# UniPrefill: Universal Long-Context Prefill Acceleration via Block-wise Dynamic Sparsification

**Qihang Fan**[1,2,3,*], **Huaibo Huang**[1,2,†], **Zhiying Wu**[3],
**Bingning Wang**[3,‡], **Ran He**[1,2]

[1]MAIS & NLPR, Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
[3]WeChat, Tencent

## Abstract

As large language models (LLMs) continue to advance rapidly, they are becoming increasingly capable while simultaneously demanding ever-longer context lengths. To improve the inference efficiency of long-context processing, several novel low-complexity hybrid architectures have recently been proposed, effectively alleviating the computational burden of long-context inference. However, existing research on long-context prefill acceleration remains predominantly focused on sparse attention mechanisms, which achieve their maximum speedup only on full-attention models. When transferred to emerging architectures — such as linear/full attention hybrids or sliding window/full attention hybrids — these prefill acceleration approaches suffer significant performance degradation. Furthermore, such methods are generally incompatible with continuous batching, making them difficult to integrate into modern inference engines such as vLLM. To this end, we propose **UniPrefill**, a prefill acceleration framework for full-attention models and for hybrid architectures that contain full-attention layers, which directly accelerates the model's computation at the token level. We further implement UniPrefill as a continuous batching operator and extend vLLM's scheduling strategy to natively support prefill-decode co-processing and tensor parallel for UniPrefill, enabling its seamless integration into vLLM. UniPrefill achieves up to **2.26x** speedup in Time-To-First-Token (TTFT), with the acceleration becoming increasingly pronounced as the number of concurrent requests grows.

## 1 Introduction

The rapid advancement of large language models (LLMs) has driven their deployment across an increasingly diverse range of real-world applications, from document understanding and code generation to multi-turn dialogue and retrieval-augmented generation [24, 25, 34, 33, 32, 1, 40]. Alongside this expansion in capability, the context lengths that LLMs are expected to process have grown dramatically — modern deployments routinely involve sequences of tens of thousands of tokens, and the demand for hundred-thousand-token or even million-token contexts is becoming commonplace. This trend places enormous pressure on inference efficiency, as the canonical Softmax Self-Attention [26] mechanism scales quadratically with sequence length, incurring prohibitive computational costs when processing long contexts.

To address the quadratic complexity bottleneck, a new generation of hybrid architectures has emerged that interleave computationally efficient layers with full attention layers. Two representative families

† Corresponding Author.
‡ Project Leader.
* Work done during internship at WeChat.

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

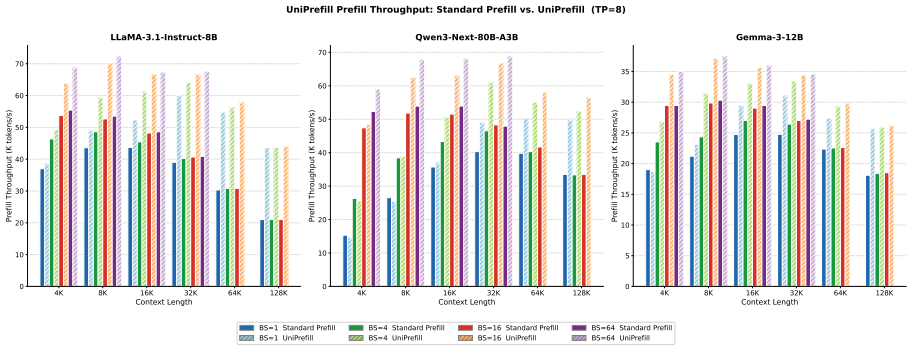



Figure 1: Prefill throughput comparison between Standard Prefill and UniPrefill across three model architectures and varying batch sizes (tensor parallel size is set to 8). All experiments are conducted within vLLM, with UniPrefill deeply integrated into vLLM's continuous batching scheduler. We evaluate prefill throughput (K tokens/s) on LLaMA-3.1-Instruct-8B [4] (full attention), Qwen3-Next-80B-A3B [22] (linear/full attention hybrid), and Gemma-3-12B [14] (sliding window/full attention hybrid) across context lengths from 4K to 128K and batch sizes of 1, 4, 16, and 64. Solid bars denote Standard Prefill and hatched bars denote UniPrefill. UniPrefill consistently achieves higher throughput across all three architectures, with gains becoming more pronounced at longer context lengths and larger batch sizes.

have gained particular traction: linear/full attention hybrids, which replace a subset of attention layers with linear recurrent mechanisms [10, 3, 36, 5, 7] to reduce per-layer complexity from $O(N^2)$ to $O(N)$; and sliding window/full attention hybrids, which restrict most attention layers to a fixed local context window while retaining a small number of global full-attention layers for long-range dependencies [14, 12]. These hybrid designs substantially reduce the theoretical complexity of long-context inference and have been widely adopted in recently released production-grade models.

Despite the proliferation of hybrid architectures, the research community's efforts on prefill acceleration have remained heavily concentrated on sparse attention [13, 21, 8]. Representative works such as MInference [13] have demonstrated impressive prefill speedups, achieving up to 10× acceleration on long sequences under the full-attention-only setting. However, this focus on sparse attention comes with a fundamental limitation: the acceleration is tightly coupled to the full attention operation itself. In hybrid architectures where full attention constitutes only a fraction of all layers, the marginal benefit of accelerating solely those attention layers diminishes considerably. For instance, in a linear/full attention hybrid with a 3:1 ratio, at most one out of every four layers can be accelerated by existing sparse attention methods, leaving the dominant computational budget entirely untouched. This architectural mismatch renders existing prefill acceleration approaches far less effective on the new generation of hybrid models.

A second, equally critical limitation of existing prefill acceleration methods is their incompatibility with continuous batching, the scheduling paradigm that underpins modern high-throughput inference engines such as vLLM [15, 42]. Methods such as FlexPrefill [16] operate on individual requests in isolation and assume static batch composition, making them fundamentally difficult to integrate into a continuous batching scheduler where requests enter and exit the batch dynamically. As a result, these methods have largely remained research prototypes and have not been successfully embedded into production inference systems.

To overcome both limitations, we propose UniPrefill, a prefill acceleration framework for full-attention models and for hybrid architectures that contain full-attention layers, which exploits a key insight: token importance can be estimated at full attention layers and propagated across all subsequent layers. Specifically, UniPrefill applies a lightweight block-wise scoring criterion at each full attention layer to identify and drop computationally redundant tokens. Once a token is dropped, it is excluded from all downstream computation in the remaining layers of the block. This cascading effect means that a single token-dropping decision at the attention layer translates into a proportional reduction in computation across the entire layer stack, not merely the attention sublayer. As a result, UniPrefill achieves substantial reductions in both attention FLOPs and GEMM FLOPs simultaneously, making it effective regardless of whether the model is a pure full-attention Transformer or hybrid architecture.

Beyond the algorithmic design, we address the systems integration challenge by implementing UniPrefill as a continuous batching operator [38] and extending vLLM [15]'s scheduler to natively support prefill-decode co-processing under UniPrefill's token-dropping regime. This tight integration allows UniPrefill to function as a transparent acceleration layer within production inference engines, without requiring changes to model weights or serving infrastructure.

We evaluate UniPrefill on RULER [11], LongBench, LongBench Pro and InfiniteBench, across three model architectures and context lengths from 4K up to 1M tokens. Results demonstrate that UniPrefill keeps accuracy close to dense prefill while achieving up to $2.26\times$ TTFT speedup at 128K and $7.06\times$ at 1M tokens, as illustrated in Fig. 1. Notably, the speedup scales favorably with both context length and the number of concurrent requests (see Fig. 1), making UniPrefill particularly well-suited for high-concurrency production serving scenarios where prefill cost is the dominant bottleneck.

Our main contributions are summarized as follows:

- We propose **UniPrefill**, a token-level prefill acceleration framework that drops tokens at full attention layers and propagates sparsity across all subsequent layers, reducing both attention and GEMM FLOPs simultaneously, which enables consistent speedups across heterogeneous hybrid architectures.
- We implement UniPrefill as a continuous batching operator and integrate it into vLLM [15] via extended scheduling strategies that support prefill-decode co-processing and tensor parallel, enabling seamless production-ready deployment.
- Extensive experiments on RULER and on realistic long-context benchmarks (LongBench, LongBench Pro, InfiniteBench) show that UniPrefill delivers up to $2.26\times$ TTFT speedup at 128K and up to $7.06\times$ at 1M tokens, with a small and task-dependent accuracy change, and that the gains scale with both context length and request concurrency.

## 2 Related Works

**Hybrid LLM Architectures.** To overcome the quadratic complexity of Softmax attention, a rich body of work has proposed efficient sequence modeling alternatives, including state space models, linear attention variants, and recurrent architectures [10, 3, 23, 36, 37, 6, 5, 18, 35, 41]. To balance efficiency and expressiveness, hybrid architectures have emerged that interleave full attention with these efficient alternatives [22, 17, 14, 28, 12], and have been widely adopted in recently released production models. However, existing prefill acceleration methods remain largely tailored to full-attention-only architectures, limiting their effectiveness on this new generation of models.

**Sparse Attention for Prefill Acceleration.** Exploiting the inherent sparsity in attention score matrices is a well-established strategy for accelerating the prefill stage. A body of work identifies static or dynamic sparse patterns — such as vertical, slash, and block-sparse structures — and skips the corresponding attention computations [13, 39, 21, 29, 16, 2]. These methods have demonstrated substantial speedups on full attention models [13, 16, 31, 27]. However, they share two fundamental limitations: their acceleration is tightly coupled to the attention operation itself, leaving FFN and GEMM computations entirely unaccelerated, and they are generally incompatible with continuous batching [38], making integration into production inference engines such as vLLM [15] non-trivial. UniPrefill addresses both limitations by operating at the token level and propagating sparsity across all layers.

## 3 Method

In this section, we present UniPrefill, a prefill acceleration framework for full-attention models and for hybrid models that contain full-attention layers. The overall pipeline is illustrated in Fig. 2.

### 3.1 Preliminaries

Consider an input sequence $\mathbf{x} = [x_1, \ldots, x_N]$ processed by a hybrid LLM consisting of $B$ blocks. Each block $b$ contains a full attention layer followed by $M_b$ sublayers (linear attention, sliding window

attention, FFN, etc.). Let $\mathbf{H}^{(b,0)} \in \mathbb{R}^{N\times d}$ denote the block input. The goal of prefill is to compute the final hidden state $\mathbf{h}_N^{(L)}$ for next-token prediction:

$$P(x_{N+1} \mid x_{1:N}) = \text{LMHead}\Big(\mathbf{h}_N^{(L)}\Big) \tag{1}$$

Standard prefill incurs $\mathcal{O}(N^2 d_k)$ per full attention layer and $\mathcal{O}(Nd^2)$ per GEMM sublayer, totaling $\mathcal{O}(N^2 d_k + M_b N d^2)$ per block.

### 3.2 Token Importance Estimation

Since next-token prediction depends solely on $\mathbf{h}_N^{(L)}$, the contribution of token $i$ to the final hidden state at block $b$ is:

$$\mathbf{h}_N^{(b,1)} = \sum_{i=1}^{N} \mathbf{A}_{N,i}^{(b)} \cdot \mathbf{v}_i^{(b)} + \mathbf{h}_N^{(b,0)}, \tag{2}$$

where $\mathbf{A}_{N,i}^{(b)} = \text{softmax}_i\Big(\mathbf{q}_N^{(b)} {\mathbf{K}^{(b)}}^\top / \sqrt{d_k}\Big)$ is the full-sequence attention weight. A token $i$ is negligible to next-token prediction when $\mathbf{A}_{N,i}^{(b)} \approx 0$.

To reduce estimation variance, we aggregate over the last $n$ query positions instead of a single position:

$$s_i^{(b)} = \frac{1}{n} \sum_{j=N-n+1}^{N} \mathbf{A}_{j,i}^{(b)}, \tag{3}$$

requiring an $n \times N$ attention computation at cost $\mathcal{O}(nNd_k)$, negligible for $n \ll N$.

In practice, importance estimation and token selection operate at *block granularity*. We partition the input sequence into non-overlapping blocks of size $G$: $\mathcal{B}_g = \{(g-1)G+1, \ldots, \min(gG, N)\}$, $g = 1, \ldots, \lceil N/G \rceil$. For efficiency, the partial GEMM $\mathbf{S} = \mathbf{Q}_{[N-n:N]}\mathbf{K}^\top \in \mathbb{R}^{n\times N}$ is computed first; an online softmax is then applied *across the full sequence dimension* to obtain properly normalised attention weights, after which scores are reduced within each block:

$$\bar{s}_g^{(b)} = \frac{1}{G} \sum_{i\in\mathcal{B}_g} \frac{1}{n} \sum_{j=N-n+1}^{N} \mathbf{A}_{j,i}^{(b)}, \tag{4}$$

where the softmax normalisation is performed over the complete key sequence before the block reduction, ensuring $\bar{s}_g^{(b)}$ reflects the true attention mass captured by block $g$. This reduces the number of selection decisions from $N$ to $\lceil N/G \rceil$ while preserving the accuracy of importance estimation.

**Relationship to SnapKV.** Our importance estimation shares a surface-level similarity with SnapKV [19], which also uses an observation window to identify important tokens. However, the two methods differ fundamentally in objective and scope. SnapKV completes a full $N \times N$ prefill across all layers before applying its selection to compress the KV cache for decode—the prefill FLOPs are entirely unaffected. UniPrefill applies selection *during* prefill, propagating the drop decision forward through all subsequent layers. Formally, whereas SnapKV saves at most $\mathcal{O}(N \cdot d_{kv})$ in decode-time memory per layer, UniPrefill saves $(1-\rho^{(b)}) \cdot M_b \cdot \mathcal{O}(Nd^2)$ in prefill-time FLOPs per block, where $\rho^{(b)} = |\mathcal{S}^{(b)}|/N$ is the token retention ratio—a quantity that grows linearly with $M_b$ and is entirely absent in SnapKV.

### 3.3 Top-$p$ Token Selection

Let $\pi$ be the permutation sorting block-level scores $\{\bar{s}_g^{(b)}\}$ in descending order. We retain the minimal set of blocks:

$$\mathcal{S}^{(b)} = \{\pi(1), \ldots, \pi(k^*)\}, \qquad k^* = \min k \;\text{ s.t. }\; \frac{\sum_{j=1}^{k} \bar{s}_{\pi(j)}^{(b)}}{\sum_g \bar{s}_g^{(b)}} \geq p \tag{5}$$

The dropped set is $\bar{\mathcal{S}}^{(b)} = [N] \setminus \mathcal{S}^{(b)}$. Two structural elements are always retained regardless of their scores: the first $A$ tokens (attention sinks [30]) and the last $n$ tokens (the query window itself), ensuring causal consistency and numerical stability.

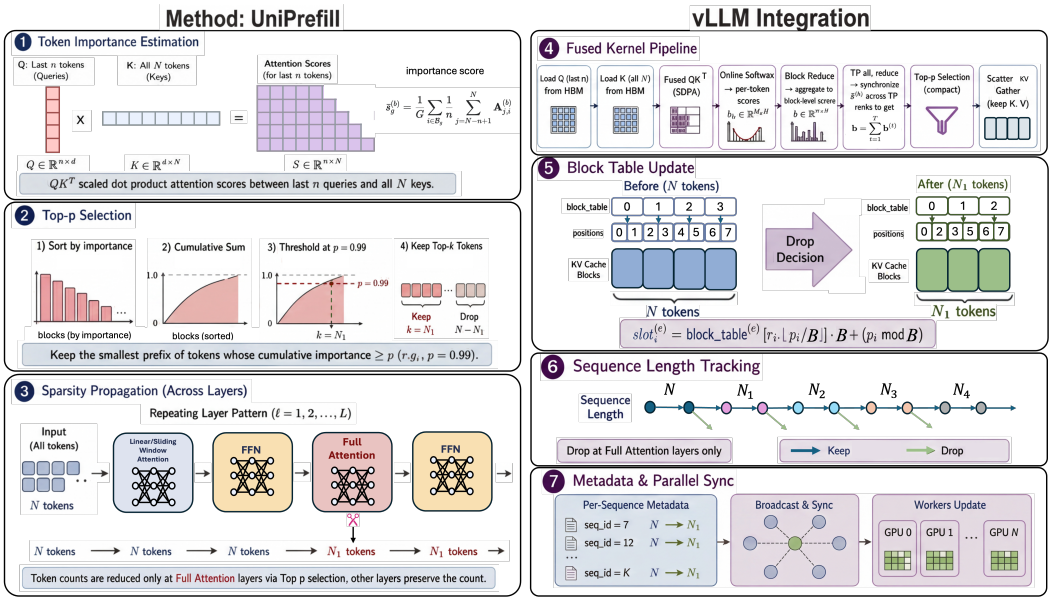



Figure 2: **Overview of UniPrefill. Left:** UniPrefill estimates token importance via block-level attention scores from the last $n$ queries **(1)**, retains the smallest set of token blocks whose cumulative importance reaches $p$ **(2)**, and propagates the resulting sparsity across all subsequent sub-layers within each repeating layer pattern **(3)**. **Right:** UniPrefill is deeply integrated into vLLM via a fused kernel pipeline **(4)**, with KV cache block tables **(5)**, per-layer sequence length tracking **(6)**, and tensor-parallel metadata synchronisation **(7)** updated accordingly.

**Error bound.** The perturbation to any retained position $j$ due to dropping $\bar{\mathcal{S}}^{(b)}$ satisfies:

$$\left\|\Delta \mathbf{h}_j^{(b,1)}\right\| \;\leq\; \left(\sum_{i \in \bar{\mathcal{S}}^{(b)}} \mathbf{A}_{j,i}^{(b)}\right) \cdot V_{\max}^{(b)} \;\leq\; (1-p) \cdot V_{\max}^{(b)} \tag{6}$$

where $V_{\max}^{(b)} = \max_i \|\mathbf{v}_i^{(b)}\|$. Setting $p = 0.99$ guarantees that at most $1\%$ of the total attention mass is discarded, providing a direct information-theoretic bound on the approximation error at the attention layer.

**Top-$p$ vs. top-$k$.** A fixed top-$k$ is insensitive to the actual distribution of attention: when attention is highly concentrated, top-$k$ retains many unnecessary tokens; when diffuse, it may drop tokens with non-trivial contributions. Top-$p$ adapts automatically—the retained set is small when attention is concentrated and large when it is diffuse—providing a uniform bound on approximation error regardless of sequence length or content, which top-$k$ cannot guarantee.

### 3.4 Sparsity Propagation Across All Layers

**Permanent dropping.** Dropping is irreversible: once a token is removed at the full attention layer of block $b$, it never re-enters the computation. Unlike cache-eviction schemes that keep a diluted copy of the sequence, we carry no stale hidden state forward. Writing $\mathcal{S}^{(b)}$ for the active set entering block $b$ (with $\mathcal{S}^{(0)} = [N]$), every remaining sublayer of the block—full attention, linear attention, sliding window attention, and FFN—is computed only on the nested set $\mathcal{S}^{(b+1)} \subseteq \mathcal{S}^{(b)}$:

$$\mathbf{H}_{\mathcal{S}^{(b+1)}}^{(b,m+1)} = f_m\left(\mathbf{H}_{\mathcal{S}^{(b+1)}}^{(b,m)}\right), \qquad m = 1, \ldots, M_b. \tag{7}$$

At block $b+1$ the next full attention layer re-scores whatever survives, so the active set is monotonically decreasing, $\mathcal{S}^{(0)} \supseteq \mathcal{S}^{(1)} \supseteq \cdots$. Original indices of retained tokens are kept solely as positional and KV-slot metadata for cache addressing (Eq. 15); dropped tokens contribute nothing to any later layer, and their key/value entries are neither recomputed nor reused. Consequently, a drop decision at layer $\ell$ reduces the token count for *all* layers $\ell' > \ell$, and the per-layer accounting below is

consistent with the implementation. In practice dropping starts at the second full-attention layer; on LLaMA-3.1-8B at 128K the local retention is $0.42$ at the first drop and $0.05$ at the last one, giving a final cumulative retention of $0.4\%$.

**FLOPs analysis.** Let $\mathcal{L}_{\text{drop}} = \{\ell_1, \ldots, \ell_K\}$ be the drop layers and let $\rho_k = |\mathcal{S}^{(\ell_k+1)}|/|\mathcal{S}^{(\ell_k)}|$ be the *local* retention of the $k$-th drop, so that $N_k = N \prod_{j \le k} \rho_j$ tokens remain active below layer $\ell_k$. With $M$ non-attention sublayers per layer, the cost below the $k$-th drop is

$$C_k = \underbrace{\mathcal{O}\big(N_k^2 d_k\big)}_{\text{attention}} + \underbrace{\mathcal{O}\big(M N_k d^2\big)}_{\text{projection / FFN}}, \tag{8}$$

whereas a sparse-attention method keeps $N_k = N$ in both terms and only discounts the first one, at the single layer where it is applied. Three observations follow, and we separate them deliberately.

*(i) Depth accumulation.* A drop at $\ell_k$ reduces the cost of *all* $L - \ell_k$ layers below it, so the total saving

$$\Delta\text{FLOPs} = \sum_{k=1}^{K} (1 - \rho_k) \cdot (L - \ell_k) \cdot \Big[\mathcal{O}\big(N_k^2 d_k\big) + \mathcal{O}\big(M N_k d^2\big)\Big] \tag{9}$$

grows with the number of layers below the first drop and with $M$—i.e., precisely in the hybrid regime where full attention is a minority of the sublayers.

*(ii) No asymptotic dominance.* An earlier version of this analysis compared the two savings as a ratio and claimed that it diverges as $N \to \infty$. That derivation was incorrect: the ratio of UniPrefill's linear-in-$N$ GEMM saving to the quadratic-in-$N$ attention saving of a single sparse layer is

$$\frac{(L - \ell_1) \cdot N d^2}{N^2 d_k} = \Theta\left(\frac{1}{N}\right) \xrightarrow{N \to \infty} 0, \tag{10}$$

so no asymptotic advantage over sparse attention follows, and we withdraw the claim. UniPrefill's advantage is structural rather than asymptotic: it is the only one of the two that also reduces the $MNd^2$ projection/FFN term, which dominates on hybrid models whose attention is already sub-quadratic.

*(iii) Context-length effect.* Empirically, longer inputs are more redundant, so adaptive top-$p$ retains a smaller fraction ($0.5\%$ at 128K vs. $0.1\%$ at 1M on Qwen3-Next, Tab. 3) and the measured speedup keeps growing with $N$. This is a property of the selection rule, not of the asymptotic ratio in (ii); we therefore restrict the claim to the range we actually measure ($\le$ 1M tokens).

**Error propagation.** Assuming each sublayer $f_m$ is $L_m$-Lipschitz, the accumulated error at block end satisfies:

$$\left\|\Delta \mathbf{h}_j^{(b, M_b+1)}\right\| \le (1 - p) \cdot V_{\max}^{(b)} \cdot \prod_{m=1}^{M_b} L_m \tag{11}$$

Layer normalization and residual connections constrain $\prod_m L_m$ in practice, preventing unbounded error amplification across layers.

### 3.5 Fused Kernel and vLLM Integration

**Kernel design.** We implement the importance estimation and top-$p$ selection pipeline as a sequence of four fused kernels operating directly on the variable-length packed token representation indexed by `cu_seqlens`, without materializing per-request tensors or padding. The pipeline proceeds as follows:

$$\mathbf{S} = \mathbf{Q}_{[N-n:N]}\mathbf{K}^\top \in \mathbb{R}^{n \times N} \xrightarrow{\text{softmax}} \mathbf{o} \in \mathbb{R}^{N} \xrightarrow{\text{reduce}} \mathbf{b} \in \mathbb{R}^{\lceil N/G \rceil} \xrightarrow{\text{top-}p} \mathcal{M} \in \{0,1\}^N \tag{12}$$

The partial GEMM kernel computes $\mathbf{S}$ with tiled $Q$-$K$ blocking and inline causal masking. The softmax kernel aggregates $\mathrm{softmax}(\mathbf{S})$ over the $n$ query rows via a numerically stable two-pass online algorithm, yielding per-token importance scores $\mathbf{o}$. The block-reduce kernel contracts $\mathbf{o}$ across both the head and spatial dimensions within each block of size $G$, producing the block-level score vector $\mathbf{b}$.

The top-$p$ kernel performs sort-and-threshold entirely on-GPU without CPU round-trips. We encode each (score, index) pair into a single `int64` word via a monotone IEEE-754 bitcast mapping:

$$\phi(x) = \begin{cases} \mathrm{bits}(x) \oplus \texttt{0x80000000} & x \geq 0 \\ \mathrm{bits}(x) \oplus \texttt{0xFFFFFFFF} & x < 0 \end{cases} \quad \Rightarrow \quad \texttt{packed} = \big(\phi(b_g) \ll 32\big) \mid g \tag{13}$$

Sorting packed words descending, computing a cumulative sum of scores, and thresholding at $p$ yields the keep mask $\mathcal{M}$, which is scattered back to original positions. A final expansion kernel lifts $\mathcal{M}$ from block to token granularity, unconditionally setting $\mathcal{M}_i = 1$ for attention-sink tokens $i < A$ and query-window tokens $i \geq N - n$.

**Tensor parallelism.** Under tensor parallelism of degree $T$, each rank observes only $1/T$ of the attention heads, yielding a partial block score $\mathbf{b}^{(t)}$. We synchronize via:

$$\mathbf{b} = \sum_{t=1}^{T} \mathbf{b}^{(t)} \tag{14}$$

before the top-$p$ kernel, ensuring a consistent drop decision across all TP ranks.

**vLLM scheduler integration.** Integrating token dropping into vLLM's continuous batching scheduler [38, 15] requires maintaining correctness across three coupled state structures: layer-wise attention metadata, KV cache slot mappings, and per-request KV length tracking across decode steps.

Upon a drop event at layer $\ell$, we propagate updated metadata to all downstream layers $\ell' > \ell$ by patching `query_start_loc`, `seq_lens`, and `num_actual_tokens` to reflect the compacted token stream $|\mathcal{S}^{(\ell)}|$. Physical KV cache slot mappings for each layer $\ell'$ are recomputed as:

$$\texttt{slot}_i^{(\ell')} = \texttt{block_table}^{(\ell')}[r_i,\ \lfloor p_i/B \rfloor] \cdot B + (p_i \bmod B) \tag{15}$$

where $p_i$ is the logical position of the $i$-th retained token, $B$ is the KV block size, and $\texttt{block_table}^{(\ell')}$ is the physical block table of layer $\ell'$—which may differ between global and sliding-window attention layers [14].

During decode, each layer $\ell'$ must attend over only the tokens that were physically written to its KV cache during prefill. We maintain a per-request drop history $\{(\ell_k, s_k^r)\}$ recording the retained sequence length $s_k^r$ after each drop event at layer $\ell_k$. The effective KV length visible to layer $\ell'$ during decode is then:

$$\mathrm{seqused}_r^{(\ell')} \ = \ s_r^{(\ell^-)} + \Delta_r, \qquad \Delta_r = \mathrm{kv_len}_r - \mathrm{orig_len}_r \tag{16}$$

where $\ell^- = \max\{\ell_k \in \mathcal{L}_{\mathrm{drop}} : \ell_k < \ell'\}$ is the last drop layer preceding $\ell'$, and $\Delta_r$ counts autoregressive tokens appended since prefill. This per-layer `seqused` correction is injected into the forward context before each decode step, ensuring every attention layer observes a KV sequence length precisely consistent with its written cache entries—without any modification to model weights or the PagedAttention memory allocator.

# 4 Experiments

We evaluate UniPrefill across two dimensions: accuracy and efficiency. For accuracy, we compare UniPrefill against existing prefill acceleration methods on the RULER [11] long-context benchmark across multiple model architectures. For efficiency, we measure prefill throughput under varying context lengths and batch sizes within our vLLM deployment. Finally, we conduct ablation studies to analyze the contribution of each design choice in UniPrefill.

## 4.1 Experimental Setup

We select three model architectures to validate the effectiveness of UniPrefill: LLaMA-3.1-8B-Instruct [4], which consists entirely of full-attention layers; Qwen3-Next-80B-A3B [22], a linear/full-attention hybrid with a 3:1 ratio; and Gemma-3-12B [14], a sliding-window/full-attention hybrid with a 5:1 ratio. We set the top-$p$ threshold to $0.99$, $0.99$, and $0.98$ for the three models, respectively. The minimum dropping granularity is set to a block size of $G = 64$ tokens, and importance scores are estimated using the last $n = 128$ query tokens. To preserve attention sinks [30], the first 128 tokens are always retained.

| Method | 4K | 8K | 16K | 32K | 64K | 128K | Avg | 4K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct (Full Attention) | | | | | | | | | | | | | |
| Baseline | 97.36 | 95.98 | 94.62 | 91.02 | 86.29 | 76.89 | 90.36 | 1.00× | 1.00× | 1.00× | 1.00× | 1.00× | 1.00× |
| LazyLLM [9] | 89.16 | 81.12 | 70.32 | 64.39 | 56.28 | 49.71 | 68.50 | 1.09× | 1.19× | 1.28× | 1.74× | 2.21× | 2.51× |
| SlimInfer [20] | 90.23 | 82.04 | 71.39 | 67.12 | 57.10 | 45.36 | 68.87 | 1.07× | 1.16× | 1.25× | 1.66× | 1.98× | 2.07× |
| MInference [13] | 96.71 | 95.78 | 95.51 | 90.76 | 87.12 | 78.21 | 90.68 | 0.82× | 0.86× | 0.98× | 1.03× | 1.15× | 1.34× |
| FlexPrefill [16] | 96.34 | 95.12 | 94.83 | 88.96 | 84.31 | 78.13 | 89.62 | 0.83× | 0.89× | 1.02× | 1.08× | 1.24× | 1.46× |
| XAttention [31] | 95.98 | 95.23 | 94.68 | 88.06 | 83.92 | 78.16 | 89.34 | 0.92× | 0.96× | 1.03× | 1.08× | 1.21× | 1.38× |
| ProxyAttn [27] | 96.78 | 95.46 | 95.49 | 89.28 | 85.31 | 78.49 | 90.14 | 0.82× | 0.89× | 1.03× | 1.11× | 1.36× | 1.79× |
| UniPrefill | 96.53 | 95.83 | 95.41 | 89.77 | 85.28 | 79.87 | **90.45** | 1.21× | 1.34× | 1.37× | 1.62× | 2.01× | 2.26× |
| Qwen3-Next-80B-A3B (Linear/Full Attention Hybrid) | | | | | | | | | | | | | |
| Baseline | 96.83 | 95.67 | 95.07 | 94.38 | 94.51 | 92.09 | 94.76 | 1.00× | 1.00× | 1.00× | 1.00× | 1.00× | 1.00× |
| LazyLLM [9] | 89.13 | 82.37 | 70.69 | 64.37 | 58.12 | 55.17 | 69.98 | 1.11× | 1.18× | 1.29× | 1.40× | 1.55× | 1.74× |
| SlimInfer [20] | 88.11 | 80.36 | 67.13 | 63.22 | 57.13 | 55.36 | 68.55 | 1.14× | 1.22× | 1.27× | 1.34× | 1.42× | 1.56× |
| MInference [13] | 96.62 | 94.38 | 94.49 | 94.27 | 94.28 | 91.81 | 94.31 | 0.96× | 0.98× | 1.00× | 1.00× | 1.02× | 1.05× |
| FlexPrefill [16] | 96.36 | 95.03 | 94.17 | 93.91 | 92.89 | 91.44 | 93.97 | 0.96× | 0.98× | 1.00× | 1.01× | 1.04× | 1.08× |
| XAttention [31] | 96.03 | 94.81 | 94.03 | 93.01 | 93.06 | 90.23 | 93.53 | 0.97× | 0.99× | 1.00× | 1.00× | 1.02× | 1.05× |
| ProxyAttn [27] | 96.31 | 94.13 | 94.23 | 93.47 | 93.51 | 91.62 | 93.88 | 0.96× | 0.98× | 1.00× | 1.02× | 1.05× | 1.11× |
| UniPrefill | 96.67 | 94.49 | 94.29 | 93.63 | 93.13 | 91.41 | **93.94** | 1.08× | 1.21× | 1.24× | 1.39× | 1.42× | 1.68× |
| Gemma-3-12B (Sliding Window/Full Attention Hybrid) | | | | | | | | | | | | | |
| Baseline | 94.01 | 89.12 | 85.98 | 80.76 | 68.89 | 61.22 | 79.99 | 1.00× | 1.00× | 1.00× | 1.00× | 1.00× | 1.00× |
| LazyLLM [9] | 86.11 | 80.34 | 75.23 | 68.42 | 54.12 | 43.38 | 67.93 | 1.23× | 1.32× | 1.37× | 1.42× | 1.53× | 1.64× |
| SlimInfer [20] | 88.23 | 83.14 | 79.12 | 69.33 | 53.02 | 40.11 | 68.83 | 1.15× | 1.24× | 1.32× | 1.39× | 1.45× | 1.52× |
| MInference [13] | 93.56 | 89.51 | 86.01 | 80.04 | 67.09 | 59.31 | 79.25 | 0.98× | 0.99× | 1.00× | 1.00× | 1.01× | 1.03× |
| FlexPrefill [16] | 93.63 | 89.16 | 85.49 | 79.23 | 65.69 | 58.63 | 78.64 | 0.98× | 0.99× | 1.00× | 1.01× | 1.02× | 1.04× |
| XAttention [31] | 93.06 | 89.24 | 85.67 | 79.14 | 66.18 | 56.24 | 78.26 | 0.99× | 1.00× | 1.01× | 1.01× | 1.02× | 1.02× |
| ProxyAttn [27] | 93.67 | 89.52 | 85.31 | 78.97 | 65.31 | 59.93 | 78.79 | 0.98× | 0.99× | 1.01× | 1.01× | 1.03× | 1.06× |
| UniPrefill | 93.18 | 89.76 | 86.47 | 79.08 | 66.32 | 58.38 | **78.87** | 1.15× | 1.21× | 1.22× | 1.26× | 1.31× | 1.49× |

Table 1: Performance vs. efficiency across different models and methods. Evaluation scores (left) and TTFT speedup relative to baselines (right) are reported. For a fair comparison, all models are evaluated using HuggingFace Transformers with batch size equals to 1.

### 4.2 Results on RULER

RULER [11] is a comprehensive long-context benchmark that evaluates LLMs across diverse task categories including retrieval, multi-hop tracing, aggregation, and question answering, with configurable context lengths up to 128K tokens. Unlike prior benchmarks that rely on simple needle-in-a-haystack tests, RULER provides a more rigorous and systematic assessment of true long-context understanding, making it a widely adopted standard for evaluating long-context LLM performance.

Tab. 1 presents RULER scores and TTFT speedups across three model architectures. UniPrefill achieves the best accuracy-efficiency tradeoff among all acceleration methods. LazyLLM and SlimInfer suffer notable accuracy degradation across all three architectures, while sparse attention methods preserve accuracy but yield diminishing speedups on hybrid architectures, with gains often below $1.1\times$ at 128K. UniPrefill strikes the optimal balance: it retains accuracy close to the Baseline while delivering up to $2.26\times$, $1.68\times$, and $1.49\times$ TTFT speedup at 128K context length on LLaMA-3.1-8B, Qwen3-Next-80B-A3B, and Gemma-3-12B, respectively, demonstrating consistent effectiveness across full-attention and hybrid architectures.

### 4.3 vLLM Integration

Tab. 2 reports prefill throughput within vLLM across three architectures. UniPrefill consistently improves throughput as context length and batch size increase, achieving up to $+109\%$, $+68\%$, and $+42\%$ gains on LLaMA-3.1-8B, Qwen3-Next-80B-A3B, and Gemma-3-12B, respectively. The speedup scales favorably with both context length and batch size, demonstrating that UniPrefill is particularly effective in the high-concurrency, long-context regime that dominates production serving workloads.

### 4.4 Scaling Beyond 128K and Beyond RULER

**Longer contexts.** Tab. 3 extends the evaluation to 256K–1M tokens on Qwen3-Next and 512K on Seed-OSS. Dense accuracy degrades over this range, yet UniPrefill tracks it closely: final retention

| BSZ | 4K | 8K | 16K | 32K | 64K | 128K | 4K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct (Full Attention) | | | | | | | | | | | | |
| 1 | 36984 | 43586 | 43697 | 39027 | 30324 | 21013 | 38522 (+4%) | 48932 (+12%) | 52314 (+20%) | 59984 (+54%) | 54786 (+81%) | 43672 (+107%) |
| 4 | 46372 | 48632 | 45436 | 40210 | 30812 | 21054 | 49336 (+6%) | 59372 (+22%) | 61148 (+35%) | 64113 (+59%) | 56108 (+82%) | 43698 (+108%) |
| 16 | 53764 | 52643 | 48221 | 40671 | 30834 | 21062 | 63762 (+19%) | 70213 (+33%) | 66762 (+38%) | 66512 (+64%) | 57678 (+87%) | 44042 (+109%) |
| 64 | 55431 | 53541 | 48603 | 40869 | — | — | 68721 (+24%) | 72139 (+35%) | 67361 (+39%) | 67618 (+65%) | — | — |
| Qwen3-Next-80B-A3B (Linear/Full Attention Hybrid) | | | | | | | | | | | | |
| 1 | 15314 | 26534 | 35712 | 40312 | 39807 | 33512 | 14621 (−5%) | 25364 (−4%) | 36912 (+3%) | 48972 (+21%) | 50324 (+26%) | 49732 (+48%) |
| 4 | 26334 | 38442 | 43326 | 46534 | 40322 | 33364 | 25498 (−3%) | 38872 (+1%) | 50442 (+16%) | 60894 (+31%) | 55136 (+37%) | 52442 (+57%) |
| 16 | 47432 | 51853 | 51544 | 48303 | 41693 | 33489 | 48446 (+2%) | 62468 (+20%) | 62983 (+22%) | 66798 (+38%) | 57938 (+39%) | 56398 (+68%) |
| 64 | 52334 | 53936 | 53936 | 47932 | — | — | 58936 (+13%) | 67848 (+26%) | 68123 (+26%) | 68631 (+43%) | — | — |
| Gemma-3-12B (Sliding Window/Full Attention Hybrid) | | | | | | | | | | | | |
| 1 | 19013 | 21203 | 24733 | 24763 | 22367 | 18103 | 18673 (−2%) | 23132 (+9%) | 29436 (+19%) | 31023 (+25%) | 27384 (+22%) | 25673 (+42%) |
| 4 | 23531 | 24361 | 27013 | 26432 | 22536 | 18403 | 26783 (+14%) | 31432 (+29%) | 33014 (+22%) | 33432 (+26%) | 29232 (+30%) | 25932 (+41%) |
| 16 | 29468 | 29867 | 29031 | 27012 | 22613 | 18513 | 34512 (+17%) | 37136 (+24%) | 35674 (+23%) | 34419 (+27%) | 29732 (+31%) | 26231 (+42%) |
| 64 | 29471 | 30324 | 29471 | 27236 | — | — | 35016 (+19%) | 37365 (+23%) | 35912 (+22%) | 34578 (+27%) | — | — |

Table 2: Prefill throughput (tokens/s) of Standard Prefill and UniPrefill measured within vLLM (TP= 8) across three model architectures, six context lengths (4K–128K), and four batch sizes (BSZ). The left half of each group reports Standard Prefill throughput; the right half reports UniPrefill throughput.

falls from $0.5\%$ at 128K to $0.1\%$ at 1M because longer inputs are proportionally more redundant, and TTFT speedup grows monotonically to $7.06\times$/$7.30\times$. We do not extrapolate beyond 1M.

**Beyond RULER.** Tab. 4 covers realistic workloads: the 17-task LongBench average moves from $48.08$ to $47.80$, LongBench Pro loses $0.36$ points uniformly across languages and difficulty splits, and on InfiniteBench—which stresses long *outputs* after irreversible compaction—En.Sum loses $0.97$ ROUGE-L with generation length unchanged ($1079.6$ vs. $1086.3$ tokens) and Math.Calc loses $0.04$ over $\sim$29K-token traces.

**Failure modes.** A controlled 64K RAG test isolates where dropping hurts: degradation grows from $-0.3/-0.1$ for direct lexical evidence to $-1.6/-1.2$ for globally distributed evidence (LLaMA/Qwen3-Next), while retention on Qwen3-Next rises from $0.42\%$ to $0.65\%$. Adaptive top-$p$ thus becomes more conservative under diffuse attention but cannot protect evidence that is consistently low-salience; losses concentrate on global counting (CWE/FWE lose $1.9$/$4.3$ on Qwen3-Next).

**Overhead and same-engine comparison.** Scoring, selection, compaction and TP synchronisation cost $5.2\%$ of baseline prefill latency at 8K and $2.2\%$ at 128K on LLaMA, since only $O(N/G)$ block scores are all-reduced per drop. Within the *same* vLLM engine, MInference improves prefill throughput by at most $1.26\times$ whereas UniPrefill reaches $2.09\times$ (LLaMA, 8K–128K, batch sizes 1–16).

### 4.5 Ablation Study

**Block Size.** We ablate the block size $G \in \{32, 64, 128\}$ (Tab. 9). At short context lengths, $G = 128$ yields the highest speedup due to lower selection overhead per drop decision. As context length grows, $G = 32$ surpasses $G = 128$ in speedup, since finer granularity allows more tokens to be dropped, achieving up to $+121\%$ and $+78\%$ throughput gain on LLaMA-3.1-8B and Qwen3-Next-80B-A3B at 128K, respectively. We adopt $G = 64$ as the default, which balances selection overhead and drop rate across all context lengths.

| Length | Baseline | UniPrefill | Final ret. | TTFT speedup (BSZ 1 / 16) | |
|---|---|---|---|---|---|
| Qwen3-Next-80B-A3B | | | | | |
| 128K | 92.09 | 91.41 | 0.5% | 1.48× | 1.68× |
| 256K | 91.21 | 90.82 | 0.3% | 2.13× | 2.36× |
| 512K | 85.31 | 85.44 | 0.1% | 3.78× | 4.00× |
| 1M | 79.02 | 79.27 | 0.1% | 7.06× | 7.30× |
| Seed-OSS | | | | | |
| 128K | 94.42 | 94.08 | 0.6% | 2.41× | 2.56× |
| 256K | 86.08 | 85.31 | 0.3% | 4.09× | 4.27× |
| 512K | 78.33 | 79.64 | 0.2% | 7.61× | 7.88× |

Table 3: RULER score, final cumulative retention and TTFT speedup beyond the 128K range of Tab. 1 (vLLM, TP= 8). Retention is the product of the local ratios at the designated drop layers, i.e., the active sequence below the last drop rather than the model-wide token usage.

| Benchmark (model) | Baseline | UniPrefill | TTFT |
|---|---|---|---|
| LongBench, 17-task avg (LLaMA-3.1-8B) | 48.08 | 47.80 | 1.63× |
| LongBench Pro, overall (Qwen3-Next) | 51.54 | 51.18 | 1.48× |
| InfiniteBench En.Sum, ROUGE-L (Qwen3-Next) | 34.95 | 33.98 | 1.48× |
| InfiniteBench Math.Calc (Qwen3-Next) | 2.16 | 2.12 | 1.48× |
| RAG stress test, 64K (LLaMA / Qwen3-Next) | — | $-1.0/-0.7$ | 1.90×/1.40× |

Table 4: Quality beyond RULER. LongBench averages 17 tasks (QA, multi-document QA, summarisation, code completion); the last row gives the mean degradation over the five evidence conditions of the 64K RAG stress test.

| last $n$ | 4K | 8K | 16K | 32K | 64K | 128K | Avg |
|---|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | | | | | | | |
| 128 | 96.53 | 95.83 | 95.41 | 89.77 | 85.28 | 79.87 | 90.45 |
| 32 | 95.32 | 94.13 | 93.18 | 86.22 | 82.63 | 75.13 | 87.77 |
| 512 | 96.72 | 96.04 | 95.63 | 90.23 | 84.96 | 79.38 | 90.49 |

Table 5: Ablation study of last $n$. RULER scores under different values of the $n$ on LLaMA-3.1-8B-Instruct. $n = 128$ is adopted as the default.

**Last $n$.** Tab. 5 reports RULER scores under different values of last $n \in \{32, 128, 512\}$. $n = 32$ leads to a noticeable accuracy drop, as too few query tokens introduce high variance in importance estimation. $n = 512$ recovers accuracy but incurs higher computational overhead. $n = 128$ achieves the best balance and is adopted as the default.

## 5 Conclusion

We present UniPrefill, a framework for long-context LLM prefill acceleration that applies to full-attention models and to hybrid architectures containing full-attention layers. By estimating token importance via block-wise top-$p$ selection at full-attention layers and propagating the sparsity mask across all subsequent sub-layers, UniPrefill simultaneously reduces attention and GEMM FLOPs, making it effective across full-attention and hybrid architectures alike. We implement UniPrefill as a fused kernel pipeline and integrate it into vLLM's continuous-batching scheduler without any model weight changes. Experiments on the RULER benchmark show that UniPrefill achieves the best accuracy-efficiency tradeoff among all compared methods, delivering up to 2.26× TTFT speedup with a small, task-dependent accuracy change, with gains scaling favorably with context length and batch size. We hope UniPrefill offers a practical solution for efficient long-context serving.

## Acknowledgments and Disclosure of Funding

This work is supported by the New Generation Artificial Intelligence-National Science and Technology Major Project (Grant No. 2025ZD0123505), the National Natural Science Foundation of China (Grant Nos. 62550062, 62425606, 62576342, 32341009), and the Beijing Natural Science Foundation (Grant Nos. L252145, 4252054). The authors declare that they have no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## References


[1] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

[2] Guanzhong Chen. VSPrefill: Vertical-slash sparse attention with lightweight indexing for long-context prefilling. *arXiv preprint arXiv:2603.04460*, 2026.

[3] Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *International Conference on Machine Learning*, 2024.

[4] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[5] Qihang Fan, Huaibo Huang, Yuang Ai, and Ran He. Rectifying magnitude neglect in linear attention. In *IEEE International Conference on Computer Vision*, 2025.

[6] Qihang Fan, Huaibo Huang, Mingrui Chen, and Ran He. Semantic equitable clustering: A simple and effective strategy for clustering vision tokens. In *IEEE International Conference on Computer Vision*, 2025.

[7] Qihang Fan, Huaibo Huang, and Ran He. Breaking the low-rank dilemma of linear attention. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2025.

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

[9] Qichen Fu, Minsik Cho, Thomas Merth, Sachin Mehta, Mohammad Rastegari, and Mahyar Najibi. Lazyllm: Dynamic token pruning for efficient long context llm inference. *arXiv preprint arXiv:2407.14057*, 2024.

[10] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[11] 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.

[12] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

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

[14] Aishwarya Kamath, Johan Ferret, Shreya Pathak, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[15] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles*, 2023.

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

[17] Barak Lenz, Opher Lieber, Alan Arazi, Amir Bergman, Avshalom Manevich, Barak Peleg, Ben Aviram, et al. Jamba: Hybrid Transformer-Mamba language models. In *ICLR*, 2025.

[18] Aonian Li, Bangwei Gong, Bo Yang, Boji Shan, Chang Liu, Cheng Zhu, Chunhao Zhang, Congchao Guo, Da Chen, Dong Li, Enwei Jiao, Gengxin Li, Guojun Zhang, Haohai Sun, Houze Dong, Jiadai Zhu, Jiaqi Zhuang, Jiayuan Song, Jin Zhu, Jingtao Han, Jingyang Li, Junbin Xie, Junhao Xu, Junjie Yan, Kaishun Zhang, Kecheng Xiao, Kexi Kang, Le Han, Leyang Wang, Lianfei Yu, Liheng Feng, Lin Zheng, Linbo Chai, Long Xing, Meizhi Ju, Mingyuan Chi, Mozhi Zhang, Peikai Huang, Pengcheng Niu, Pengfei Li, Pengyu Zhao, Qi Yang, Qidi Xu, Qiexiang Wang, Qin Wang, Qiuhui Li, Ruitao Leng, Shengmin Shi, Shuqi Yu, Sichen Li, Songquan Zhu, Tao Huang, Tianrun Liang, Weigao Sun, Weixuan Sun, Weiyu Cheng, Wenkai Li, Xiangjun Song, Xiao Su, Xiaodong Han, Xinjie Zhang, Xinzhu Hou, Xu Min, Xun Zou, Xuyang Shen, Yan Gong, Yingjie Zhu, Yipeng Zhou, Yiran Zhong, Yongyi Hu, Yuanxiang Fan, Yue Yu, Yufeng Yang, Yuhao Li, Yunan Huang, Yunji Li, Yunpeng Huang, Yunzhi Xu, Yuxin Mao, Zehan Li, Zekang Li, Zewei Tao, Zewen Ying, Zhaoyang Cong, Zhen Qin, Zhenhua Fan, Zhihang Yu, Zhuo Jiang, and Zijia Wu. Minimax-01: Scaling foundation models with lightning attention. *arXiv preprint arXiv:2501.08313*, 2025.

[19] Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

[20] Lingkun Long, Rubing Yang, Yushi Huang, Desheng Hui, Ao Zhou, and Jianlei Yang. Sliminfer: Accelerating long-context llm inference via dynamic token pruning. In *AAAI*, 2025.

[21] Enzhe Lu, Zhejun Jiang, Jingyuan Liu, Yulun Du, Tao Jiang, Chao Hong, Shaowei Liu, Weiran He, Enming Yuan, Yuzhi Wang, Zhiqi Huang, Huan Yuan, Suting Xu, Xinran Xu, Guokun Lai, Yanru Chen, Huabin Zheng, Junjie Yan, Jianlin Su, Yuxin Wu, Neo Y. Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. Moba: Mixture of block attention for long-context llms. *arXiv preprint arXiv:2502.13189*, 2025.

[22] Qwen Team. Qwen3-Next: Towards ultimate training & inference efficiency. `https://qwen.ai/blog?id=e34c4305036ce60d55a0791b170337c2b70ae51d`, 2025.

[23] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023.

[24] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[25] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[26] 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.

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

[28] Bangjun Xiao, Bingquan Xia, Bo Yang, et al. MiMo-V2-Flash technical report. *arXiv preprint arXiv:2601.02780*, 2026.

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

[30] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *ICLR*, 2024.

[31] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. In *International Conference on Machine Learning*, 2025.

[32] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[33] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang Parameter, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2025.

[34] An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, Junyang Lin, Kai Dang, Kexin Yang, Le Yu, Mei Li, Minmin Sun, Qin Zhu, Rui Men, Tao He, Weijia Xu, Wenbiao Yin, Wenyuan Yu, Xiafei Qiu, Xingzhang Ren, Xinlong Yang, Yong Li, Zhiying Xu, and Zipeng Zhang. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025.

[35] Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations*, 2025.

[36] Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *International Conference on Machine Learning*, 2024.

[37] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, 2024.

[38] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. Orca: A distributed serving system for transformer-based generative models. In *Operating Systems Design and Implementation*, 2022.

[39] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Annual Meeting of the Association for Computational Linguistics*, 2025.

[40] Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, Hao Yu, Hongning Wang, Jiadai Sun, Jiajie Zhang, Jiale Cheng, Jiayi Gui, Jie Tang, Jing Zhang, Juanzi Li, Lei Zhao, Lindong Wu, Lucen Zhong, Mingdao Liu, Minlie Huang, Peng Zhang, Qinkai Zheng, Rui Lu, Shuaiqi Duan, Shudan Zhang, Shulin Cao, Shuxun Yang, Weng Lam Tam, Wenyi Zhao, Xiao Liu, Xiao Xia, Xiaohan Zhang, Xiaotao Gu, Xin Lv, Xinghan Liu, Xinyi Liu, Xinyue Yang, Xixuan Song, Xunkai Zhang, Yifan An, Yifan Xu, Yilin Niu, Yuantao Yang, Yueyan Li, Yushi Bai, Yuxiao Dong, Zehan Qi, Zhaoyu Wang, Zhen Yang, Zhengxiao Du, Zhenyu Hou, and Zihan Wang. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.

[41] Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, Y. T. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025.

[42] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems*, 2024.

## A Implementation and deployment details

UniPrefill is implemented and evaluated on top of vLLM v0.16.0 [15], with full support for prefill-decode co-processing and tensor parallelism, making it compatible with standard production deployment configurations. All throughput experiments are conducted on a single node of 8 NVIDIA H20-96G GPUs under CUDA 12.8, with tensor parallelism degree TP $= 8$, reflecting a typical multi-GPU serving setup in real-world scenarios. The continuous-batching operator is implemented as a set of fused Triton kernels, which are hardware-agnostic by design and theoretically portable across different accelerator platforms.

## B Determinism of the reported numbers

UniPrefill performs no training, no sampling and no stochastic pruning: the model weights are fixed, decoding is greedy, importance scores are accumulated in float32 and the top-$p$ rule is a deterministic sort. Repeated runs with different seeds therefore produce the same scores, and Tab. 6 should be read as a determinism check rather than as a seed-robustness study. We have corrected the corresponding statement in the paper checklist accordingly; every number in the main tables is a single deterministic evaluation.

| random seed | 4K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|
| 0 | 96.53 | 95.83 | 95.41 | 89.77 | 85.28 | 79.87 |
| 321 | 96.53 | 95.83 | 95.41 | 89.77 | 85.28 | 79.87 |
| 3467 | 96.53 | 95.83 | 95.41 | 89.77 | 85.28 | 79.87 |

Table 6: Determinism check on LLaMA-3.1-8B-Instruct: three runs with different seeds give identical RULER scores, as expected for a method without stochastic components.

## C Per-layer retention

Tab. 7 reports the local retention $\rho_k$ measured at each designated drop layer, together with the product that defines the final cumulative retention. Dropping starts at the second full-attention layer; the drop layers are $[7, 11, 15, 19, 23, 27]$ for LLaMA, $[7, 11, 15, 19, 23, 27, 31, 35, 39, 43]$ for Qwen3-Next and $[11, 17, 23, 29, 35, 41]$ for Gemma, with $p = 0.99$, $0.99$ and $0.98$ respectively. Retention becomes more aggressive at longer context lengths, which is what produces the growing speedups of Tab. 3; the per-layer values are not monotone, since which blocks carry the remaining attention mass depends on content.

| Length | Successive local retention $\rho_k$ | Final |
|---|---|---|
| | LLaMA | |
| 8K | [0.82, 0.93, 0.92, 0.86, 0.52, 0.16] | 5.0% |
| 32K | [0.66, 0.82, 0.86, 0.71, 0.42, 0.08] | 1.1% |
| 128K | [0.42, 0.89, 0.77, 0.86, 0.32, 0.05] | 0.4% |
| | Qwen3-Next | |
| 8K | [0.86, 0.97, 0.93, 0.92, 0.86, 0.81, 0.91, 0.46, 0.45, 0.21] | 2.0% |
| 32K | [0.74, 0.96, 0.94, 0.96, 0.81, 0.71, 0.97, 0.42, 0.41, 0.17] | 1.0% |
| 128K | [0.65, 0.96, 0.96, 0.92, 0.77, 0.72, 0.90, 0.39, 0.45, 0.11] | 0.5% |
| | Gemma | |
| 8K | [0.72, 0.94, 0.93, 0.52, 0.42, 0.21] | 2.8% |
| 32K | [0.67, 0.89, 0.95, 0.49, 0.47, 0.12] | 1.5% |
| 128K | [0.62, 0.91, 0.93, 0.51, 0.37, 0.09] | 0.9% |

Table 7: Local retention at each drop layer and the resulting final cumulative retention. The latter describes only the active sequence below the last drop; it is neither the model-wide token usage nor the FLOP ratio.

## D Hyperparameter sensitivity and operator overhead

**Top-$p$ sensitivity.** The left panel of Tab. 8 sweeps $p$ around each model's operating point at 128K. As $p$ increases, the retained fraction grows monotonically ($0.30 \rightarrow 0.65\%$ on LLaMA), and so does *agreement* with dense prefill ($96.8 \rightarrow 98.5\%$), while the speedup degrades gracefully ($2.12\times \rightarrow 2.00\times$). RULER itself need not be monotone—a handful of changed outputs can happen to score higher—which is why we report agreement as the faithfulness metric rather than benchmark accuracy. The sweep also shows that UniPrefill is not brittle to small threshold changes: $\pm 0.005$ in $p$ moves agreement by about one point.

**Operator overhead.** The right panel breaks the end-to-end overhead of Tab. 2 into its three stages at 128K. Importance scoring plus top-$p$ selection dominates, while compaction and metadata maintenance (original positions, KV-slot indices, per-layer sequence lengths) stay at $0.1$–$0.2\%$, and tensor-parallel synchronisation costs at most $0.9\%$ because only $O(N/G)$ block scores are all-reduced per drop. Total overhead is $1.3$–$2.2\%$ of baseline prefill latency at 128K—smaller than at 8K, since the fixed cost is amortised by the larger FLOP reduction.

| $p$ | RULER | Agreement | Final ret. | Score/select | Compact/meta | TP sync |
|---|---|---|---|---|---|---|
| LLaMA | | | | | | |
| 0.985 | 79.31 | 96.8% | 0.30% | 4.1% | 0.2% | 0.9% |
| 0.990 | 79.87 | 97.6% | 0.40% | 2.8% | 0.1% | 0.5% |
| 0.995 | 79.63 | 98.5% | 0.65% | 1.8% | 0.1% | 0.3% |
| Qwen3-Next | | | | | | |
| 0.985 | 90.86 | 96.5% | 0.39% | 2.7% | 0.2% | 0.9% |
| 0.990 | 91.41 | 97.4% | 0.50% | 1.5% | 0.1% | 0.6% |
| 0.995 | 91.72 | 98.3% | 0.78% | 0.8% | 0.1% | 0.4% |
| Gemma | | | | | | |
| 0.975 | 57.66 | 94.8% | 0.67% | 2.9% | 0.2% | 0.8% |
| 0.980 | 58.38 | 96.1% | 0.90% | 1.6% | 0.1% | 0.5% |
| 0.985 | 59.02 | 97.2% | 1.18% | 0.9% | 0.1% | 0.3% |

Table 8: Left: top-$p$ sweep at 128K; "agreement" is the fraction of examples whose final answer equals dense prefill. Right: operator overhead as a percentage of baseline prefill latency at 128K, broken down by stage (TP sync covers the block-score all-reduce); these are per-model measurements, repeated on each row for readability. Sweeping the protected sink length between 64 and 256 changes RULER by at most 0.05 points.

## E Linearity of the recurrent state under dropping

Dropping perturbs the recurrent state of the intervening linear-attention layers, since those layers no longer see the removed tokens. On Qwen3-Next-80B-A3B-Instruct we measure the cosine similarity between UniPrefill and dense states at the Gated DeltaNet layer preceding each full-attention layer. Layer 6 precedes the first drop; later points accumulate the effect of all preceding drops:

| Linear layer | 6 | 10 | 14 | 18 | 22 | 26 | 30 | 34 | 38 | 42 | 46 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Cosine similarity | 1.000 | 0.991 | 0.982 | 0.975 | 0.964 | 0.958 | 0.949 | 0.941 | 0.934 | 0.927 | 0.921 |

Similarity decreases gradually and stays above $0.92$ throughout: dropping perturbs the recurrent state but does not destabilise it. Functionally, this is confirmed by LongBench Pro, where the overall score moves from $51.54$ to $51.18$ uniformly across English/Chinese and all difficulty splits.

## F Ablation details

**Block size $G$.** Tab. 9 varies the selection granularity between 32, 64 and 128 tokens on LLaMA-3.1-8B and Qwen3-Next. The granularity trades selection overhead against drop resolution. With $G = 128$ there are four times fewer block decisions per drop point, which pays off at short contexts ($+26\%$ vs. $+19\%$ at 4K on LLaMA) but caps how much can be removed at long contexts and costs

accuracy on the full-attention model (average RULER $90.45 \rightarrow 88.57$). With $G = 32$ the opposite holds: finer blocks remove more low-salience tokens, reaching $+121\%$ and $+78\%$ throughput at 128K on LLaMA and Qwen3-Next, at the price of twice the selection cost and a small accuracy change. $G = 64$ is the best compromise across lengths and models, and is used everywhere in this paper; the hybrid model is markedly less sensitive to $G$ than the full-attention model, because its per-layer attention cost is already sub-quadratic.

| block size | 4K | 8K | 16K | 32K | 64K | 128K | Avg | 4K | 8K | 16K | 32K | 64K | 128K |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Llama-3.1-8B-Instruct | | | | | | | | | | | | | |
| 64 | 96.53 | 95.83 | 95.41 | 89.77 | 85.28 | 79.87 | 90.45 | +19% | +33% | +38% | +64% | +87% | +109% |
| 128 | 94.32 | 93.62 | 93.07 | 88.12 | 83.38 | 78.90 | 88.57 | +26% | +38% | +45% | +62% | +81% | +96% |
| 32 | 93.42 | 94.63 | 95.46 | 90.23 | 85.67 | 79.88 | 89.88 | +19% | +32% | +36% | +72% | +96% | +121% |
| Qwen3-Next-80B-A3B | | | | | | | | | | | | | |
| 64 | 96.67 | 94.49 | 94.29 | 93.63 | 93.13 | 91.41 | 93.94 | +2% | +20% | +22% | +38% | +39% | +68% |
| 128 | 96.52 | 94.69 | 94.13 | 93.41 | 92.67 | 92.06 | 93.91 | +5% | +22% | +23% | +34% | +37% | +56% |
| 32 | 92.17 | 94.88 | 94.72 | 93.89 | 93.66 | 92.68 | 93.67 | +0% | +19% | +22% | +44% | +51% | +78% |

Table 9: Ablation study of block size $G$. The left panel reports RULER scores under different values of $G$, and the right panel reports the corresponding TTFT speedup of UniPrefill relative to the Baseline.

## G Limitations and Broader Impacts

**Attention mass is an empirical proxy.** The last-$n$-query attention mass correlates with, but does not certify, the causal contribution of a token to the final prediction. Our leave-one-block-out analysis shows a strong rank correlation with end-to-end block-removal loss and near-oracle selection quality, yet the mechanism can fail when evidence receives consistently low salience: in the 64K RAG stress test the degradation grows to $-1.6$ points for globally distributed evidence, and global aggregation tasks (CWE, FWE) lose up to $4.3$ points.

**Degradation is not uniform, and inference is not lossless.** Average RULER changes from $94.76$ to $93.94$ on Qwen3-Next and from $79.99$ to $78.87$ on Gemma. UniPrefill trades a modest, task-dependent quality loss for a large prefill speedup; the aggregate numbers should not be read as lossless acceleration.

**Architecture scope.** Importance scores are obtained at full-attention layers, so models without any full-attention layer—pure state-space, pure linear-attention, or sliding-window-only models—are outside the current scope. "Architecture-agnostic" in this paper means full-attention models and hybrid models that contain full-attention layers.

**Hyperparameter dependence.** The top-$p$ threshold is model-specific ($0.99$, $0.99$, $0.98$) and fixed across context lengths and tasks. The sweep in Tab. 8 shows smooth, monotone behaviour in agreement with dense prefill, but per-model selection is still required.

**Interaction with other inference optimisations.** We evaluate FP8 weight and KV-cache quantization and observe $0.1$–$0.6$ additional RULER points of loss together with a further TTFT improvement. Prefix caching, speculative decoding and other KV-compression schemes are complementary but untested, and combining them with permanent token dropping may introduce interactions that we have not characterised.

**Broader impacts.** This work accelerates the prefill phase of long-context LLM inference and does not change model weights or outputs by design; it inherits the societal considerations of the underlying models, including potential misuse and the environmental cost of large-scale serving, which it reduces. Ethical deployment and safety alignment remain important but lie outside the technical scope of this study.

## NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Yes — Sec. G (Appendix) discusses the limitations in detail: attention mass is an empirical proxy for causal importance, degradation is task-dependent and not uniform, models without full-attention layers are outside the scope, the top-$p$ threshold is model-specific, and interactions with other inference optimisations are untested.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [N/A]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

Answer: [Yes]

Justification: The paper fully disclose all the information needed to reproduce the main experimental results of the paper.

Guidelines:

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

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

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

   Answer: [Yes]

   Justification: The paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results.

   Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: The paper specify all the training and test details necessary to understand the results.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: UniPrefill has no stochastic component — the pretrained weights are fixed, decoding is greedy, and both the importance scores and the top-$p$ selection are deterministic — so repeated runs with different seeds produce identical scores. The seed table is therefore a determinism check rather than a variance estimate. Because there is no run-to-run variability to average over, every number in the tables is a single deterministic evaluation and no error bars are reported.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: The paper provide sufficient information on the computer resources needed to reproduce the experiments.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

Justification: Yes — Sec. G (Appendix, "Broader Impacts") discusses both: accelerated long-context serving lowers the cost and energy of deployment, while it inherits the risks of the underlying models, including misuse.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

Answer: [N/A]

Justification: The paper poses no such risks.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: The creators or original owners of assets used in the paper are properly credited and the license and terms of use explicitly are mentioned and properly respected.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.

- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

16. **Declaration of LLM usage**

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

    Answer: [N/A]

    Justification: This work focuses on system-level optimizations for LLM inference (e.g., accelerating prefill phase through block sparse attention). LLMs are the target of optimization, not a component of our methodology.

    Guidelines:

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