output deviation provides a selection signal beyond attention mass, we conduct a block-level diagnostic using exact quantities computed with the dense output o_i. Figure 3 (a) shows the distribution of exact block attention mass m_{ib}, which is concentrated near zero and already provides strong discrimination between blocks. Figure 3 (b) plots the output deviation $\|\bar{v}_{ib} - o_i\|_2$, which is divided into two distinct modes. The low-

deviation mode corresponds to blocks whose values already lie close to the dense output, whereas the high-deviation mode contains blocks that would shift the output substantially if removed. Figure 3 (c) displays the importance factor $I_{ib} = m_{ib}\|\bar{v}_{ib} - o_i\|_2$, which POEM's practical score in Eq. 6 approximates. Incorporating the deviation reshapes the distribution relative to mass alone: blocks that rank high by mass but contribute values close to the output are lowered, while blocks with moderate mass but large deviation gain relative importance. As Figure 3 (d) shows, this change of distribution alters the selected block set in more than 74% of all heads, indicating that output deviation provides an important complementary signal for reliable block selection.

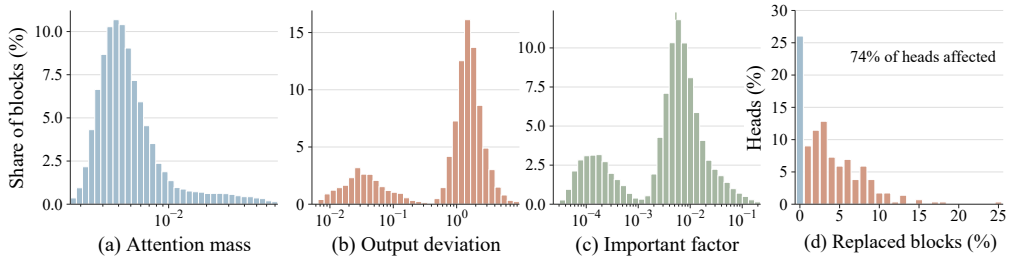

Figure 3: (a) Distribution of exact block attention mass m_{ib}. (b) Distribution of output deviation $\|\bar{v}_{ib} - o_i\|_2$. (c) Distribution of exact importance $I_{ib} = m_{ib}\|\bar{v}_{ib} - o_i\|_2$. (d) Percentage of blocks replaced per head after output-relative reranking, 74% of heads are affected.

Module ablation. Table 4 separately ablates covariance correction and the output reference on Qwen3-8B. Removing either component worsens perplexity, confirming that covariance improves block-output estimation, while the output reference enables output-relative deviation. Table 5 presents the module ablation on RULER at 128K context for Llama-3.1-8B, where we gradually add each component of POEM to a baseline that selects blocks by mass alone. Incorporating the output deviation into the block score yields a broad improvement, lifting the average from 69.94 to 72.14 at almost the same budget (19.50% vs. 19.56%). Further adding threshold calibration gives this score a slight increase (72.38) while reducing the density from 19.56% to 18.53%, confirming that calibration can leverage differences in head sensitivity.

Table 4: PG19 ablation at 32k with 0.1 density on Qwen3-8B.

Variant	Word PPL	Δ PPL
POEM	43.65	+0.00
w/o cov.	44.01	+0.36
w/o output	43.95	+0.30

Table 5: POEM module ablation on RULER at 128K. Best results are in bold, and second-best results are underlined.

Method	Retrieval								MHT	Agg.		QA		Overall	Budget
	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.	
Baseline	**100.00**	**99.00**	**100.00**	92.00	62.00	43.00	93.75	97.75	68.20	0.10	73.00	44.42	36.00	69.94	19.50%
+ output deviation	**100.00**	**99.00**	**100.00**	93.00	69.00	**48.00**	**95.75**	98.25	71.00	0.10	**73.67**	52.08	**38.00**	72.14	19.56%
+ threshold calibration	**100.00**	98.00	**100.00**	**94.00**	**70.00**	46.00	**95.75**	**98.50**	**74.80**	**0.20**	73.00	**54.75**	36.00	**72.38**	**18.53%**

Ablation studies. We conduct ablation studies on the output-relative coefficient β, total threshold p, stage 1 threshold p_1, and different scoring proxies on LongBench, RULER, and PG19 (Rae et al., 2019). Figures 4 (b)–(d) use Llama 3.1-8B. Figure 4 (a) shows that performance peaks at $\beta = 0.3$ for Qwen 3-14B and Llama 3-8B, and at 0.2 for Qwen 3-8B. Larger values degrade performance, suggesting that moderate deviation weighting improves selection but excessive weighting introduces noise from the attention mass. Figure 4 (b) plots the gain against added latency as p increases on RULER. The gain increases from $p = 0.8$ to $p = 0.9$, but drops at $p = 0.95$ despite the higher latency, showing that $p = 0.9$ provides the best trade-off among the evaluated settings. Figure 4 (c) varies the Stage-1 threshold p_1 under $p = 0.9$. PG19 perplexity decreases as p_1 grows from 0.90 to 0.94, since a larger candidate pool gives the output-relative proxy more room to improve upon mass-only selection. Beyond 0.94, however, the coarse attention score used in Stage-1 admit increasingly unreliable blocks into the pool, and because the final retention threshold p is held fixed,

these blocks displace trustworthy candidates and thus worsen the perplexity. Figure 4 (d) confirms our output-relative proxy's consistent superiority over mass-only and STEM scoring at matched PG19 densities. Details for all ablations are provided in Appendix C.3.

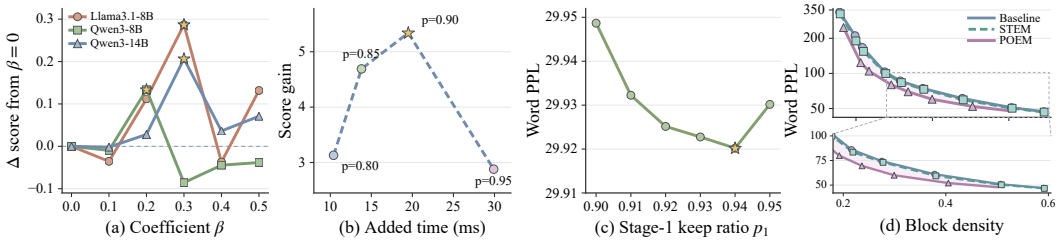

Figure 4: POEM ablations: (a) output-relative weight. (b) score–time trade-off. (c) Stage-1 mass threshold. (d) PG19 perplexity vs density for different scoring proxies.

Total efficiency analysis. Figure 5 compares the total attention run time of five state-of-the-art methods against the FlashAttention-2 baseline, from 16k to 128k, on Qwen3-8B. In these methods, POEM maintains fastest in all the length. At the length of 128k tokens, POEM even accelerates Flashattention-2 by 6.5×, from 928.0 ms to 143.9 ms, saving a massive amount of time.

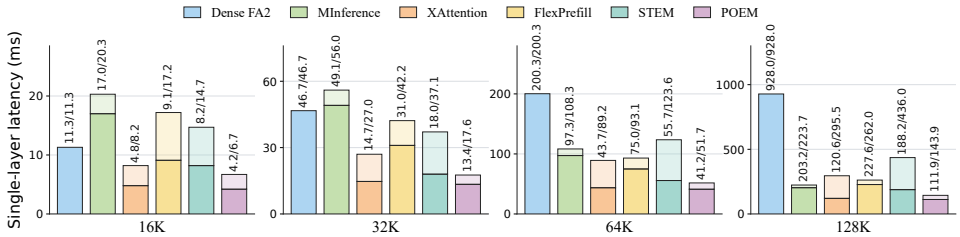

Figure 5: Attention latency (kernel/total). Dark/light segments show kernel/selector time. POEM is fastest throughout, reaching 6.5× speedup over dense FA2 at 128K.

Latency breakdown of POEM. Furthermore, we analysis the detailed attention execution composition of POEM in Figure 6. The results show that the two-stage selection process of POEM is modest. As length increasing, the selection process consistently accounts for only a small part of the total time of attention execution. The attention kernel time takes the majority, accounting for around 80% of the total times when lengths exceeds 64k. We conduct this study on Qwen3-8B.

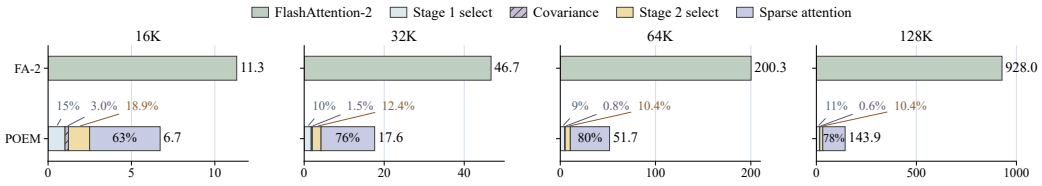

Figure 6: Latency breakdown of POEM. Selection overhead remains modest across context lengths, while sparse attention dominates the execution time at long contexts.

6 Conclusion

We showed that the output error of sparse attention grows not only with the attention mass of the removed blocks but also with their output deviation from the dense attention output. POEM therefore scores blocks by both quantities and achieves the best accuracy among sparse attention methods with up to a 6.5× attention speedup over FlashAttention-2 at 128K tokens. Future work could estimate block outputs more accurately before attention and extend the output-relative criterion to KV cache eviction during decoding and to trainable sparse attention.

AI USE STATEMENT

Generative AI tools were used to assist with drafting and language editing of this manuscript, including improvements to clarity, grammar, and organization. They were not treated as sources of scientific evidence: technical statements, citations, equations, and reported results were checked by the authors against the cited literature, mathematical derivations, implementation, and experimental records. All AI-assisted text was reviewed and revised by the authors. The authors take full responsibility for the final content of the paper, including all text, claims, and artifacts produced with the assistance of generative AI.

ETHICS STATEMENT

This work studies the efficiency of long-context language-model inference using publicly available pretrained models and established benchmark datasets; it does not involve human participants, newly collected personal data, or interaction with deployed users. Because POEM approximates dense attention, it can change model outputs in ways that are not fully captured by aggregate benchmark scores, potentially increasing factual errors or amplifying biases already present in the underlying models and data. We therefore do not recommend deploying the method in safety-critical or high-stakes settings without application-specific evaluation and safeguards. Although sparse attention may reduce the memory and computation required for inference, the experiments themselves consume GPU resources. Users of the method remain responsible for complying with the licenses and usage conditions of the underlying models and datasets.

REPRODUCIBILITY STATEMENT

Section 4 specifies the two-stage POEM algorithm, its output-relative objective, selection metric, and hyperparameters. The models, datasets, baselines, hardware, attention implementation, evaluation framework, and shared evaluation settings are documented in Section 5 and Appendix C.4. Appendix A provides the detailed derivation of the sparse-attention output error and the pooled approximation used by POEM. The experiments use publicly available model checkpoints and benchmarks, and all methods are evaluated using the same examples and decoding settings. To support independent reproduction, the implementation, evaluation commands, configuration files, software versions, and random seeds will be included in the anonymized supplementary material.

REFERENCES

Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 4895–4901, 2023.

Yongqi An, Chang Lu, Kuan Zhu, Tao Yu, Chaoyang Zhao, Hong Wu, Ming Tang, and Jinqiao Wang. Rest-kv: Robust kv cache eviction with layer-wise output reconstruction and spatial-temporal smoothing. *arXiv preprint arXiv:2605.08840*, 2026.

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

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Zefan Cai, Wen Xiao, Hanshi Sun, cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Animashree Anandkumar, Abedelkadir Asi, and Junjie Hu. R-kv: Redundancy-aware kv cache compression for reasoning models. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 60980–61005. Curran Associates, Inc., 2025. doi: 10.52202/085713-2038.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations*, volume 2024, pp. 35549–35562, 2024.

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

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024a.

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

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

Raghavv Goel, Junyoung Park, Mukul Gagrani, Dalton Jones, Matthew Morse, Harper Langston, Mingu Lee, and Chris Lott. Caote: Kv cache selection for llms via attention output error-based token eviction. *arXiv preprint arXiv:2504.14051*, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated deltanet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026.

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

Xiang Hu, Xinyu Wei, Hao Gu, Minshen Zhang, Tian Liang, Huayang Li, Lei Zhu, Yan Wang, Sirui Han, Yushi Bai, et al. Hierarchical sparse attention done right: Toward infinite context modeling. *arXiv preprint arXiv:2607.02980*, 2026.

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

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

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

Hoang Anh Duy Le, Sahil Joshi, Zeyu Yang, Zhaozhuo Xu, and Anshumali Shrivastava. Scout before you attend: Sketch-and-walk sparse attention for efficient llm inference. *arXiv preprint arXiv:2602.07397*, 2026.

Zhiwei Li, Lei Zhu, Hao Gu, Xiang Hu, Yan Wang, Haitao Mi, Sirui Han, Leo Liang, and Zhijiang Guo. Sas: Simple attention sparsification via end-to-end optimization of context ranking. *arXiv preprint arXiv:2609.13141*, 2026.

Chaofan Lin, Jiaming Tang, Shuo Yang, Hanshuo Wang, Tian Tang, Boyu Tian, Ion Stoica, Song Han, and Mingyu Gao. Twilight: Adaptive attention sparsity with hierarchical top-p pruning. *Advances in Neural Information Processing Systems*, 38:119406–119432, 2026.

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

Siran Liu, Zane Cao, and Yongchao He. A unified sparse attention via multi-granularity compression. *arXiv preprint arXiv:2512.14082*, 2025b.

Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, et al. Moba: Mixture of block attention for long-context llms. *Advances in Neural Information Processing Systems*, 38:17790–17815, 2026.

Amirkeivan Mohtashami and Martin Jaggi. Landmark attention: Random-access infinite context length for transformers. *arXiv preprint arXiv:2305.16300*, 2023.

Lin Niu, Xin Luo, Linchuan Xie, Yifu Sun, Guanghua Yu, Jianchen Zhu, and S Kevin Zhou. Stem: Rethinking causal information flow in sparse attention. *arXiv preprint arXiv:2603.06274*, 2026.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *International Conference on Learning Representations*, volume 2024, pp. 31932–31951, 2024.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, and Timothy P Lillicrap. Compressive transformers for long-range sequence modelling. *arXiv preprint arXiv:1911.05507*, 2019.

Isaac Rehg. Kv-compress: Paged kv-cache compression with variable compression rates per attention head. *arXiv preprint arXiv:2410.00161*, 2024.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

Yutao Sun, Zhenyu Li, Yike Zhang, Tengyu Pan, Bowen Dong, Yuyi Guo, and Jianyong Wang. Efficient attention mechanisms for large language models: A survey. *arXiv preprint arXiv:2507.19595*, 2025.

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

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023.

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

Xinghao Wang, Pengyu Wang, Dong Zhang, Chenkun Tan, Shaojun Zhou, Zhaoxiang Liu, Shiguo Lian, Fangxu Liu, Kai Song, and Xipeng Qiu. Sparser block-sparse attention via token permutation, 2026b.

Yixuan Wang, Huang He, Siqi Bao, Hua Wu, Haifeng Wang, Qingfu Zhu, and Wanxiang Che. Proxyattn: Guided sparse attention via representative heads, 2026c.

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

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

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

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

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin gödel machine: open-ended evolution of self-improving agents. In *International Conference on Learning Representations*, volume 2026, pp. 104223–104294, 2026.

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

Weilin Zhao, Zihan Zhou, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, et al. Infllm-v2: Dense-sparse switchable attention for seamless short-to-long adaptation. In *International Conference on Learning Representations*, volume 2026, pp. 69374–69390, 2026.

A Theoretical Analysis and Proofs

This section separates the theoretical steps in POEM's motivation: a local upper bound relating predictive divergence to layer-output error, an exact identity for sparse-attention output error, and the first-order approximation that makes block scoring tractable. Algorithmic conventions, experimental protocols, and efficiency accounting are deferred to later sections so that each derivation states only the claim, assumptions, proof, and scope that it needs. Throughout the appendices, bold uppercase letters denote matrices, bold lowercase letters denote vectors, and plain lowercase letters denote scalars. Calligraphic letters denote sets or structured collections, and operator names retain their conventional notation. An index identifying a block, head, or token does not remove boldface when the indexed object is still a matrix or vector; scalar coordinates are written in plain type.

A.1 Derivation of the Prediction-to-Layer-Output Bound

The objective in Eq. 7 is defined on the final prediction distribution, whereas sparse block selection directly changes an intermediate attention output. This derivation connects these two quantities. We first propagate a layer-output perturbation to the final logits and then expand the predictive KL divergence around the dense logits. The resulting upper bound relates final prediction error to layer-output reconstruction error, providing the first step of the hierarchical decomposition in Section 4.2.

Proposition (prediction-to-layer-output bound). For a fixed query block i, suppose only layer ℓ is perturbed and the downstream map from its projected attention output to the logits is locally twice differentiable. Then, in a neighborhood of the dense computation path, the predictive KL divergence satisfies Eq. A.12; in particular, its leading term is at most quadratic in the Frobenius norm of the layer-output perturbation.

Assumptions. We apply sparse attention only at layer ℓ and keep all other layers dense. The downstream logit map is twice continuously differentiable, with bounded first and second derivatives near the dense output, and the perturbation is small enough for the Taylor expansions below. For query block i, define the attention-output perturbation and its vectorization as

$$\Delta\mathbf{O}_{\ell,i} = \widetilde{\mathbf{O}}_{\ell,i} - \mathbf{O}_{\ell,i}, \qquad \boldsymbol{\varepsilon}_{\ell,i} = \mathrm{vec}(\Delta\mathbf{O}_{\ell,i}),$$

so that

$$\|\boldsymbol{\varepsilon}_{\ell,i}\|_2 = \|\Delta\mathbf{O}_{\ell,i}\|_F.$$

For compactness, write $\mathbf{x}_{\ell,i} = \mathrm{vec}(\mathbf{O}_{\ell,i})$ and let $f_{\ell\to z}$ denote the downstream network from this vectorized attention output to the final logits. Thus,

$$\mathbf{z}_i^d = f_{\ell\to z}(\mathbf{x}_{\ell,i}), \qquad \mathbf{z}_i^s = f_{\ell\to z}(\mathbf{x}_{\ell,i} + \boldsymbol{\varepsilon}_{\ell,i}), \qquad \Delta\mathbf{z}_i = \mathbf{z}_i^s - \mathbf{z}_i^d.$$

Proof. *From output perturbations to logits.* Writing z_{ic} for coordinate c of $\mathbf{z}_i$, Taylor's theorem gives

$$\Delta z_{ic} = \nabla f_{\ell\to z,c}(\mathbf{x}_{\ell,i})^\top \boldsymbol{\varepsilon}_{\ell,i} + \frac{1}{2}\boldsymbol{\varepsilon}_{\ell,i}^\top \mathbf{H}_{\ell\to z,c}(\mathbf{x}_{\ell,i} + \theta_c\boldsymbol{\varepsilon}_{\ell,i})\boldsymbol{\varepsilon}_{\ell,i}, \qquad \theta_c \in (0,1), \tag{A.1}$$

where $\mathbf{H}_{\ell\to z,c}$ is the Hessian of the cth logit with respect to the vectorized attention output. If these Hessians are bounded in a neighborhood of the dense path, stacking Eq. A.1 over all logits gives

$$\Delta\mathbf{z}_i = \mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i} + \mathbf{r}_{\ell,i}, \qquad \|\mathbf{r}_{\ell,i}\|_2 = \mathcal{O}\big(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2^2\big), \tag{A.2}$$

where $\mathbf{J}_{\ell\to z,i}$ is the Jacobian of $f_{\ell\to z}$ evaluated on the dense computation path. For notational convenience, the Jacobian is taken with respect to the vectorized attention output.

The local KL expansion. Let $\mathbf{z}_i^d$ be the dense logits and define

$$\mathbf{p}_i^d = \mathrm{softmax}(\mathbf{z}_i^d), \qquad \mathbf{p}_i^s = \mathrm{softmax}(\mathbf{z}_i^d + \Delta\mathbf{z}_i), \qquad \Delta l_i^{\mathrm{pred}} = d_{\mathrm{KL}}(\mathbf{p}_i^d \,\|\, \mathbf{p}_i^s).$$

To calculate this KL divergence explicitly, define the log-partition function

$$a(\mathbf{z}) = \log\left(\sum_c \exp(z_c)\right). \tag{A.3}$$

Because $\log[\mathrm{softmax}(\mathbf{z})]_c = z_c - a(\mathbf{z})$, we have

$$\begin{aligned}\Delta l_i^{\mathrm{pred}} &= \sum_c p_{ic}^d \left[\log p_{ic}^d - \log p_{ic}^s\right] \\ &= \sum_c p_{ic}^d \left[-\Delta z_{ic} + a(\mathbf{z}_i^d + \Delta\mathbf{z}_i) - a(\mathbf{z}_i^d)\right] \\ &= a(\mathbf{z}_i^d + \Delta\mathbf{z}_i) - a(\mathbf{z}_i^d) - (\mathbf{p}_i^d)^\top \Delta\mathbf{z}_i, \end{aligned} \tag{A.4}$$

where the last equality uses $\sum_c p_{ic}^d = 1$. Differentiating the same log-partition expression once and then once more keeps the calculation in one chain:

$$\begin{aligned} \frac{\partial a(\mathbf{z})}{\partial z_c} &= \frac{\exp(z_c)}{\sum_j \exp(z_j)} = p_c, & \frac{\partial^2 a(\mathbf{z})}{\partial z_c \partial z_d} &= p_c\big(\mathbb{I}[c=d] - p_d\big), \\ \nabla a(\mathbf{z}_i^d) &= \mathbf{p}_i^d, & \nabla^2 a(\mathbf{z}_i^d) &= \mathrm{Diag}(\mathbf{p}_i^d) - \mathbf{p}_i^d(\mathbf{p}_i^d)^\top = \mathbf{F}_i. \end{aligned} \tag{A.5}$$

Expanding the log-partition function in Eq. A.4 therefore gives

$$a(\mathbf{z}_i^d + \Delta\mathbf{z}_i) = a(\mathbf{z}_i^d) + (\mathbf{p}_i^d)^\top \Delta\mathbf{z}_i + \frac{1}{2}\Delta\mathbf{z}_i^\top \mathbf{F}_i \Delta\mathbf{z}_i + \mathcal{O}\big(\|\Delta\mathbf{z}_i\|_2^3\big)\,. \tag{A.6}$$

The constant terms cancel in Eq. A.4, and its linear term cancels the explicit term $-(\mathbf{p}_i^d)^\top \Delta\mathbf{z}_i$. Hence,

$$\Delta l_i^{\mathrm{pred}} = \frac{1}{2}\Delta\mathbf{z}_i^\top \mathbf{F}_i \Delta\mathbf{z}_i + \mathcal{O}\big(\|\Delta\mathbf{z}_i\|_2^3\big)\,, \tag{A.7}$$

This calculation also shows directly why the first-order term vanishes and why the quadratic term is governed by the softmax Fisher matrix $\mathbf{F}_i$.

Bounding the quadratic term. Substituting Eq. A.2 into Eq. A.7, the quadratic term can be expanded as

$$\begin{aligned}\frac{1}{2}\Delta\mathbf{z}_i^\top \mathbf{F}_i \Delta\mathbf{z}_i &= \frac{1}{2}(\mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i} + \mathbf{r}_{\ell,i})^\top \mathbf{F}_i (\mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i} + \mathbf{r}_{\ell,i}) \\ &= \frac{1}{2}\boldsymbol{\varepsilon}_{\ell,i}^\top \mathbf{J}_{\ell\to z,i}^\top \mathbf{F}_i \mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i} + \boldsymbol{\varepsilon}_{\ell,i}^\top \mathbf{J}_{\ell\to z,i}^\top \mathbf{F}_i \mathbf{r}_{\ell,i} + \frac{1}{2}\mathbf{r}_{\ell,i}^\top \mathbf{F}_i \mathbf{r}_{\ell,i}. \end{aligned} \tag{A.8}$$

Locally, $\mathbf{J}_{\ell\to z,i}$ and $\mathbf{F}_i$ are bounded, while $\|\mathbf{r}_{\ell,i}\|_2 = \mathcal{O}(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2^2)$. The second and third terms in Eq. A.8 are therefore respectively $\mathcal{O}(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2^3)$ and $\mathcal{O}(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2^4)$. Moreover, $\|\Delta\mathbf{z}_i\|_2 = \mathcal{O}(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2)$, so the remainder in Eq. A.7 is also $\mathcal{O}(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2^3)$. Collecting these terms yields

$$\Delta l_i^{\mathrm{pred}} = \frac{1}{2}\boldsymbol{\varepsilon}_{\ell,i}^\top \mathbf{J}_{\ell\to z,i}^\top \mathbf{F}_i \mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i} + \mathcal{O}\big(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2^3\big)\,. \tag{A.9}$$

This expression connects the layer-output perturbation to the final predictive divergence through the downstream Jacobian and the softmax Fisher matrix. Assume locally that

$$\|\mathbf{J}_{\ell\to z,i}\|_2 \le l_{\ell,i}, \qquad \|\mathbf{F}_i\|_2 \le \kappa_i.$$

Since $\mathbf{F}_i$ is positive semidefinite, the leading quadratic term satisfies

$$\begin{aligned}\boldsymbol{\varepsilon}_{\ell,i}^\top \mathbf{J}_{\ell\to z,i}^\top \mathbf{F}_i \mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i} &= (\mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i})^\top \mathbf{F}_i (\mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i}) \\ &\le \|\mathbf{F}_i\|_2 \|\mathbf{J}_{\ell\to z,i}\boldsymbol{\varepsilon}_{\ell,i}\|_2^2 \\ &\le \|\mathbf{F}_i\|_2 \|\mathbf{J}_{\ell\to z,i}\|_2^2 \|\boldsymbol{\varepsilon}_{\ell,i}\|_2^2 \\ &\le \kappa_i l_{\ell,i}^2 \|\boldsymbol{\varepsilon}_{\ell,i}\|_2^2. \end{aligned} \tag{A.10}$$

Thus,

$$\Delta l_i^{\mathrm{pred}} \le \frac{1}{2}\kappa_i l_{\ell,i}^2 \|\boldsymbol{\varepsilon}_{\ell,i}\|_2^2 + \mathcal{O}\big(\|\boldsymbol{\varepsilon}_{\ell,i}\|_2^3\big)\,. \tag{A.11}$$

Defining $c_{\ell,i} = \frac{1}{2}\kappa_i l_{\ell,i}^2$ and using $\|\boldsymbol{\varepsilon}_{\ell,i}\|_2 = \|\widetilde{\mathbf{O}}_{\ell,i} - \mathbf{O}_{\ell,i}\|_F$, we obtain

$$\Delta l_i^{\mathrm{pred}} \le c_{\ell,i} \left\|\widetilde{\mathbf{O}}_{\ell,i} - \mathbf{O}_{\ell,i}\right\|_F^2 + \mathcal{O}\left(\left\|\widetilde{\mathbf{O}}_{\ell,i} - \mathbf{O}_{\ell,i}\right\|_F^3\right). \tag{A.12}$$

From the layer bound to head errors. Let H be the number of heads and $\Delta\mathbf{O}_{\ell,i,h}$ the matrix of head-h output perturbations over query block i. The projected perturbation satisfies $\Delta\mathbf{O}_{\ell,i} = \sum_{h=1}^{H} \Delta\mathbf{O}_{\ell,i,h}\mathbf{W}_O^{(h)}$. Submultiplicativity and the triangle inequality first give

$$\|\Delta\mathbf{O}_{\ell,i}\|_F \le \sum_h \|\mathbf{W}_O^{(h)}\|_2 \|\Delta\mathbf{O}_{\ell,i,h}\|_F.$$

Applying Cauchy–Schwarz to this sum then gives

$$\|\Delta\mathbf{O}_{\ell,i}\|_F^2 \le H \sum_h \|\mathbf{W}_O^{(h)}\|_2^2 \|\Delta\mathbf{O}_{\ell,i,h}\|_F^2. \tag{A.13}$$

Substituting this inequality into the leading term of Eq. A.12 yields the weighted head-error bound in Section 4.2. Together, the two bounds link the final predictive divergence to the reconstruction errors of individual attention heads and show where head-dependent weights enter the decomposition.

A.2 Exact Sparse-Attention Output Error

We now work within one attention head and retain the original token-level causal mask inside each block. The main text writes a single query row with scalar mass m_{ib}, conditional block output $\bar{\mathbf{v}}_{ib}$, and dense output $\mathbf{o}_i$. Here a query block contains several such rows: $\mathbf{m}_{ib}$ stacks their masses, $\mathbf{D}_{ib} = \mathrm{Diag}(\mathbf{m}_{ib})$ applies them rowwise, $\mathbf{W}_{ib}$ stacks the corresponding conditional block outputs, and $\mathbf{O}_i^{\mathrm{dense}}$ stacks the dense outputs. Thus each row of Eq. A.18 reduces to the scalar-query identity used in the main text; the matrix notation introduces no different objective. For query block i, let $\mathbf{A}_{ib}$ denote the dense attention-score submatrix associated with key block b. Define

$$\mathbf{m}_{ib} = \mathbf{A}_{ib}\mathbf{1}, \qquad \mathbf{D}_{ib} = \mathrm{Diag}(\mathbf{m}_{ib}), \qquad \mathbf{W}_{ib} = \mathbf{D}_{ib}^{-1}\mathbf{A}_{ib}\mathbf{V}_b. \tag{A.14}$$

Here, $\mathbf{m}_{ib}$ collects the scalar block masses of all query rows in query block i. For a zero-mass row, we define the corresponding row of $\mathbf{W}_{ib}$ as zero rather than inverting a zero diagonal entry. Such a row contributes nothing to $\mathbf{D}_{ib}\mathbf{W}_{ib}$. Since each dense attention row sums to one, its output decomposes as

$$\mathbf{O}_i^{\mathrm{dense}} = \sum_{b\le i} \mathbf{A}_{ib}\mathbf{V}_b = \sum_{b\le i} \mathbf{D}_{ib}\mathbf{W}_{ib}. \tag{A.15}$$

Let $\mathcal{T}_i$ and $\mathcal{R}_i$ be the retained and removed block sets, respectively, and define

$$\mathbf{D}_{i\mathcal{R}} = \sum_{b\in\mathcal{R}_i} \mathbf{D}_{ib}. \tag{A.16}$$

Recomputing softmax over the retained tokens divides each query row by its retained probability mass. Therefore,

$$\mathbf{O}_i^{\mathrm{sparse}}(\mathcal{T}_i) = (\mathbf{I} - \mathbf{D}_{i\mathcal{R}})^{-1} \sum_{b\in\mathcal{T}_i} \mathbf{D}_{ib}\mathbf{W}_{ib}. \tag{A.17}$$

Substituting Eq. A.15 gives the exact output error

$$\begin{aligned}\mathbf{E}_i(\mathcal{T}_i) &= \mathbf{O}_i^{\mathrm{dense}} - \mathbf{O}_i^{\mathrm{sparse}}(\mathcal{T}_i) \\ &= (\mathbf{I} - \mathbf{D}_{i\mathcal{R}})^{-1} \sum_{b\in\mathcal{R}_i} \mathbf{D}_{ib}\left(\mathbf{W}_{ib} - \mathbf{O}_i^{\mathrm{dense}}\right).\end{aligned} \tag{A.18}$$

The single-token form of Eq. A.18 has also been used to score cached tokens in KV-cache eviction (Goel et al., 2025; An et al., 2026). For multiple removed blocks, submultiplicativity and the triangle inequality give

$$\|\mathbf{E}_i(\mathcal{T}_i)\|_F \le \left\|(\mathbf{I} - \mathbf{D}_{i\mathcal{R}})^{-1}\right\|_2 \sum_{b\in\mathcal{R}_i} \left\|\mathbf{D}_{ib}\left(\mathbf{W}_{ib} - \mathbf{O}_i^{\mathrm{dense}}\right)\right\|_F. \tag{A.19}$$

Eq. A.19 shows that each removed block affects the output error through both its attention mass and output deviation. We therefore rank blocks by the mass-weighted output deviation $m_{ib}\|\bar{\mathbf{v}}_{ib} - \mathbf{o}_i\|_2$, retaining blocks with larger estimated effects on the attention output.

A.3 Approximation for Block Output

POEM approximates each block output by applying a first-order Taylor expansion at the pooled query. The corresponding diagnostic and complexity analysis appear in Sections D and E.

The derivation below follows the row-vector convention of the main text. Accordingly, its QK logit is $\mathbf{q}\mathbf{k}^\top$ and its covariance correction is $\mathbf{q}\mathbf{C}$.

A first-order block-output approximation. For a fully visible key block $\mathcal{B}_b$ of size n_b, define

$$\bar{\mathbf{k}}_b = \frac{1}{n_b}\sum_{t\in\mathcal{B}_b}\mathbf{k}_t, \qquad \bar{\mathbf{v}}_b = \frac{1}{n_b}\sum_{t\in\mathcal{B}_b}\mathbf{v}_t, \qquad \mathbf{C}_b = \frac{1}{n_b}\sum_{t\in\mathcal{B}_b}(\mathbf{k}_t-\bar{\mathbf{k}}_b)^\top(\mathbf{v}_t-\bar{\mathbf{v}}_b). \tag{A.20}$$

For the pooled query $\bar{\mathbf{q}}_i$, let $a_t = \bar{\mathbf{q}}_i(\mathbf{k}_t-\bar{\mathbf{k}}_b)^\top/\sqrt{d_h}$. Subtracting the common block-mean logit does not change the softmax, and the centered logits satisfy $\sum_t a_t = 0$. Introduce

$$\alpha_t(u) = \frac{\exp(ua_t)}{\sum_{r\in\mathcal{B}_b}\exp(ua_r)}, \qquad \mathbf{g}(u) = \sum_{t\in\mathcal{B}_b}\alpha_t(u)\mathbf{v}_t. \tag{A.21}$$

The parameter u interpolates between uniform weights at $u=0$ and the within-block softmax at $u=1$. At $u=0$, $\alpha_t(0)=1/n_b$ and $\mathbf{g}(0)=\bar{\mathbf{v}}_b$. Moreover,

$$\alpha_t'(u) = \alpha_t(u)\left(a_t - \sum_{r\in\mathcal{B}_b}\alpha_r(u)a_r\right), \tag{A.22}$$

$$\alpha_t'(0) = \frac{1}{n_b}\left(a_t - \frac{1}{n_b}\sum_r a_r\right) = \frac{a_t}{n_b}. \tag{A.23}$$

Therefore,

$$\mathbf{g}'(0) = \frac{1}{n_b}\sum_t a_t\mathbf{v}_t \tag{A.24}$$

$$= \frac{\bar{\mathbf{q}}_i}{\sqrt{d_h}}\left[\frac{1}{n_b}\sum_t(\mathbf{k}_t-\bar{\mathbf{k}}_b)^\top(\mathbf{v}_t-\bar{\mathbf{v}}_b)\right] = \frac{\bar{\mathbf{q}}_i}{\sqrt{d_h}}\mathbf{C}_b. \tag{A.25}$$

The first-order Taylor expansion $\mathbf{g}(1)\approx\mathbf{g}(0)+\mathbf{g}'(0)$ gives

$$\mathbf{g}(1) \approx \bar{\mathbf{v}}_b + \frac{\bar{\mathbf{q}}_i}{\sqrt{d_h}}\mathbf{C}_b =: \widehat{\mathbf{v}}_{ib}. \tag{A.26}$$

The omitted terms are second order in the centered logits a_t. When their within-block spread is moderate, these terms are typically small and are therefore omitted; Section D.1 evaluates the resulting approximation directly.

All configured local blocks, including the diagonal query–key block, are mandatory and are retained without using this approximation for selection. The pooled-query estimate is applied only to the remaining optional key blocks, which precede the local window and are fully visible to every token in query block i. For these optional blocks, all tokens in query block i use the mean query $\bar{\mathbf{q}}_i$ during scoring instead of their individual queries. After selection, the final sparse-attention kernel still applies the exact token-level causal mask. The diagnostic in Section D.1 measures the effects of query pooling and the first-order truncation.

B Algorithm and Implementation Details

This section connects the selection formula to the exact operations used by Algorithm 1.

B.1 From Block Estimates to the Selection Score

A coarse global-output reference. The pooled QK term in Eq. 6 is $z_{ib}^{\mathrm{pool}} = \bar{\mathbf{q}}_i\bar{\mathbf{k}}_b^\top/\sqrt{d_h}$. Following the main text, the implementation approximates this term with the antidiagonal scorer; denote its

output by $\widehat{z}_{ib}^{\mathrm{AD}} \approx z_{ib}^{\mathrm{pool}}$. All optional blocks evaluated by the scorer are fully visible, fixed-size key blocks; boundary and local blocks are mandatory and do not participate in the optional-block ranking. Consequently, the block cardinality is constant over the scored blocks and no block-dependent cardinality term is needed. Normalizing the antidiagonal estimates over visible blocks gives the pooled weights a_{ib} used by Stage 1 and by the coarse output reference:

$$a_{ib} = \frac{\exp(\widehat{z}_{ib}^{\mathrm{AD}})}{\sum_{c \le i} \exp(\widehat{z}_{ic}^{\mathrm{AD}})}, \qquad \widehat{\mathbf{o}}_i = \sum_{b \le i} a_{ib} \bar{\mathbf{v}}_b, \qquad r_{ib}^{\mathrm{value}} = \|\widehat{\mathbf{v}}_{ib} - \widehat{\mathbf{o}}_i\|_2. \tag{B.1}$$

Here $\widehat{\mathbf{o}}_i$ aggregates all visible pooled values and contains no query-conditioned covariance correction; only the Stage 2 candidate block outputs include that correction.

Taking the logarithm of the separable importance proxy for positive masses gives $\log m_{ib} + \log \|\bar{\mathbf{v}}_{ib} - \mathbf{o}_i\|_2$. Substituting the pooled QK term for the first component and the approximated output deviation for the second, and introducing β yields the score in the main text. Using the antidiagonal approximation to the pooled QK term, the implementation evaluates

$$s_{ib} = \widehat{z}_{ib}^{\mathrm{AD}} + \beta \log r_{ib}^{\mathrm{value}}, \tag{B.2}$$

which is the operational form of Eq. 6 after replacing its pooled QK term by the antidiagonal estimate and substituting r_{ib}^{value} from Eq. B.1. Thus the implementation uses antidiagonal scoring to estimate the first term displayed in the main text; it does not replace that term by the logarithm of a separate Stage 1 mass estimator.

Ranking, mass accumulation, and mandatory blocks. Stage 1 selects the shortest prefix of blocks in descending a_{ib} order whose cumulative estimated mass reaches p_1 of the total visible mass, then takes the union with the mandatory sink/local set $\mathcal{M}_i$. Stage 2 initializes $\mathcal{S}_i = \mathcal{M}_i$, credits the masses of mandatory blocks toward the target, and evaluates and sorts by s_{ib} only the remaining optional candidates $\mathcal{C}_i \setminus \mathcal{M}_i$. It appends them until

$$\sum_{b \in \mathcal{S}_i} a_{ib} \ge p_{2,g(h)} \sum_{b \in \mathcal{C}_i} a_{ib}. \tag{B.3}$$

The combined score thus determines only the insertion order; the quantity accumulated against the threshold is the original pooled weight a_{ib}, not a renormalized softmax of the combined score. Mandatory blocks are never duplicated; if their mass alone meets the target, no optional block is added. Block discreteness may still cause overshoot. Note that this criterion covers pooled attention mass and does not guarantee coverage under exact dense attention. The final sparse attention is recomputed from the original token-resolution Q/K/V tensors.

C Experimental Setup

This section collects the model, dataset, and controlled-diagnostic protocols; later diagnostic sections refer back to these populations rather than mixing their setup with the theoretical derivations.

C.1 Reconstruction-Error Evaluation Setup

Table 2 compares block-scoring proxies on LongBench prompts, each containing exactly 8,192 tokens, using Qwen3-8B. All methods use the same antidiagonal QK scorer and the pooled weights obtained by normalizing its scores. The comparison varies only the selection score: pooled attention mass alone, a STEM-style value-magnitude score with $\beta = 0.2$, and POEM's output-relative score with $\beta = 0.2$. This controlled comparison is distinct from running each method's complete native implementation.

Fixed selection budget. Blocks contain 128 tokens. Each query/head retains at most 16 visible blocks, corresponding to 2,048 tokens; if fewer than 16 blocks are visible, all are retained. This is a fixed-count diagnostic, not the top-p protocol used for the main task results.

Layer and logit metrics. For the layer-5, layer-15, and layer-35 columns, only the specified decoder layer is sparsified and all other layers remain dense. We measure hidden-state MSE immediately after that layer over all $8,192 \times 4,096$ elements, then average the per-text MSE over the 30 paired texts. Layer indices are one-based. The "Final logits" column instead sparsifies all 36 decoder layers and measures raw-logit MSE after the final normalization and language-model head over all prompt positions and vocabulary entries. It is neither a KL divergence nor an isolated attention-head error.

C.2 Exact-Score Reranking Diagnostic

Figure 3 uses Llama-3.1-8B-Instruct on RULER examples. For every evaluated block, we compute the exact dense attention mass m_{ib}, conditional block output $\bar{\mathbf{v}}_{ib}$, dense head output $\mathbf{o}_i$, and exact importance $I_{ib} = m_{ib}\|\bar{\mathbf{v}}_{ib} - \mathbf{o}_i\|_2$. Panel (d) compares mass-only selection with exact output-relative reranking at the same retention setting and reports the fraction of evaluated heads for which the selected block set changes. This diagnostic is separate from the benchmark scores in Table 3.

C.3 Ablation Experimental Setup

The ablations vary one factor at a time while keeping the remaining model and selector settings fixed as described in the main text and the reproducibility configuration below.

Output-deviation weight. Figure 4(a) sweeps $\beta \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ for Llama-3.1-8B, Qwen3-8B, and Qwen3-14B, and reports the score change relative to $\beta = 0$ for each model.

Overall threshold and latency. Figure 4(b) uses Llama-3.1-8B on RULER and sweeps the overall retention threshold over five levels, $p \in \{0.75, 0.80, 0.85, 0.90, 0.95\}$. At each level, we record the RULER score and selection latency under the same evaluation setting. The plot reports the increment associated with raising p to the labeled level, rather than the absolute result at that level. Specifically, the point labeled $p = 0.80$ reports the score difference between $p = 0.80$ and $p = 0.75$, together with the corresponding latency difference; the points labeled 0.85, 0.90, and 0.95 are defined analogously relative to their immediately preceding levels. The five evaluated thresholds therefore produce four plotted transitions. The $p = 0.75$ result serves as the reference for the first transition and is not shown as a separate point.

Stage 1 threshold. Figure 4(c) uses Llama-3.1-8B on PG19, fixes the overall target at $p = 0.90$, and sweeps $p_1 \in \{0.90, 0.91, 0.92, 0.93, 0.94, 0.95\}$. All other settings are fixed, and each point reports word perplexity under the corresponding candidate-pool threshold.

Scoring proxy and density. Figure 4(d) uses Llama-3.1-8B on PG19 and compares mass-only, STEM, and POEM scoring at matched block densities. Each point reports word perplexity, with the inputs and density fixed across the three selection scores.

Component ablation. Table 4 evaluates Qwen3-8B on 20 books from the PG19 test split, using 32K-token inputs and a fixed attention density of approximately 0.1. The same books and density are used for POEM, the variant without covariance correction, and the variant without the output-relative reference. Word perplexity is computed over this fixed 20-book evaluation population.

C.4 Models, Datasets, and Evaluation Scope

Backbones and context lengths. We evaluate Llama-3.1-8B (Grattafiori et al., 2024) and Qwen3-8B/14B (Yang et al., 2025). Model parameters remain frozen. The long-context Qwen runs use YaRN (Peng et al., 2024) to extend the 32K context window to 128K tokens; this setting should not be inferred for every 8K diagnostic.

Datasets and reporting. LongBench (Bai et al., 2024) contains 21 tasks in six categories; we use each prescribed metric and report the unweighted macro average of category scores. RULER (Hsieh et al., 2024) is reported at 4K, 8K, 16K, 32K, 64K, and 128K and is averaged uniformly over lengths. PG19 (Rae et al., 2019) supplies long-form text for perplexity ablations and selected block

diagnostics. Task evaluation uses lm-evaluation-harness (Gao et al., 2024a); component diagnostics use the explicit samples, precision, thresholds, and aggregation rules stated with each diagnostic and are not pooled into the task benchmarks.

C.5 Reproducibility Configuration

All models are inference-only and retain their released weights. Experiments use PyTorch on an NVIDIA A100, 128-token query/key blocks, antidiagonal scoring with $B_1 = 16$ in Stage 1 and $B_2 = 8$ in Stage 2, and FlashAttention-2 for the final token-resolution attention. Llama-3.1-8B uses four sink and four local blocks; Qwen3-8B/14B use one of each. Stage 1 uses $p_1 = 0.94$. The output-deviation coefficient is 0.3 for Llama-3.1-8B and Qwen3-14B and 0.2 for Qwen3-8B. Before calibration, $p_2 = 0.90/0.94$; calibrated groups use $0.81/0.94$. The logarithm floor is 10^{-6}.

The diagnostic subsections additionally state model trajectory, activation precision, prompt length, sample count, layer indexing, candidate definition, and aggregation unit. A reproduction artifact should preserve the ordered example identifiers and tokenized lengths for every diagnostic and for the two calibration manifests in Section F; this prevents nominally identical datasets from silently producing different subsets after tokenization or filtering.

D Approximation and Preselection Diagnostics

We first test whether the two output estimates are faithful to their distinct targets, and then test the operational question for preselection: whether the mass-only candidate pool excludes blocks that an all-visible output-relative selector would retain. The block-output and global-output plots use different units and populations and are therefore complementary diagnostics, not a head-to-head comparison.

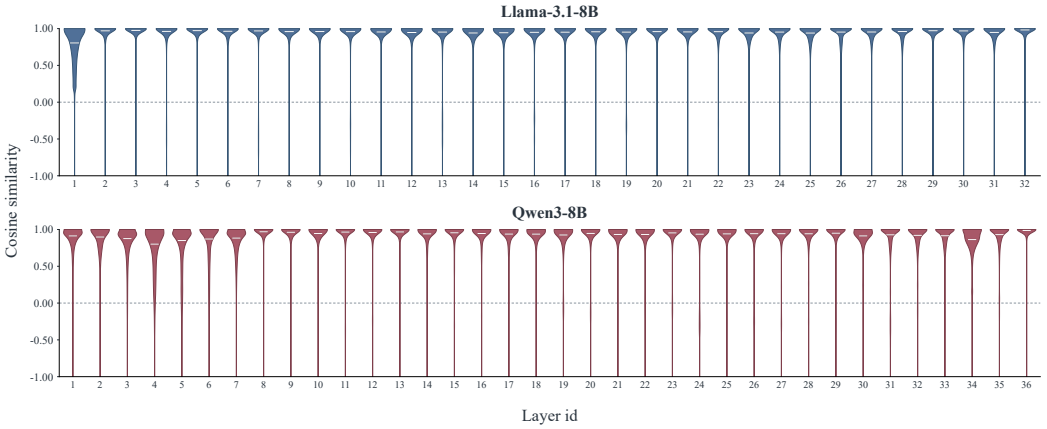

Figure 7: Cosine similarity between the first-order estimate $\widehat{\mathbf{v}}_{ib}$ and the exact token-level causal block output $\mathbf{v}_{ib}^{\text{exact}}$. White bars mark weighted medians.

D.1 Block-Output Approximation

Block-output diagnostic: reference and evaluation population. We use 30 shared LongBench text identities for Llama-3.1-8B and Qwen3-8B, tokenize them separately for each model, and retain 8,192 tokens without a chat template. Activations follow the dense BF16 model trajectory. All layers, 32 query heads, and 64 query blocks are evaluated. Blocks contain 128 tokens. Within each layer, every text/head/query-block row has equal total weight, divided equally among its evaluated key blocks. Layers are indexed from one in the figure.

For query block $\mathcal{Q}_i$ and key block $\mathcal{B}_b$, the reference is

$$\mathbf{v}_{ib}^{\text{exact}} = \frac{1}{|\mathcal{Q}_i|} \sum_{t \in \mathcal{Q}_i} \frac{\sum_{u \in \mathcal{B}_b,\, u \le t} \exp(\mathbf{q}_t \mathbf{k}_u^\top / \sqrt{d_h}) \mathbf{v}_u}{\sum_{u \in \mathcal{B}_b,\, u \le t} \exp(\mathbf{q}_t \mathbf{k}_u^\top / \sqrt{d_h})}. \tag{D.1}$$

We compare the reference with $\widehat{\mathbf{v}}_{ib} = \bar{\mathbf{v}}_b + \bar{\mathbf{q}}_i \mathbf{C}_b / \sqrt{d_h}$ using cosine similarity:

$$c_{ib} = \frac{\widehat{\mathbf{v}}_{ib} (\mathbf{v}_{ib}^{\text{exact}})^\top}{\|\widehat{\mathbf{v}}_{ib}\|_2 \|\mathbf{v}_{ib}^{\text{exact}}\|_2} \tag{D.2}$$

for each text, layer, head, and query–key block pair. Equal row weighting prevents query blocks with more evaluated key blocks from dominating the distribution.

Block-output cosine similarity. Figure 7 shows high cosine similarity for most evaluated blocks. The weighted median exceeds 0.9 in 31 of 32 Llama layers and 29 of 36 Qwen layers. The median is lower in Llama's first layer (0.805), several early Qwen layers, and Qwen layer 34. Thin lower tails also indicate poorly aligned pairs in otherwise well-aligned layers. These results support the first-order estimate as a useful block-level signal while identifying layer-dependent limitations. Cosine similarity measures direction, not vector-magnitude error; it does not by itself bound the error in the block-output norm or guarantee that approximate scores preserve the exact block ranking.

D.2 Accuracy of the Global Output Estimate

The global reference $\widehat{\mathbf{o}}_i$ anchors the output deviation used for block selection. To examine how much information this inexpensive estimate retains about the final attention output, we compare it with the dense attention output before the multi-head output projection.

Reference and aggregation. Figure 8 evaluates the raw pooled reference $\widehat{\mathbf{o}}_i$ on 30 LongBench texts of 8,192 tokens and 128-token blocks. The output is estimated using the Stage 1 procedure with $p_1 = 0.94$. Let $\mathcal{X}$ contain all text–block pairs. For each layer ℓ and query head h, we compute one cosine similarity by summing the dot products and squared norms over all pairs in $\mathcal{X}$ before taking their ratio:

$$c_{\ell h}^{\text{global}} = \frac{\sum_{x \in \mathcal{X}} \widehat{\mathbf{o}}_{\ell h x}^\top \mathbf{o}_{\ell h x}^{\text{dense}}}{\sqrt{\sum_{x \in \mathcal{X}} \|\widehat{\mathbf{o}}_{\ell h x}\|_2^2} \sqrt{\sum_{x \in \mathcal{X}} \|\mathbf{o}_{\ell h x}^{\text{dense}}\|_2^2}}. \tag{D.3}$$

Thus, each layer contributes 32 head-level cosine values to the violin plot.

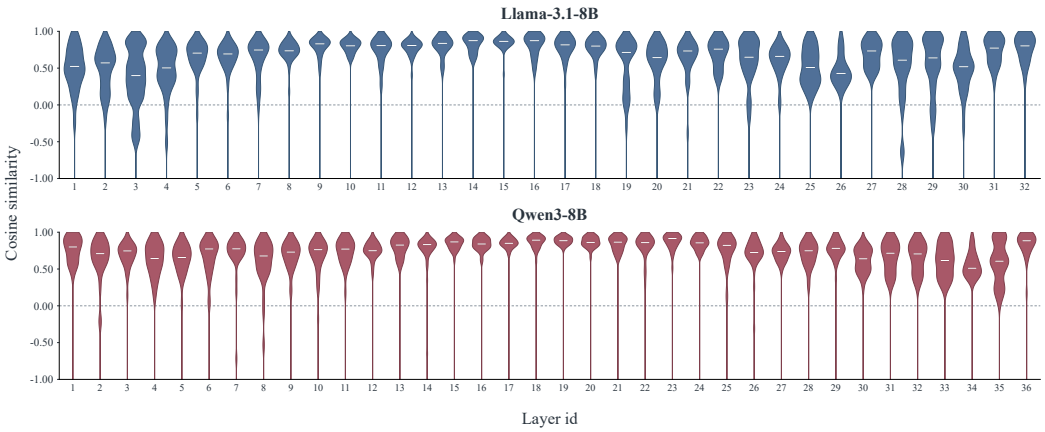

Figure 8: Head-level cosine similarity between the Stage 1 global-output estimate and the dense attention output across layers.

Results and interpretation. Every layer has a positive median pooled cosine. The layer medians range from 0.399 to 0.873 for Llama and from 0.510 to 0.914 for Qwen, with stronger alignment across many middle layers. Several layers have a broad spread across heads, and some heads have negative pooled cosine. These results show that the low-cost estimate captures a meaningful portion of the dense output information and can serve as a coarse prior for block selection. However, the variation across layers and heads also shows that this prior is only approximate and provides limited output information in the less-aligned cases.

D.3 Stage 1 Candidate-Pool Coverage

Stage 1 selects candidates using attention mass alone, whereas Stage 2 combines routing relevance with value-dependent output deviation. We first answer the operational question—whether preselection excludes blocks that the combined score would retain—and then use the component distributions to explain the observed coverage.

Diagnostic setup. All four figures in this section use Llama-3.1-8B at layers 5, 15, 25, and 31 (zero-based), evaluated on two fixed PG19 texts at nested input lengths of 16K, 32K, 48K, and 64K. The activations follow the same dense BF16 trajectory. Each layer/length combination contains 512 query rows: two texts, 32 heads, and eight sampled query-block rows per text and head. We retain the diagnostic's native antidiagonal QK scores, their normalized pooled weights, covariance-corrected value deviations, and frozen output-reference calibration. The distribution plots apply no additional per-row, per-head, or per-layer normalization to these recorded deviations. Keeping the sampled inputs and layer indices aligned makes it possible to compare the mass profiles, deviation distributions, and selection coverage within this diagnostic.

Core coverage result. Figure 9 uses an all-visible Stage 2 reference: output-relative selection is applied to all causally visible blocks with a raw mass target of 0.90 and $\beta = 0.6$. The reference is fixed while p_1 changes. With $\mathcal{S}^{\text{full}}_{i,h}$ denoting the reference selection and $\mathcal{C}_{i,h}(p_1)$ the mass-only pool,

$$r_{\text{miss}}(p_1) = \frac{\sum_{i,h} |\mathcal{S}^{\text{full}}_{i,h} \setminus \mathcal{C}_{i,h}(p_1)|}{\sum_{i,h} |\mathcal{S}^{\text{full}}_{i,h}|}. \tag{D.4}$$

The missed fraction is 0.973–2.383% at $p_1 = 0.90$, 0.029–0.224% at $p_1 = 0.92$, and at most 0.035% at $p_1 = 0.94$. Thus the deployed conservative pool covers at least 99.96% of the reference-selected blocks in every measured layer/length combination.

Attention mass versus value deviation. Figures 10 and 11 compare two groups defined by cumulative attention mass: the shell between 0.85 and 0.90, and the tail beyond 0.94. The plots cover layers 5, 15, 25, and 31 and input lengths from 16K to 64K (zero-based layer indices).

Attention-mass separation. Figure 10 shows a clear and persistent gap between the two groups. Across every sampled layer and context length, the 0.85–0.90 shell retains markedly larger raw Stage 1 block mass over almost the entire within-group rank range. By contrast, the tail beyond 0.94 drops rapidly and approaches zero for a large fraction of its blocks. The same ordering appears from 16K through 64K and from the early layer 5 to the late layer 31, although the precise curve shape changes with depth. This separation means that a tail block begins with a substantial QK-score disadvantage before the value-aware term is applied. Consequently, only tail blocks with unusually large output deviation can plausibly overtake candidates already included in the conservative mass pool.

Output-deviation overlap. Figure 11 presents a different pattern. The ranked output-deviation curves of the shell and tail largely overlap in layers 5 and 25, while the tail is only moderately higher through much of layer 15. The strongest difference occurs in layer 31, where the upper portion of the tail has larger deviations than the corresponding shell profile. Even there, the gap contracts toward the lower-ranked portion of the distributions, and the two groups remain on a comparable scale over most ranks. Thus, low attention mass does not imply uniformly negligible value deviation, but neither does the tail exhibit a broad value advantage that would generally compensate for the mass gap in Figure 10.

Joint interpretation. Together, the two figures motivate the asymmetric two-stage design. Stage 1 uses attention mass to remove a region whose scores are consistently weak, whereas Stage 2 preserves the value term for finer discrimination inside a conservative candidate pool. The figures are distributional rather than paired blockwise comparisons: ranks are computed independently within each group and component, so points at the same horizontal position need not represent the same block. The profiles therefore explain the empirical coverage trend without asserting that mass alone determines the final selection.

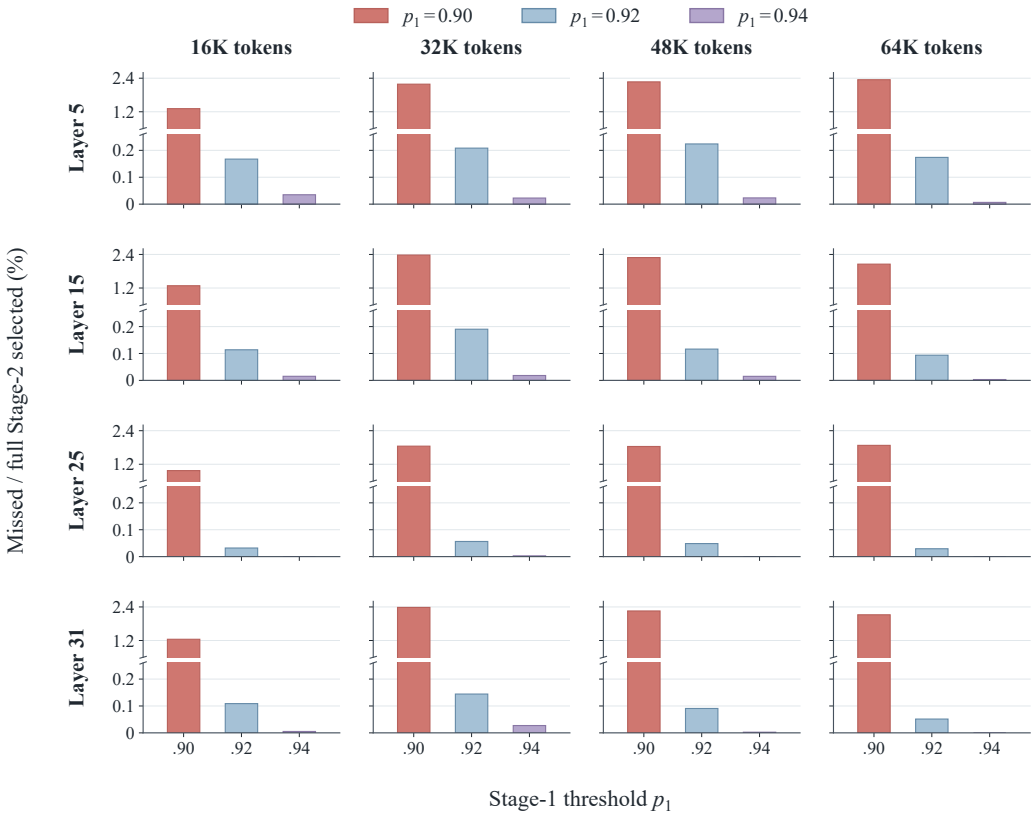

Figure 9: Layerwise coverage of value-aware selections by mass-only preselection for $p_1 \in \{0.90, 0.92, 0.94\}$. The missed fraction is the pooled count of all-visible reference-selected blocks outside the Stage 1 pool divided by the pooled reference-selected count. At $p_1 = 0.94$, the missed fraction is at most 0.035% across all 16 measured layer/length combinations.

For positive deviations, a discarded block b can outrank a candidate c under Eq. 6 only if

$$\beta \log \frac{r_{ib}^{\text{value}}}{r_{ic}^{\text{value}}} > \widehat{z}_{ic}^{\text{AD}} - \widehat{z}_{ib}^{\text{AD}}. \tag{D.5}$$

Here $\widehat{z}_{ib}^{\text{AD}}$ is the antidiagonal estimate of the pooled QK component in the combined log score. For $\beta > 0$, a large routing-score gap requires a correspondingly large value-deviation ratio. The observed profiles support using the value term to refine a conservative candidate pool rather than scoring every low-mass block. This explains why the conservative pool has high measured coverage without claiming that attention mass alone is sufficient for the final ranking.

Layerwise distribution of value deviation. Figure 12 separates the all-visible deviation distribution by layer and input length. Each panel pools the visible block observations for its 512 query rows, with equal weight per block observation. The 16 panels contain 2,295,552 observations in total. The vertical axis reports the percentage of observations in each 0.05-wide $\log_{10}$ bin. Common axes retain the original norm scale and all observed values, so both location and spread can be compared across panels. The distribution includes retained and discarded blocks; it is not conditioned on either stage's final selection.

The typical deviation increases across the four sampled depths. Across the four lengths, the median is 0.787–0.817 in layer 5, 1.202–1.267 in layer 15, 1.765–1.779 in layer 25, and 2.076–2.085 in layer 31. Within each layer, the largest median is less than 6% above the smallest across these lengths. The late layer also has a broader right tail: its 95th percentile is 6.864–7.180, compared

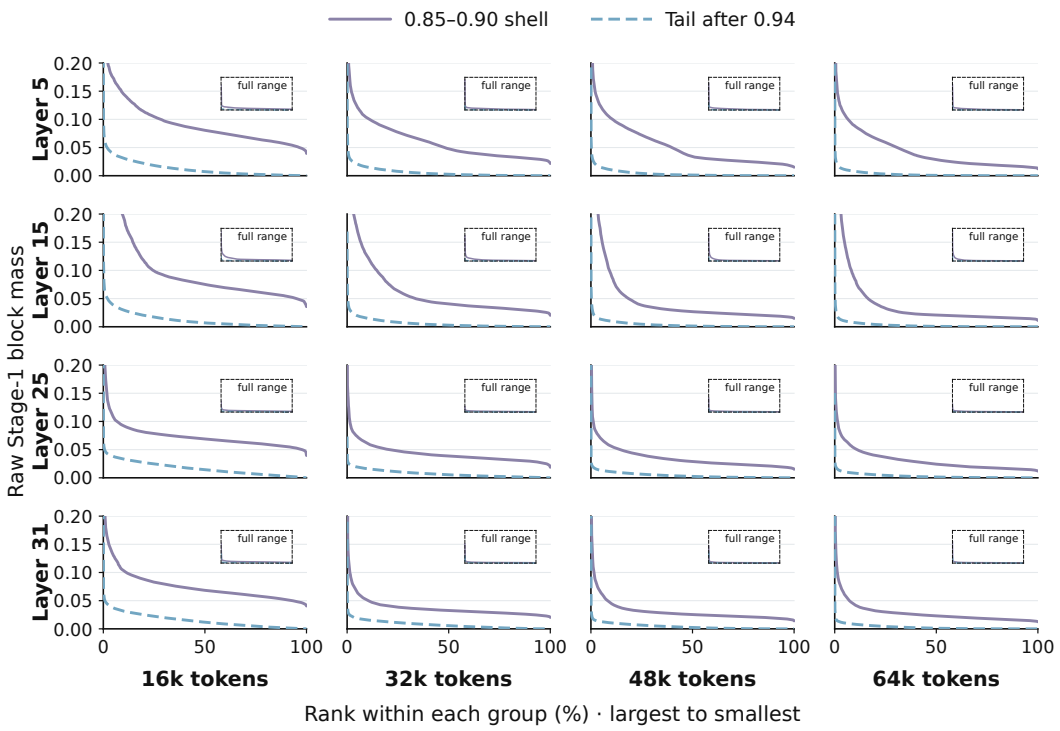

Figure 10: Attention-mass profiles for the 0.85–0.90 cumulative-mass shell and the tail beyond 0.94, sorted independently within each group. The tail has substantially lower raw Stage 1 block mass over most ranks across all sampled layers and context lengths.

with at most 3.501 in the other three layers. Nevertheless, deviations above 10 account for only 0.037–0.050% of layer-31 observations and do not occur in the other plotted layers.

These layerwise views explain why a pooled histogram can obscure meaningful depth-dependent differences. They also clarify the role of the value term: most blocks have moderate deviations, while a small tail can receive substantially higher scores. Whether such a block should survive depends jointly on its deviation and attention mass, as expressed by Eq. D.5. A marginal deviation histogram alone cannot establish that preselection preserves the desired blocks; the following coverage test evaluates that question using the joint score.

The threshold effect is consistent across the four depths, including layer 31 with its broader deviation tail. The length dependence is not strictly monotone: for example, at $p_1 = 0.90$, layer 15 reaches 2.378% at 32K and decreases to 2.053% at 64K. Hence the observed coverage is better summarized by its range across layers and lengths than by assuming that longer inputs necessarily cause more omissions.

Together, the profiles and coverage measurements support using a conservative mass-only pool before applying the more expensive output-relative score. The profiles identify the large mass gap that a tail block must overcome, while the coverage test directly counts the exceptions under the combined score. The diagnostic uses $\beta = 0.6$, rather than the model-specific coefficients of the main benchmark. Its two-text coverage result does not guarantee zero omissions on other inputs, nor does a small missed-block fraction alone bound output error: missed blocks can differ in their individual contributions.

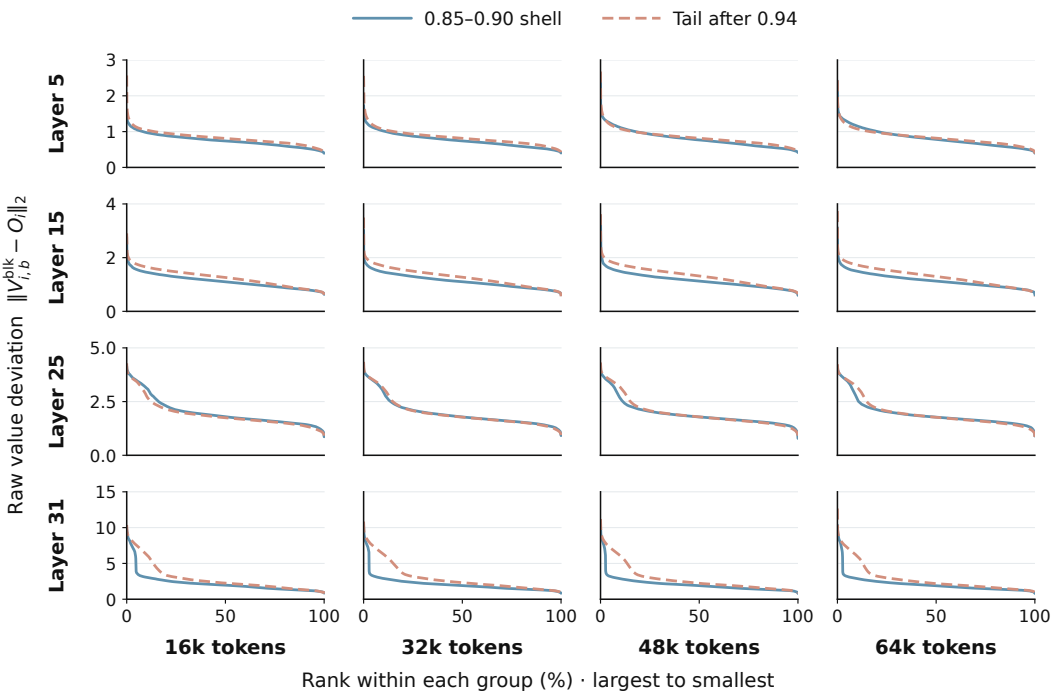

Figure 11: Output-deviation profiles for the 0.85–0.90 cumulative-mass shell and the tail beyond 0.94, sorted independently within each group. The profiles largely overlap except for the upper-ranked tail in later layers, most notably layer 31.

E Runtime and Memory Analysis

E.1 Block-Output Estimator FLOPs

The relevant baseline is the exact token-level block output in Eq. D.1. Let

$$n_{ib} = \sum_{t \in \mathcal{Q}_i} |\{u \in \mathcal{B}_b : u \le t\}| \tag{E.1}$$

be the number of visible query–key pairs. Counting a multiply–add as two FLOPs, exact QK products and value accumulation cost

$$f_{\text{exact}}(i,b) = 4 n_{ib} d_h, \tag{E.2}$$

where $n_{ib} = b_q b_k$ for a fully visible block. A cached covariance descriptor requires one matrix–vector product and one vector addition,

$$f_{\text{Taylor}} = 2d_h^2 + d_h. \tag{E.3}$$

For a fully visible block, the fraction of per-candidate arithmetic retained by the estimator is therefore

$$\frac{f_{\text{Taylor}}}{f_{\text{exact}}} = \frac{2d_h + 1}{4 b_q b_k}, \qquad \text{saving} = 1 - \frac{2d_h + 1}{4 b_q b_k}. \tag{E.4}$$

When $b_q = b_k = d_h = d$, exact block computation is $\mathcal{O}(d^3)$ whereas the estimate is $\mathcal{O}(d^2)$ per candidate. At $d = 128$, Eq. E.4 gives a 99.61% reduction in per-candidate arithmetic.

The covariance costs $2b_k d_h^2 + \mathcal{O}(b_k d_h + d_h^2)$ to construct, but it is computed once per KV-head/key-block descriptor and reused across the corresponding query heads and query blocks. Query pooling likewise costs $2b_q d_h$ once per query block. If P candidate pairs reuse U unique descriptors across Q query blocks, the estimator requires

$$P(2d_h^2 + d_h) + U\left[2b_k d_h^2 + \mathcal{O}(b_k d_h + d_h^2)\right] + 2Q b_q d_h \tag{E.5}$$

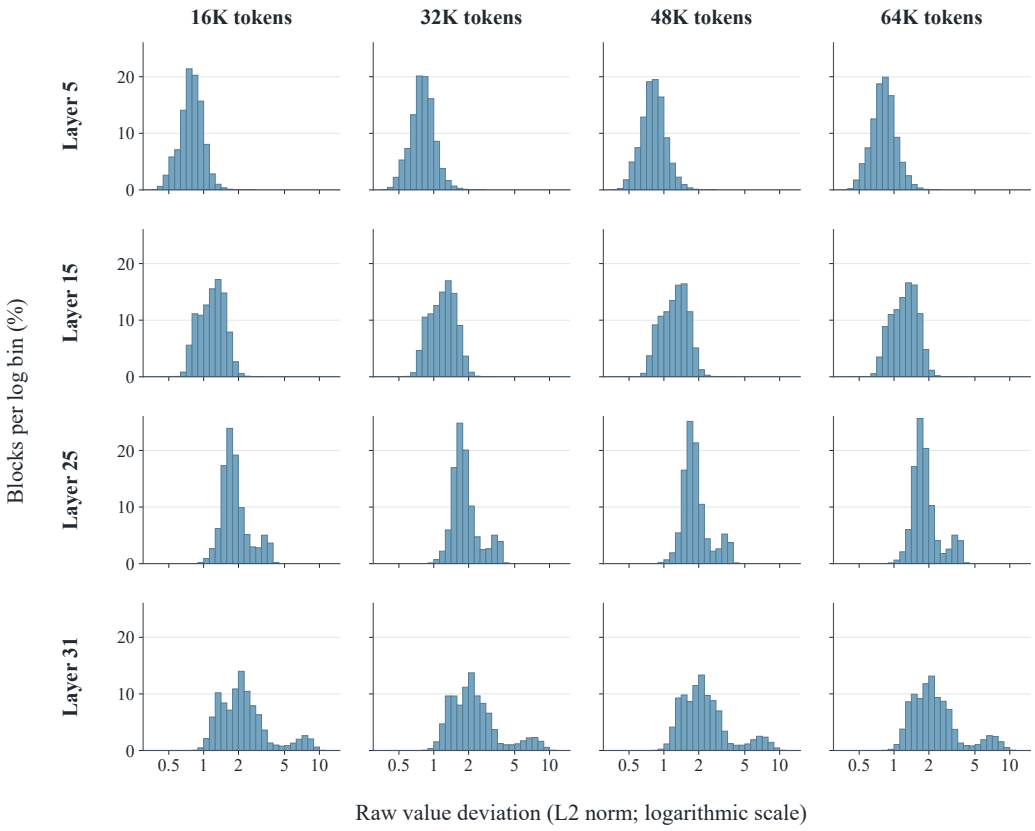

Figure 12: Layerwise distributions of raw value-deviation norms over all visible blocks. Rows show Llama-3.1-8B layers 5, 15, 25, and 31 (zero-based); columns show four input lengths. Each panel pools two PG19 texts, 32 heads, and eight query rows per text and head, with equal weight per visible block observation. The horizontal axis is logarithmic, and bins have width 0.05 in $\log_{10}$ deviation. All panels share the same axes, without additional per-head or per-row normalization. Layer 31 has a broader right tail, while each layer's central distribution varies modestly across lengths.

FLOPs, compared with $4d_h \sum_{(i,b)} n_{ib}$ for exact token-level block outputs. The descriptor and pooling terms are therefore amortized as their reuse increases. For a causal diagonal block, Eq. E.2 uses the actual smaller n_{ib}, so its saving is lower than the full-block expression. These counts exclude memory traffic, kernel launches, Stage 1, and the final sparse attention. End-to-end measurements appear in Figures 5 and 6.

E.2 Memory Overhead of Q/K Copies and Covariance

We first calculate the storage required by the auxiliary tensors used for low-precision scoring and block-output approximation. These tensors consist of low-precision Q/K copies and a blockwise key–value covariance cache. In practice, our implementation stores the Q/K copies in INT8 and the covariance cache in BF16. We then measure the actual instantaneous memory overhead of the INT8 Q/K copies and covariance cache.

Storage calculation. For batch size one, sequence length n divisible by block size $b = 128$, head dimension d_h, h_q query heads, and h_{kv} KV heads, let $b_{\text{tot}} = \lceil n/b \rceil$. An INT8 entry occupies one byte, and a BF16 covariance entry occupies two bytes. The buffer sizes are

$$m_{\text{copies}} = nd_h(h_q + h_{kv}) \text{ bytes}, \qquad m_{\text{cov}} = 2h_{kv}b_{\text{tot}}d_h^2 \text{ bytes}. \tag{E.6}$$

The covariance cache is shared by query heads using the same KV head. At 128K with $d_h = 128$, $h_q = 32$, and $h_{kv} = 8$, the Q and K copies require 512 and 128 MiB, respectively, and the covariance cache requires 256 MiB.

Verification with real model activations. We replay saved dense-trajectory Q/K/V activations from Qwen3-8B (YaRN $\times 4$, layer index 18) for four fixed inputs: two PG19 texts and two InfiniteBench inputs (code debugging and key–value retrieval). The actual INT8 quantization and BF16 covariance routines are executed on an NVIDIA A100 80GB at 16K, 32K, 64K, and 128K, with batch size one. After warm-up, we measure the actual instantaneous memory increase produced by the INT8 Q/K copies and covariance cache during complete POEM execution.

Table 6: Predicted and measured auxiliary tensor storage (MiB). The actual INT8 copies and BF16 covariance cache match their shape-based predictions on all four inputs at each length. Q/K measurements also include the small FP32 quantization-scale tensors.

	Q/K copies		Covariance	
Length	Predicted payload	Measured with scales	Predicted	Measured
16K	80	80.020	32	32
32K	160	160.039	64	64
64K	320	320.078	128	128
128K	640	640.156	256	256

Table 6 confirms that these buffers occupy the predicted storage in the implementation. The measured Q/K difference is exactly the FP32 scale storage, $4(h_q + h_{kv})b_{\text{tot}}$ bytes; at 128K it is 0.15625 MiB. All 48 component checks (Q copy, K copy, and covariance across 16 input/length combinations) match the tensor payload plus its returned metadata exactly. Instrumented and uninstrumented complete POEM calls produce bitwise-identical attention outputs on all 16 cases.

Both terms grow linearly with n for fixed block size and head geometry. Their payload sizes sum to 896 MiB at 128K, but this sum is not a claim about the complete model's peak GPU memory. The INT8 copies are temporary Stage 1 tensors, while covariance is constructed for Stage 2; their lifetimes need not overlap. This accounting is limited to the two buffer families and excludes the original Q/K/V activations and other attention workspaces.

End-to-end peak GPU memory. We additionally record the peak allocated GPU memory during full-model prefilling. Table 7 shows that the runtime memory peaks of POEM and dense attention are nearly identical. Even at 128K tokens, POEM increases the peak by only 2.005 MiB.

Table 7: End-to-end peak allocated GPU memory during full-model prefilling. The increase is computed from the unrounded measurements.

Length	Dense (GiB)	POEM (GiB)	Increase (MiB)
16K	19.147	19.147	0.033
32K	23.030	23.031	0.127
64K	30.796	30.797	0.503
128K	46.329	46.331	2.005

The model-level peak is determined by the allocation of MLP activations. When this peak is reached, the temporary low-precision Q/K copies and the covariance cache used by POEM have already been released. These buffers therefore do not overlap with the dominant MLP allocation, explaining why their standalone storage does not translate into a comparable increase in the end-to-end peak memory.

F Offline Threshold Calibration

Algorithm 2 expands the error-ranked batch-bisection procedure in Section 4.3. A group is a layer-specific set of query heads sharing KV projections. Its members share one threshold. To keep

calibration notation consistent with the main configuration, let $\theta_g = p_1 p_{2,g}$ denote the group's overall estimated mass target; the candidate-relative threshold used by Algorithm 1 is $p_{2,g} = \theta_g / p_1$. Groups are ordered by increasing error in a single-group perturbation diagnostic, with all other groups kept at the reference setting. This ordering prioritizes groups expected to tolerate a smaller budget.

Data partition. Calibration uses two fixed, disjoint manifests: $\mathcal{D}_{\mathrm{rank}}$ for the dense-logit cache and single-group KL probes, and $\mathcal{D}_{\mathrm{search}}$ for accepting or rejecting threshold changes. The benchmark test instances reported in Section 5 belong to neither manifest and are evaluated only after thresholds are frozen. The example identifiers, tokenized lengths, sampled prediction positions, and random seed must be stored with the released configuration so that the split can be reconstructed exactly.

Definition of the group error. We first run dense attention on $\mathcal{D}_{\mathrm{rank}}$ and cache its final vocabulary logits. Set every group's overall target to 0.9 and evaluate the resulting sparse model against this dense reference. Writing $\boldsymbol{\theta}^{\mathrm{ref}} = (0.9, \ldots, 0.9)$, define

$$d(\boldsymbol{\theta}) = \frac{1}{|\mathcal{I}|} \sum_{(x,t)\in\mathcal{I}} d_{\mathrm{KL}}\big(\mathrm{softmax}(\mathbf{z}^{d}_{x,t}) \,\big\|\, \mathrm{softmax}(\mathbf{z}^{s}_{x,t}(\boldsymbol{\theta}))\big)\,, \qquad d_{\mathrm{ref}} = d(\boldsymbol{\theta}^{\mathrm{ref}}), \tag{F.1}$$

where $\mathcal{I}$ is the fixed set of example–position pairs drawn only from $\mathcal{D}_{\mathrm{rank}}$. The softmax is over the vocabulary, with the dense distribution as the first argument of KL, consistent with Eq. 7. For each group g, independently reduce only its threshold by a factor of 0.9 and recompute the full model's final logits:

$$\theta_h^{(g)} = \begin{cases} 0.81, & h = g, \\ 0.9, & h \neq g, \end{cases} \qquad e_g := d(\boldsymbol{\theta}^{(g)}), \qquad \Delta e_g := e_g - d_{\mathrm{ref}}. \tag{F.2}$$

All examples, prediction positions, and other settings stay fixed. Each probe starts from $\boldsymbol{\theta}^{\mathrm{ref}}$, so reductions do not accumulate across groups while building this ranking. We order groups by increasing e_g; subtracting the common baseline d_{ref} gives the identical ordering by Δe_g. Lower values indicate that the single-group reduction better preserves the dense prediction distribution. This KL error determines the trial order; the task score below determines whether a trial reduction is accepted.

Search rule. Initialize every group at $\tau_0 = 0.9$ and evaluate its score on $\mathcal{D}_{\mathrm{search}}$. Each trial changes only the current batch to $\rho\tau_0$, with $\rho = 0.9$, leaving previously accepted changes in place. Accept a trial if its search-set score does not decrease. If the score decreases, restore the accepted configuration and split the batch into two contiguous halves in the error ordering. The lower-error half is tested first. A rejected singleton retains τ_0. The procedure makes a single reduction decision per group; it does not repeatedly multiply already accepted thresholds by ρ. The search terminates when the stack is empty or when the predefined maximum number of calibration steps $T_{\max}$ is reached. If the step limit is reached first, groups in the remaining unvisited batches retain their current thresholds.

Final configuration and calibration cost. All reported runs use $p_1 = 0.94$, $\tau_0 = 0.90$, and $\rho = 0.90$. Consequently, an unreduced group uses $p_{2,g} = 0.90/0.94 \approx 0.95745$, while an accepted reduced group uses $p_{2,g} = 0.81/0.94 \approx 0.86170$. The frozen artifact therefore needs only one bit per group in addition to the shared constants: unreduced or reduced. Its practical effect is assessed by the module ablation in Table 5.

Algorithm 2 Error-Ranked Threshold Calibration

Input: Groups $\mathcal{G}$; errors e_g from Eq. F.2; initial threshold τ_0; reduction factor $\rho = 0.9$; fixed-search-set score $s(\boldsymbol{\theta})$; maximum calibration steps $T_{\max}$.
Output: Frozen group thresholds $\boldsymbol{\theta}$.

$\mathcal{G} \leftarrow \mathrm{SortAscending}(\mathcal{G}, e_g)$
$\boldsymbol{\theta} \leftarrow (\tau_0, \ldots, \tau_0)$; $s_{\mathrm{best}} \leftarrow s(\boldsymbol{\theta})$; $\mathcal{Q} \leftarrow [\mathcal{G}]$ ▷ LIFO stack
$t \leftarrow 0$
while $\mathcal{Q} \neq \emptyset$ **and** $t < T_{\max}$ **do**
  $\mathcal{B} \leftarrow \mathrm{Pop}(\mathcal{Q})$; $\boldsymbol{\theta}' \leftarrow \boldsymbol{\theta}$
  $t \leftarrow t + 1$
  Set $\theta'_g \leftarrow \rho\tau_0$ for every $g \in \mathcal{B}$
  $s' \leftarrow s(\boldsymbol{\theta}')$
  if $s' \geq s_{\mathrm{best}}$ **then**
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}'$; $s_{\mathrm{best}} \leftarrow s'$
  else if $|\mathcal{B}| > 1$ **then**
    $(\mathcal{B}_{\mathrm{lo}}, \mathcal{B}_{\mathrm{hi}}) \leftarrow \mathrm{Bisect}(\mathcal{B})$
    $\mathrm{Push}(\mathcal{Q}, \mathcal{B}_{\mathrm{hi}})$; $\mathrm{Push}(\mathcal{Q}, \mathcal{B}_{\mathrm{lo}})$
  end if
end while
return $\boldsymbol{\theta}$