DashAttention: Differentiable and Adaptive Sparse Hierarchical Attention

Yuxiang Huang[1]*, **Nuno M. T. Gonçalves**[2,3,4]*, **Federico Alvetreti**[5], **Lei Li**[4], **Xu Han**[1], **Edoardo M. Ponti**[6], **André F. T. Martins**[2,3,7,8], **Marcos V. Treviso**[2,3,7,8]
[1]Tsinghua University. [2]Instituto Superior Técnico, Universidade de Lisboa.
[3]Instituto de Telecomunicações. [4]Carnegie Mellon University. [5]Sapienza University of Rome.
[6]Imperial College London. [7]Gandara AI. [8]ELLIS Unit Lisbon.
huang-yx21@mails.tsinghua.edu.cn, nuno.m.goncalves@ulisboa.pt

Abstract

Current hierarchical attention methods, such as NSA and InfLLMv2, select the top-k relevant key-value (KV) blocks based on coarse attention scores and subsequently apply fine-grained softmax attention on the selected tokens. However, the top-k operation assumes the number of relevant tokens for any query is fixed and it precludes the gradient flow between the sparse and dense stages. In this work, we propose DashAttention (Differentiable and Adaptive Sparse Hierarchical Attention), which leverages the adaptively sparse α-entmax transformation to select a *variable* number of blocks according to the current query in the first stage. This in turn provides a prior for the second-stage softmax attention, keeping the entire hierarchy *fully differentiable*. Contrary to other hierarchical attention methods, we show that DashAttention is non-dispersive, translating to a better long-context modeling ability. Experiments with large language models (LLMs) show that DashAttention achieves comparable accuracy to full attention with 75% sparsity and a better Pareto frontier than NSA and InfLLMv2, especially in high-sparsity regimes. We also provide an efficient, GPU-aware implementation of DashAttention in Triton, which achieves a speedup of up to 3.3$\times$ over FlashAttention-3 at inference time. Overall, DashAttention offers a cost-effective strategy to model long contexts.[2]

1 Introduction

The complexity of long-context tasks is impacted by the quantity of information to be retrieved, its obfuscation (i.e., whether it is distinguishable from noise), and its distribution throughout the context (e.g., scattered or concentrated) [1, 2]. Hence, to achieve good performance in such tasks, models must remain *selective* enough to ignore irrelevant tokens, but also *sufficiently flexible* to recover the positions that matter for a given query, irrespective of their number, position, or similarity to the rest of the context. Dense softmax attention [3] handles the second requirement but fails the first, since every visible token receives nonzero probability mass, which can cause dispersion [4]. Hard sparse routing methods such as top-k block selection [5, 6] address the first requirement; however, they do so by imposing a fixed budget, sacrificing the second requirement, and by severing the differentiable path between coarse routing decisions and fine-grained token-level attention. Therefore, achieving both query-dependent flexibility and stringent token-level selectivity remains an open challenge.

This paper proposes DashAttention (Differentiable and Adaptive Sparse Hierarchical Attention), a multi-stage attention mechanism designed to meet both requirements. Instead of routing with a hard top-k operator over block scores, we route with α-entmax [7], an adaptive sparse distribution

*Equal contribution.
[2]Code available at https://github.com/fasa-org/dash-attention.

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

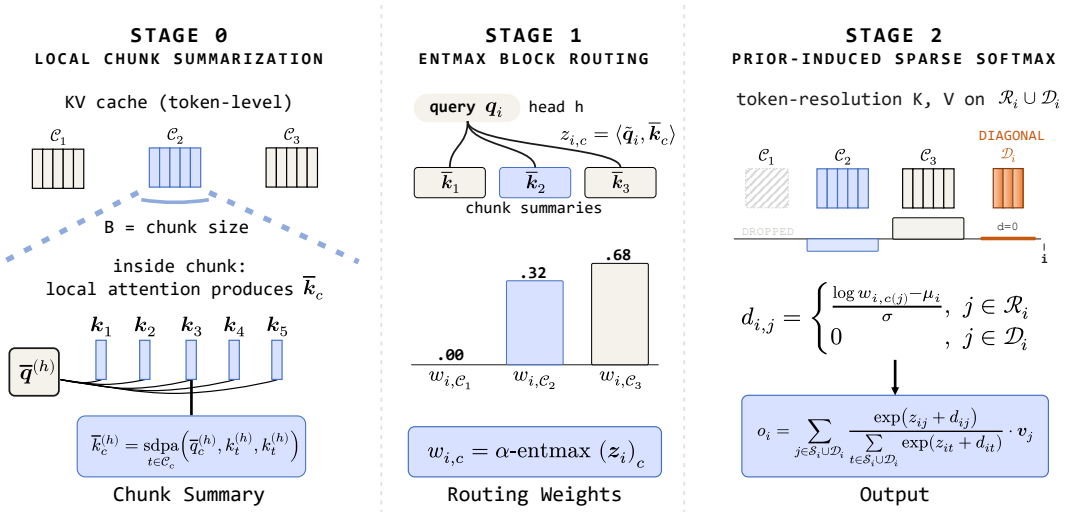


Figure 1: High-level overview of DashAttention. Stage 0 builds chunk summaries by local SDPA; Stage 1 routes with α-entmax to obtain an adaptively sparse support; Stage 2 refines at token resolution with a softmax whose logits are biased by $d_{i,j}$, derived from the router weights, maintaining full differentiability and FlashAttention compatibility. See Section 3 for details.

whose support is learned from the input itself and whose nonzero masses remain differentiable. The chunk-level probabilities are then consumed by a second online-softmax refinement stage that operates at full resolution, but only on tokens inside the routed chunks. This way, the model learns *where* and *how much* to look at coarsely, and *what* to read precisely, while keeping the entire hierarchy differentiable end-to-end.

This design is motivated by both theory and practice. On the theory side, recent work has shown that softmax full attention can be dispersive in long contexts, whereas sparse alternatives such as α-entmax can preserve concentration and improve long-context capabilities [4, 8]. On the systems side, modern sparse attention methods almost always rely on a hierarchical decomposition, since selecting tokens based directly on attention scores is too expensive: the query-key multiplications must still be computed, leaving little opportunity to reduce computation [5, 6, 9]. Our key intuition is to combine these two ideas into a single hierarchical attention mechanism that is inherently adaptive, fully differentiable, and efficient. Our main contributions are as follows:

- We analyze the limitations of top-k-based sparse attentions and introduce DashAttention, an end-to-end differentiable method that adaptively allocates sparsity across attention heads.
- We integrate DashAttention into long-context continual pretraining and show that it outperforms existing hierarchical sparse attention methods while matching full-attention accuracy.
- We present an efficient GPU-aware implementation of DashAttention that achieves speedups of 3.36× and 1.35× over FlashAttention-3 and InfLLMv2, respectively.

2 Background

Standard scale dot-product attention. Given a set of matrices $\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n\times d}$ containing d-dimensional representations for n queries, keys, and values, the *scale dot-product self-attention* (SDPA) at a single head is computed as follows [3]:

$$\boldsymbol{Z} = \frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d}} \in \mathbb{R}^{n\times n}, \quad \boldsymbol{P} = \pi\left(\boldsymbol{Z} + \boldsymbol{M}\right), \quad \boldsymbol{O} = \boldsymbol{P}\boldsymbol{V} \in \mathbb{R}^{n\times d}, \tag{1}$$

where $\boldsymbol{M}$ is the attention mask and $\pi : \mathbb{R}^n \to \triangle_n$ is a transformation applied row-wise that maps logits to vectors in the n-dimensional probability simplex $\triangle_n := \{\boldsymbol{p} \in \mathbb{R}^n : \mathbf{1}^\top \boldsymbol{p} = 1, \boldsymbol{p} \geq \mathbf{0}\}$, with softmax being the most common choice. FlashAttention [10] efficiently implements Eq. (1) by reducing memory to $\mathcal{O}(n)$ while preserving softmax semantics.

Adaptively sparse attention: α-entmax. Softmax attention is inherently dense, as every key receives a positive weight. A principled *differentiable* sparse alternative is the α-entmax function [7],

$$\alpha\text{-entmax}(\boldsymbol{s}) = [(\alpha - 1)\boldsymbol{s} - \tau\mathbf{1}]_+^{\frac{1}{\alpha-1}}, \tag{2}$$

where $[\cdot]_+$ is the ReLU function, and $\tau \in \mathbb{R}$ is the (unique) normalizing constant which ensures the output is a valid probability distribution, $\sum_i \alpha\text{-entmax}(\boldsymbol{s})_i = 1$. Importantly, for $\alpha > 1$, α-entmax can produce sparse probability vectors. When $\alpha \to 1$, α-entmax recovers softmax, whereas $\alpha = 2$ leads to sparsemax [11]; the sparsity increases monotonically with α. This means that coordinates with $(\alpha - 1)s_i \leq \tau$ become exactly zero. As such, α-entmax yields dynamic sparsity, where the pattern of zeros (as well as its number) depends on the input $\boldsymbol{s}$. Efficient GPU implementations for α-entmax attention have been developed in AdaSplash [12, 13]. More information on the α-entmax transformation, including the distinction to top-k softmax, is provided in App. A.

Hierarchical trainable sparse attention. Recently proposed pretraining-compatible sparse attention methods [5, 6, 9] typically adopt a coarse-to-fine hierarchical design: the context is divided into blocks or pages, summarized by lightweight representations, routed via top-k coarse relevance scores, and followed by exact token-level SDPA within the selected KV segments. However, their fixed top-k selection overlooks the heterogeneous information demands across tokens, attention heads, and layers. As shown in Sec. 5.4, DashAttention dynamically allocates sparsity across tokens and heads, which is crucial for improving performance under a fixed sparsity budget.

3 DashAttention

Motivation. The key innovation of DashAttention is that the coarse router is itself a sparse attention mechanism—specifically, α-entmax from Eq. (2)—rather than a top-k truncation applied to dense scores. Since α-entmax is fully differentiable, the hierarchical attention remains end-to-end trainable, while ensuring a level of sparsity adaptive to each specific token sequence. Hence, DashAttention is built around two complementary inductive biases: α-entmax is used to suppress irrelevant chunks, whereas softmax is used to obtain a relative ranking inside small selected regions. The resulting attention is therefore *sparse* where sparsity is computationally useful, and *dense* where density is semantically meaningful. This combination also matters for optimization. In methods based on top-k block routing, the budget k itself becomes a central hyperparameter, and the support changes through a discontinuous top-k map; thus, gradients do not directly capture how routing scores should change block membership. In DashAttention, by contrast, the support is learned from the data, the coarse masses are differentiable, and the refinement stage preserves a direct gradient path from the final loss to the block-summarization (Stage 0) and block-routing (Stage 1) parameters.

Problem setup and notation. Assume the model has model dimension d and head dimension d_h. We let h_q and h_{kv} denote the number of query and key-value heads, respectively. When using GQA [14], we denote the query-group of the r^{th} KV head as $\mathcal{G}_r$, and the grouping factor as g_q. We partition the prefix into contiguous chunks of size B, and denote the index set of chunk c by $\mathcal{C}_c$. For a query position i, DashAttention proceeds in three stages, illustrated in Fig. 1: (Stage 0) we first summarize each chunk into a compact representation using local attention driven by a learned summarization head; (Stage 1) the query attends to these chunk summaries with α-entmax, producing a sparse routing distribution; (Stage 2) only the routed chunks are expanded back to token resolution and refined with softmax. We explain each stage next.

3.1 Stage 0: Local Chunk Summarization

Existing methods use mean pooling [6, 9] or a learned MLP [5] to summarize each KV chunk. However, they are either less expressive or hard to adapt from a pretrained model. To mitigate that, we introduce a local-attention mechanism by using a learned query vector $\bar{\boldsymbol{q}} \in \mathbb{R}^{h_{kv} \times d_h}$ initialized at zero. As a result, the model begins from a conservative mean-pooling regime and only gradually learns more expressive chunk summaries. During inference, we compute a summary vector for each chunk by a local SDPA as follows, where $r \in \{1, \ldots, h_{kv}\}$ indexes the key-value head:

$$\bar{\boldsymbol{k}}_c^{(r)} = \sum_{t \in \mathcal{C}_c} \frac{\exp\left(\langle \bar{\boldsymbol{q}}^{(r)}, \boldsymbol{k}_t^{(r)} \rangle / \sqrt{d_h}\right)}{\sum_{u \in \mathcal{C}_c} \exp\left(\langle \bar{\boldsymbol{q}}^{(r)}, \boldsymbol{k}_u^{(r)} \rangle / \sqrt{d_h}\right)} \boldsymbol{k}_t^{(r)}, \tag{3}$$

This design is more expressive than mean pooling used in MoBA and InfLLMv2, as it can be interpreted as a learned summary head that reads each chunk from within the chunk itself. It is also easier to adapt compared with MLPs used in NSA. At the start of training, the inner products in Eq. (3) are zero, causing the local softmax in Eq. (3) to reduce to uniform mean pooling. This enables a smooth transition from mean pooling to weighted average of the key vectors within the chunk. Notably, chunk summaries become fixed once a chunk has been fully generated, since the $\bar{\boldsymbol{k}}_c$ only depends on $\boldsymbol{k}$'s from positions in $\mathcal{C}_c$. Therefore, this design also guarantees no additional recomputation during inference, aligned with previous chunk summarization methods. Ablation studies on local attention are conducted in App. F.5.

3.2 Stage 1: Entmax Chunk Routing

Given each query head h associated with key-value head $r = \lfloor h/g_q \rfloor$, we compute chunk-level logits $\bar{z}_{i,c}^{(h)} = \langle \boldsymbol{q}_i^{(h)}, \bar{\boldsymbol{k}}_c^{(r)} \rangle / \sqrt{d_h}$ for all causal chunks c visible from position i. These logits are then transformed into a sparse chunk distribution with α-entmax with a scaling factor γ:

$$\hat{\boldsymbol{w}}_i^{(h)} = \alpha\text{-entmax}\left(\gamma \bar{\boldsymbol{z}}_i^{(h)}\right) \in \triangle_{\lfloor n/B \rfloor}, \quad h \in \{1, 2, \cdots, h_q\}. \tag{4}$$

The support $\hat{\mathcal{S}}_i^{(h)} = \{c \mid w_{i,c}^{(h)} > 0\}$ determines chunk positions that are retained for the next stage, while other chunks are discarded. Because routing is performed with α-entmax rather than top-k, the support is not determined by a fixed budget. Instead, it is induced by the geometry of the chunk scores themselves. As a result, informative queries may route to many chunks, while sharp queries may route to very few, which enables dynamic sparsity allocation among different tokens, heads, and layers which is one of the defining advantages of DashAttention over fixed-budget sparse attention.

We note that GQA is commonly used in modern LLMs. Therefore, when $h_q > h_{kv}$, multiple query heads share the same key-value head, so Stage 1 must produce a shared routed support for each GQA group. In DashAttention, for each group $r \in \{1, \cdots, h_{kv}\}$ with member query heads $\mathcal{G}_r$, we average the head-wise probability scores: $\boldsymbol{w}_i^{(r)} = \sum_{h \in \mathcal{G}_r} \hat{\boldsymbol{w}}_i{}^{(h)}/g_q \in \triangle_{\lfloor n/B \rfloor}$, which remains sparse and with a support $\mathcal{S}_i^{(r)} = \cup_{h \in \mathcal{G}_r} \hat{\mathcal{S}}_i^{(h)}$ determined jointly by the chunk-score geometry within $\mathcal{G}_r$.

3.3 Stage 2: Prior-Induced Sparse Softmax Attention

Here, we derive the final stage of DashAttention. As mentioned in Sec. 1, our goal is to make DashAttention both adaptively sparse and readily adaptable from softmax attention. To this end, we start from the variational form of softmax [15]:

$$\text{softmax}(\boldsymbol{z}) = \underset{\boldsymbol{p} \in \triangle_n}{\arg\max}\, \boldsymbol{z}^\top \boldsymbol{p} - \text{KL}(\boldsymbol{p} \,\|\, \boldsymbol{u}), \tag{5}$$

where $\boldsymbol{u}$ denotes an uniform probability vector and $\boldsymbol{z}$ the attention logits where $z_{i,j} = \langle \boldsymbol{q}_i, \boldsymbol{k}_j \rangle / \sqrt{d_h}$ and we omit the notation for attention heads for simplicity. Instead, we carry the knowledge from Stage 1 ($\boldsymbol{w}$) by varying the reference distribution in the KL term from $\boldsymbol{u}$ to another distribution $\boldsymbol{g} \in \triangle_n$, which is obtained from the block routing scores $\boldsymbol{w}$ via $\boldsymbol{g} = g_\sigma(\boldsymbol{w})$ and described in detail in Sec. 3.4. We therefore obtain a token-grained sparse probability distribution that enables a block-sparse differentiable softmax computation. Given the Stage 1 scores of the i^{th} token $\boldsymbol{w}_i$, we compute the attention probability at the i^{th} token as follows:

$$\boldsymbol{p}_i^{\text{route}} = \underset{\boldsymbol{p} \in \triangle_n}{\arg\max}\, \boldsymbol{z}_i^\top \boldsymbol{p} - \text{KL}\left(\boldsymbol{p} \,\|\, g_\sigma(\boldsymbol{w}_i)\right), \quad p_{i,j}^{\text{route}} = \frac{g_{i,j} \exp(z_{i,j})}{\sum_{t \le i} g_{i,t} \exp(z_{i,t})}, \tag{6}$$

where $g_{i,j} = g_\sigma(\boldsymbol{w}_i)_j$. By requiring $g_{i,j} = 0$ whenever $w_{i,j} = 0$ with an appropriate construction of g_σ, the resulting softmax attention becomes naturally sparse, since if $g_{i,j} \to 0^+$, $-\text{KL}(p_j \| g_{i,j}) \to +\infty$ for any $p_j > 0$, thereby masking Stage 2's softmax probability with an entmax-computed dynamic mask. At the same time, the properties of dynamic sparsity and full differentiability are preserved through the entmax score $\boldsymbol{w}$. The final i^{th} output becomes

$$\boldsymbol{o}_i = \boldsymbol{V}^\top \boldsymbol{p}_i^{\text{route}} = \sum_{j \in \mathcal{S}_i^{\text{route}}} \frac{g_{i,j} \exp(z_{i,j})}{\sum_{t \in \mathcal{S}_i^{\text{route}}} g_{i,t} \exp(z_{i,t})} \cdot \boldsymbol{v}_j, \tag{7}$$

where $\mathcal{S}_i^{\text{route}} = \{c \mid g_{i,c} > 0\}$ is shared within the same query group to enable similar matrix multiplication tiling layout with NSA [5] and InfLLMv2 [6]. Note that this form is readily compatible with existing efficient attention kernels. In practice, we employ AdaSplash-2 [13] for Stage 1 (α-entmax attention) and FlashAttention [16] for Stage 2 (softmax attention), enabling efficient training and inference. Moreover, Eq. (7) is fully differentiable with respect to $\boldsymbol{q}$, $\boldsymbol{k}$, $\boldsymbol{v}$, and the prior score $\boldsymbol{g}$. As a result, in DashAttention gradients can propagate back to Stage 1's α-entmax scores $\boldsymbol{w}$, thereby making the local chunk summarization from Stage 0 trainable.

3.4 Controlling Prior Strength and Handling Diagonal Chunks

Since the diagonal or near-diagonal region may contain fewer than B tokens, the Stage 1 priors $\boldsymbol{g} = g_\sigma(\boldsymbol{w})$ cannot be obtained due to the absence of block summaries. At the same time, replacing $\mathrm{KL}(\boldsymbol{p} \,\|\, \boldsymbol{u})$ with $\mathrm{KL}(\boldsymbol{p} \,\|\, \boldsymbol{g})$ would drift the outcome $\boldsymbol{p}_i^{\text{route}}$ to a coarsely estimated $\boldsymbol{g}$, since the summary vectors provide only an approximation of each block. Therefore, to properly estimate $\boldsymbol{g}$, we split the attention mass of $\boldsymbol{w}$ into two branches: a routed branch $\mathcal{R}_i$ over blocks selected in Stage 1, and a diagonal branch $\mathcal{D}_i$ that requires prior estimation. To control the prior strength, we introduce a hyperparameter σ and construct a strength-reduced prior probability as follows:

$$w'_{i,j} = \frac{1}{B} \cdot \frac{w_{i,c(j)}^{1/\sigma}}{\mathbf{1}^\top \boldsymbol{w}_i^{1/\sigma}}, \tag{8}$$

where $c(j)$ represents the chunk associated with the j^{th} element. As $\sigma \to \infty$, the prior gradually becomes uniform over the routed support: $w'_{i,j} \to 1/|\mathcal{R}_i|$ for $j \in \mathcal{R}_i$, while $w'_{i,j} = 0$ for $j \notin \mathcal{S}_i$, thereby corresponding to only applying softmax on selected tokens without any additional prior from Stage 1. With that in mind, we define the prior transformation $g_\sigma : \triangle_{\lfloor n/B \rfloor} \to \triangle_n$ as follows:

$$g_\sigma(\boldsymbol{w}_i)_j = g_{i,j} = \begin{cases} \lambda_i w'_{i,j}\,, & j \in \mathcal{R}_i, \\ \frac{1}{|\mathcal{D}_i|}(1-\lambda_i)\,, & j \in \mathcal{D}_i, \\ 0, & \text{otherwise.} \end{cases} = \begin{cases} \frac{\lambda_i w_{i,c(j)}^{1/\sigma}}{B \cdot \mathbf{1}^\top \boldsymbol{w}_i^{1/\sigma}}\,, & j \in \mathcal{R}_i, \\ \frac{1}{|\mathcal{D}_i|}(1-\lambda_i)\,, & j \in \mathcal{D}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

Note that $g_\sigma(\boldsymbol{w}_i)$ is a probability vector and adds up to one. By this construction, the prior assigns probability mass λ_i to $\mathcal{R}_i$ and $1-\lambda_i$ to $\mathcal{D}_i$. In practice, we choose λ_i as a function of how informative the routing distribution is relative to a uniform baseline. Let $\boldsymbol{u}_{\mathcal{R}_i}$ be the uniform distribution over the routed support and $\boldsymbol{w}'_{\mathcal{R}_i}$ denotes the probabilities of $\boldsymbol{w}'_i$ restricted to $\mathcal{R}_i$, then:

$$\lambda_i = \mathrm{sigmoid}\left(\mathrm{KL}(\boldsymbol{u}_{\mathcal{R}_i} \,\|\, \boldsymbol{w}'_{\mathcal{R}_i}) + \log \frac{|\mathcal{R}_i|}{|\mathcal{D}_i|} \right), \tag{10}$$

This has an intuitive interpretation: if the block router is nearly uniform ($\mathrm{KL} \approx 0$), the model allocates mass roughly proportional to $|\mathcal{R}_i|/(|\mathcal{R}_i| + |\mathcal{D}_i|)$, whereas if the router is highly informative (large KL), more mass is given to the routed branch We also have $\lim_{\sigma \to +\infty} \lambda_i = |\mathcal{R}_i|/(|\mathcal{R}_i| + |\mathcal{D}_i|)$, showing that the priors can be weakened by utilizing a larger σ. Next, in Proposition 1, we show a straightforward way to compute Eq. (9) in softmax attention. The proof is given in App. D.

Proposition 1. *Computing the attention output $\boldsymbol{o}_i$ with Eq. (7) and Eq. (9) is equivalent to first calculate $\mu_i = \mathrm{mean}_{j \in \mathcal{R}_i}\{\log w_{i,c(j)}\}$, followed by*

$$d_{i,j} = \begin{cases} \frac{\log w_{i,c(j)} - \mu_i}{\sigma}\,, & j \in \mathcal{R}_i, \\ 0\,, & j \in \mathcal{D}_i. \end{cases} \qquad \boldsymbol{o}_i = \sum_{j \in \mathcal{R}_i \cup \mathcal{D}_i} \frac{\exp(z_{i,j} + d_{i,j}) \cdot \boldsymbol{v}_j}{\sum_{t \in \mathcal{R}_i \cup \mathcal{D}_i} \exp(z_{i,t} + d_{i,t})}.$$

3.5 GPU-Aware Implementation

We implement DashAttention in Triton [17], consisting of three fused kernels that serve both prefill and incremental decoding, backed by a chunk-representation cache in HBM (High Bandwidth Memory). The three fused kernels (one per stage) are described below, with complete algorithms deferred to App. G.

Stage 0 kernel. A single Triton kernel lets the learned per-head summary token $\bar{\boldsymbol{q}}^{(r)}$ attend to its keys via an online softmax. Because at this stage the keys are also used as values, the same loaded $\boldsymbol{K}$ tile will serve both purposes, so the chunk representation $\bar{\boldsymbol{k}}_c^{(r)}$ in Eq. (3) is produced in a single pass with no extra HBM round-trip. Once a chunk is complete, it is written back, persisted in the chunk representation cache, and reused across all subsequent query positions and decoding steps.

Stage 1 kernel. The chunk logits $\boldsymbol{z}_i^{(h)}$ are formed by attending to the cached chunk representations. Since the number of chunks $T_c = \lfloor n/B \rfloor$ is small (e.g., $T = 256$ for $n = 16$K with $B = 64$), the entire chunk-score row typically fits in registers, and the entmax threshold τ is solved in place by the AdaSplash-2 solver [13]. For GQA, we then average the entmax probabilities across each group's query heads before pruning, keeping the support consistent within the KV group. The sparse support $\mathcal{S}_i$ is stored as a bitpacked block mask $\boldsymbol{M}_i \in \{0,1\}^{h_{kv} \times T_c}$, packing 32 consecutive column chunks per `int32`, while the per-chunk routing bias $d_{i,j}$ is materialized alongside for use in Stage 2.

Stage 2 kernel. A masked FlashAttention kernel iterates over the chunks marked active in $\boldsymbol{M}$, walking the bitpacked mask. For each selected chunk, we add the per-chunk routing bias to the attention logits before the online softmax, in a single fused pass over $\mathcal{S}_i$, keeping the kernel fully compatible with FlashAttention. For incremental decoding, a split-KV variant partitions $\boldsymbol{M}$ across thread blocks along the KV dimension, exposing enough parallelism to saturate the GPU for a single query [10]. For training, the backward pass can reuse $\boldsymbol{M}$ thanks to the sparse α-entmax Jacobian [7].

4 Theoretical Analysis

As shown in [8, 12, 13], softmax attention suffers from *dispersion* in long-context settings: the Shannon entropy of its attention distribution $\mathcal{H}(\boldsymbol{p})$ grows as $\lim_{n\to\infty} \frac{\mathcal{H}(\boldsymbol{p})}{\log n} = 1$ with sequence length n, making long-range dependency modeling increasingly difficult. Top-k sparse attention mitigates this issue by bounding the entropy by $\log k$. However, existing hierarchical sparse attention methods [5, 6] combine GQA with post-softmax head aggregation before top-k selection, allowing dispersion to re-emerge during aggregation. We formalize this property as follows:

Definition 1 (Head aggregation)**.** *Given $f : \mathbb{R}^n \to \triangle_n$, and H bounded sequence $\{\boldsymbol{z}^{(h)}\}_{h=1}^H \subset \mathbb{R}^n$, and the aggregation weights $\theta = (\theta_1, \cdots, \theta_H)^\top \in \triangle_H$, the aggregated probability is given by* $\mathrm{aggr}_f\left(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(H)}; \boldsymbol{\theta}\right) = \sum_{h=1}^H \theta_h \cdot f\left(\boldsymbol{z}^{(h)}\right) \in \triangle_n$.

Thus, softmax head aggregation is dispersive by construction, whereas the entmax head aggregation in DashAttention is non-dispersive.

Theorem 1 (Dispersion in head aggregation, Informal)**.** *Under any finite H and $\theta \in \triangle_H$:*

1. Softmax head aggregation, i.e., $f = \mathrm{softmax}$, is dispersive;

2. Denote $\boldsymbol{p}^{(h)} = \alpha\text{-entmax}(\boldsymbol{z}^{(h)})$. If $\|\boldsymbol{p}^{(h)}\|_0 = \mathcal{O}\left(n^{\beta_h}\right)$ with $\beta_h \in (0,1)$, then entmax head aggregation, i.e., $f = \alpha\text{-entmax}$, is not dispersive.

Since softmax head aggregation is dispersive, it can undermine the non-dispersive property of top-k sparse attention and lead to noisy selection. In contrast, DashAttention performs entmax head aggregation directly on sparse entmax scores, mitigating this issue and perform better on challenging retrieval tasks, such as MK1–MK3 in Table 1. We provide a detailed proof in App. E.

5 Experiments

We comprehensively evaluate DashAttention from multiple perspectives. First, we assess its task performance across representative benchmarks (Sec. 5.1). Second, we analyze its computational efficiency (kernel inference speed) under different inference settings (Sec. 5.2). Third, we examine the effectiveness–efficiency tradeoff induced by different sparsity configurations and compare the resulting Pareto frontiers against relevant baselines (Sec. 5.3). Finally, we study the dynamic sparsity allocation behavior of DashAttention through ablations to better understand how it distributes computation across layers (Sec. 5.4).

Table 1: Results on RULER with a 16K context length. Best results in bold.

	Method	RULER-16K													Avg. ↑ (%)	Sparsity (%)
		SG1	*SG2*	*SG3*	*MK1*	*MK2*	*MK3*	*MV*	*MQ*	*VT*	*CWE*	*FWE*	*QA1*	*QA2*		
1B	FullAttn	100.0	100.0	100.0	98.0	78.0	48.0	87.0	89.0	17.6	1.8	66.7	44.0	36.0	66.6	0.0
	NSA	98.0	92.0	86.0	60.0	20.0	6.0	56.0	53.5	25.6	**1.6**	60.7	32.0	**36.0**	48.3	75.0
	InfLLMv2	**100.0**	**100.0**	**100.0**	92.0	52.0	16.0	**94.0**	**90.0**	40.8	1.0	59.3	**36.0**	30.0	62.4	75.0
	DashAttention	**100.0**	**100.0**	96.0	**94.0**	**70.0**	**26.0**	89.5	87.5	**46.0**	0.8	**63.3**	**36.0**	34.0	**64.9**	**75.7**
3B	FullAttn	100.0	100.0	98.0	100.0	94.0	60.0	96.0	96.5	20.4	5.8	19.3	60.0	50.0	69.2	0.0
	NSA	**100.0**	92.0	90.0	66.0	24.0	10.0	64.5	62.0	19.2	4.2	28.7	36.0	40.0	49.0	75.0
	InfLLMv2	**100.0**	**100.0**	98.0	94.0	56.0	28.0	95.5	**97.0**	**22.8**	**5.0**	16.0	**56.0**	**48.0**	62.8	75.0
	DashAttention	**100.0**	**100.0**	**100.0**	**100.0**	**88.0**	**40.0**	**96.5**	95.5	21.2	1.6	**42.7**	52.0	42.0	**67.7**	**75.4**
8B	FullAttn	100.0	100.0	100.0	98.0	100.0	96.0	100.0	99.5	58.4	49.2	78.0	78.0	52.0	85.3	0.0
	NSA	**100.0**	82.0	68.0	60.0	34.0	12.0	73.5	74.5	30.0	42.8	68.7	36.0	34.0	55.0	75.0
	InfLLMv2	**100.0**	**100.0**	**100.0**	**98.0**	82.0	52.0	96.0	98.0	55.6	42.4	**77.3**	66.0	**58.0**	78.9	75.0
	DashAttention	**100.0**	**100.0**	**100.0**	**98.0**	**96.0**	**86.0**	**100.0**	**99.0**	**60.0**	**49.6**	72.0	**72.0**	54.0	**83.6**	**75.7**

Table 2: Results on RULER with 8B models at longer context lengths. Best results in bold.

Method	*SG1*	*SG2*	*SG3*	*MK1*	*MK2*	*MK3*	*MV*	*MQ*	*VT*	*CWE*	*FWE*	*QA1*	*QA2*	Avg. ↑ (%)	Sparsity (%)
RULER-32K															
FullAttn	100.0	100.0	98.0	96.0	48.0	24.0	63.5	79.0	40.8	1.0	73.3	28.0	34.0	60.4	0.0
NSA	94.0	46.0	40.0	26.0	2.0	0.0	31.0	25.5	25.6	**4.2**	58.0	**24.0**	24.0	30.8	87.5
InfLLMv2	**100.0**	92.0	**94.0**	62.0	**20.0**	**4.0**	52.5	56.5	36.8	1.0	**74.0**	22.0	**28.0**	49.5	87.5
DashAttention	**100.0**	**98.0**	88.0	**84.0**	16.0	2.0	**76.5**	**67.5**	**44.0**	0.8	67.3	20.0	26.0	**53.1**	**88.4**
RULER-64K															
FullAttn	100.0	100.0	90.0	84.0	26.0	0.0	29.5	46.5	62.0	0.4	61.3	18.0	24.0	49.4	0.0
NSA	**100.0**	32.0	32.0	22.0	2.0	0.0	13.0	15.0	19.2	0.2	73.3	28.0	24.0	27.8	90.6
InfLLMv2	**100.0**	**92.0**	**86.0**	50.0	**4.0**	0.0	22.0	32.5	**46.4**	**0.4**	**78.7**	26.0	24.0	43.2	90.6
DashAttention	**100.0**	**92.0**	80.0	**54.0**	**4.0**	0.0	**41.0**	**62.5**	38.0	0.2	59.3	**34.0**	**30.0**	**45.8**	**91.6**
RULER-128K															
FullAttn	100.0	98.0	72.0	78.0	10.0	0.0	30.5	33.0	0.4	0.2	61.3	12.0	18.0	39.5	0.0
NSA	92.0	12.0	4.0	8.0	0.0	0.0	7.0	10.0	30.4	**0.4**	**70.0**	18.0	14.0	20.5	95.1
InfLLMv2	**100.0**	50.0	46.0	32.0	**4.0**	0.0	16.5	20.0	2.4	0.2	68.0	12.0	**22.0**	28.7	95.1
DashAttention	**100.0**	**70.0**	**58.0**	**60.0**	2.0	0.0	**34.5**	**42.5**	**40.8**	0.0	62.7	**20.0**	16.0	**39.0**	**95.3**

5.1 Task Performance

Baselines and Experimental Setup. Besides full attention (FullAttn), we compare DashAttention with other GQA-based hierarchical softmax attention methods under long-context adaptation, namely NSA [5] and InfLLMv2 [6], and follow the same long-context adaptation setting for all methods. Concretely, we start with models pretrained with softmax full attention, continue pretraining them on 16K long-context data (InfLLMv2-5B dataset [18]), and then apply a short SFT stage (MiniCPM-4 Dataset [19]). We use the 1B, 3B, and 8B variants of MiniCPM-4 [19] as base models. During training, we gradually increase α from 1.25 to 1.5, and use $\alpha = 1.5$ for inference. For NSA and InfLLMv2 baselines, we keep the sparsity to 75% during training. Full details are in App. B.1.1.

Benchmark Datasets. We evaluate long-context performance on RULER [20] and HELMET [21]. Given the importance of matching sparsity ratios [2], for NSA and InfLLMv2, we set the attention sparsity to 75% during both training and inference, whereas for DashAttention we tune the Stage 1 factor γ to match a comparable sparsity level. To assess general capabilities, we further evaluate on MMLU [22], MMLU-Pro [23], CSQA [24], IFEval [25], HellaSwag [26], GSM8K [27], MATH [28], DROP [29], MBPP [30], and HumanEval [31]. Additional details are provided in App. B.1.2.

Long-Context Performance. Table 1 (RULER) and Table 3 (HELMET) show that DashAttention consistently outperforms NSA and InfLLMv2 in terms of both performance and sparsity. In particular, results on RULER and the Recall task in HELMET demonstrate that DashAttention exhibits a significantly stronger capability in challenging long-context retrieval tasks. Moreover, DashAttention achieves better performance than the baselines across multiple model sizes and evaluation tasks. DashAttention also sustains this advantage at longer contexts: Table 2 evaluates

Table 3: Results on HELMET with a 16K context length. Best results in bold.

	Method	HELMET-16K							Overall ↑ (%)	Sparsity (%)
		Recall	ICL	Rerank	RAG	LongQA	Cite	Summ.		
1B	FullAttn	66.6	59.0	13.9	34.5	40.1	8.9	4.2	32.5	0.0
	NSA	23.4	46.0	15.0	35.0	34.6	6.4	4.4	23.6	75.0
	InfLLMv2	43.7	59.2	15.5	37.0	**38.7**	**7.6**	4.9	29.5	75.0
	DashAttention	**51.8**	**61.0**	**16.5**	**38.5**	36.4	7.4	**6.7**	**31.2**	**75.4**
3B	FullAttn	78.6	44.8	25.3	48.5	42.5	12.3	9.7	37.4	0.0
	NSA	29.9	45.2	20.5	45.5	39.7	7.3	7.1	27.9	75.0
	InfLLMv2	58.0	44.2	**24.8**	**49.3**	**47.7**	6.0	**9.8**	34.2	75.0
	DashAttention	**66.9**	**51.2**	20.0	47.8	35.9	**10.3**	8.3	**34.3**	**75.4**
8B	FullAttn	98.3	56.0	37.8	53.3	52.4	20.0	16.5	47.7	0.0
	NSA	35.0	59.8	27.9	48.3	51.0	13.7	**14.7**	35.8	75.0
	InfLLMv2	81.4	**63.6**	36.8	**54.0**	**53.2**	**18.3**	13.7	45.9	75.0
	DashAttention	**88.3**	62.8	**39.7**	**54.0**	52.3	17.4	13.7	**46.9**	**75.4**

Table 4: General task performance with 8B models. Best results in bold.

Method	MMLU	MMLU-Pro	CSQA	IFEval	HellaSwag	GSM8K	MATH	DROP	MBPP	HumanEval	Avg.
FullAttn	73.6	46.1	80.0	80.0	72.9	76.3	22.2	11.3	63.0	69.5	59.5
InfLLMv2	73.2	45.6	**80.6**	79.0	72.8	**77.1**	22.2	10.3	62.4	67.7	59.1
NSA	73.2	**46.3**	80.4	80.0	72.5	75.1	**22.4**	10.5	61.6	**70.1**	59.2
DashAttention	**73.5**	46.2	80.3	**80.7**	**73.1**	76.8	22.2	**10.7**	**63.0**	67.7	**59.4**

the 8B model at 32K, 64K, and 128K, where DashAttention achieves the best average accuracy and the highest sparsity among the sparse methods at every length, with the largest gains on the retrieval-oriented tasks MV, MQ, and VT. For these evaluations, we extrapolate the context window using LongRoPE [32] with a $4\times$ factor for 32K and an $8\times$ factor for 64K and 128K. We further show in Table 5 that DashAttention can also inference with softmax full attention.

DashAttention with Softmax Decoding. To close the gap with standard full softmax attention, we first investigate the impact of the chunk size in Stage 0 in App. F.2 (Table 9), where we find that reducing chunk size naturally leads to better results. However, this also reduces efficiency, as if chunk size is 1 (no chunking), DashAttention reverts to α-entmax attention followed by softmax attention, and thus it incurs the total cost of both at inference time. An alternative is to simply use softmax at test time without Stage 0 and Stage 1, where highly optimized implementations already exist (e.g., in vLLM [33] and SGLang [34]). Table 5 reports the results of training with DA and inference with FA, showing even higher performance than FA-trained models.

Table 5: Long-context performance of DashAttention (DA) models with softmax full attention (FA).

	RULER		HELMET	
	FA	DA+FA	FA	DA+FA
1B	66.6	**70.4**	32.5	**33.8**
3B	69.2	**71.9**	**37.4**	37.0
8B	85.3	**86.7**	47.7	**48.7**

General Task Performance. We further evaluate DashAttention and the baselines on short-context general tasks to verify that DashAttention does not notably degrade short-context performance. During inference, both DashAttention and InfLLMv2 use softmax full attention, while NSA uses compressed attention without sparsity. As shown in Table 4, DashAttention achieves performance on par with the original Full Attention model, and slightly outperforms NSA and InfLLMv2.

Compatibility with Hybrid Architectures. We further verify that DashAttention transfers to other architectures. We conduct an 8K context length continual pretraining on Tiny-Aya-Base [35], a 4B hybrid model with interleaved sliding-window attention (SWA) and full attention layers and parallelized FFN, for 10B tokens. DashAttention and InfLLMv2 are applied to the full attention layers while we keep the SWA layers untouched, and we evaluate the trained

Table 6: RULER-8K results on Tiny-Aya-Base. Best results in bold.

Met.	*S*	*MK*	*MV*	*MQ*	*VT*	*WE*	*QA*	Avg. ↑ (%)
FA	99.0	92.3	91.0	97.0	94.4	44.0	61.5	82.1
Iv2	99.0	82.3	**96.3**	**96.8**	**95.4**	43.3	59.0	79.8
DA	**99.7**	**89.7**	91.3	**96.8**	87.0	**45.6**	**61.5**	**81.3**

Table 7: Wall-clock speedup over full attention with FlashAttention (higher is better). Columns are chunk sparsity s, the fraction of chunks not routed to Stage 2; the corresponding top-k is $(1-s)\cdot N/64$. Iv2 denotes InfLLMv2 and DA denotes DashAttention. Best results in bold.

	Prefill									Decoding								
	$s = 75\%$			$s = 87.5\%$			$s = 93.7\%$			$s = 75\%$			$s = 87.5\%$			$s = 93.7\%$		
N	NSA	Iv2	DA	NSA	Iv2	DA	NSA	Iv2	DA	NSA	Iv2	DA	NSA	Iv2	DA	NSA	Iv2	DA
16K	0.71	0.96	**1.34**	1.23	1.63	**2.09**	1.91	2.40	**2.87**	0.40	1.20	**1.54**	0.43	1.43	**1.67**	0.46	1.73	**1.93**
32K	0.76	1.04	**1.39**	1.25	1.67	**2.11**	2.19	2.78	**2.94**	0.57	1.46	**1.69**	0.65	1.85	**2.26**	0.73	2.29	**2.69**
64K	0.82	1.01	**1.55**	1.33	1.77	**2.28**	2.24	2.86	**3.02**	0.71	1.59	**1.97**	0.95	2.28	**2.69**	1.12	2.92	**3.27**
96K	0.82	1.04	**1.63**	1.31	1.73	**2.34**	2.32	3.06	**3.09**	0.80	1.73	**1.96**	1.04	2.38	**2.72**	1.34	3.10	**3.36**

models on RULER-8K with 4K activated tokens. As shown in Table 6, DashAttention outperforms InfLLMv2 on average and matches full attention, indicating that our method is effective beyond the MiniCPM-4 family.

5.2 Efficiency Benchmark

We benchmark DashAttention's two-stage kernels in two regimes: *prefill* and *decoding*. We compare against **FlashAttention3** [36], **NSA** [5],[3] and **InfLLMv2** [6]. All methods use a chunk size of 64. We sweep chunk sparsity $s \in \{75\%, 87.5\%, 93.75\%\}$, with corresponding routing top-k equal to $(1-s)\cdot N/64$; the DashAttention active-block bitmask is randomized to the target sparsity to isolate the last stage's cost. Prefill is run at batch size 1 and decoding at batch size 24. See App. B.2 for the full benchmark setup, including hardware and motivation for these batch sizes.

Prefill. DashAttention is the fastest method at every operating point (Table 7, left), with speedups over dense FlashAttention ranging from $1.34\times$ to $3.09\times$. The gap over InfLLMv2 and NSA is widest at the densest setting, where these methods' top-k overhead is least amortized, and narrows toward extreme sparsity, but DashAttention stays ahead in every cell.

Decoding. In the memory-bound decoding regime (Table 7, right), DashAttention is again the fastest at every operating point, reaching $3.36\times$ over FlashAttention at n_{kv}=96K, s=93.75%, against $3.10\times$ for InfLLMv2. Compared to FA, the advantage grows monotonically with both context length and sparsity as the cost of dense attention eclipses Stage 1's routing overhead. Compared to InfLLMv2, this result follows from DashAttention's Stage 2 (Sec. 3.5): we walk the bit-packed active-block mask in a single fused pass, avoiding the explicit top-k and per-query index materialization InfLLMv2 performs between its scoring and attention stages.

5.3 Cost-Effectiveness Analysis

We combine performance and efficiency to examine each method's accuracy–sparsity trade-off.

For the 8B model, we sweep the sparsity (γ for DashAttention[4], k for NSA and InfLLMv2) of each sparse method to obtain points at increasing sparsity levels and report the resulting HELMET overall accuracy. DashAttention dominates NSA and InfLLMv2 across the sweep (Fig. 2) and slightly exceeds full attention at low to moderate sparsity. As sparsity increases, the gap to the baselines widens sharply: at ∼90% sparsity, DashAttention retains 39.4% overall accuracy, exceeding InfLLMv2 by 9% and NSA by 19%. This highlights DashAttention's adaptiveness: fixed top-k over-allocate easy queries or under-allocate hard ones, whereas α-entmax reshapes the support adaptively.

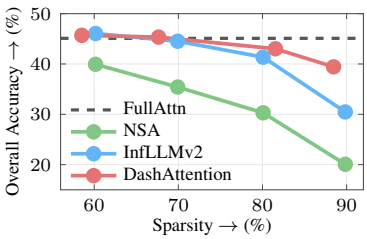


Figure 2: Accuracy–Sparsity Pareto frontiers on HELMET.

[3]No official NSA implementation has been released; we thus use a open-source Triton implementation at https://github.com/XunhaoLai/native-sparse-attention-triton.

[4]While α in α-entmax also controls sparsity, we choose to adjust a temperature γ instead as it allows us to re-use kernels optimized for specific values of α, such as $\alpha = 1.5$ in our case, which enjoys fast convergence and reduces all expressions to linear or quadratic operations; thus avoiding exponentials and logarithms and making the overall method faster.

5.4 Dynamic Sparsity Analysis

Since DashAttention uses α-entmax in Stage 1 to induce sparsity, it can dynamically allocate sparsity across different layers according to the geometry of the Stage 1 scores. To show this property, we measure the sparsity of each layer using a 16K-length input from RULER-SG1. The analysis is shown in Fig. 3. Notably, the early layers tend to remain denser, while the middle layers become sparser, automatically producing an effect similar to budget allocation strategies proposed in prior works [37, 38, 9].

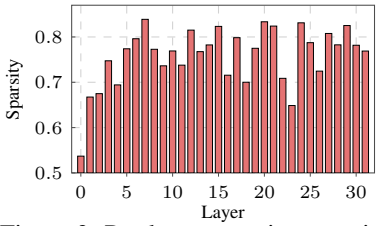


Figure 3: Per-layer attention sparsity.

6 Related Work

Attention and KV Cache Optimizations. Attention sparsification methods are developed to mitigate the computational and memory bottlenecks of long-context attention. Early approaches use static patterns, such as attention sinks [39] and sliding windows [40], to approximate full attention, while later works introduce random [41] and dynamic patterns [42–45], e.g., block sparsity [43, 46], to better exploit semantic structure. Prior methods further exploit head heterogeneity by assigning different sparse patterns to different attention heads [43, 47–49], improving long-context modeling. Although these optimizations significantly improve efficiency, they introduce a training-inference mismatch. In parallel, KV cache optimization methods have been proposed to reduce memory access cost and GPU memory usage. Eviction methods [42, 50–52] permanently discard redundant KV pairs, while KV offloading methods [53–55] store most KV cache on CPU and fetch only query-relevant parts during decoding. KV cache compression can also be learned end-to-end through continued training [55–57]. KV quantization methods [58–60] further reduce memory footprint by storing KV cache in low-bit formats. These approaches are orthogonal to our method, as they primarily compress KV cache rather than optimize attention sparsity.

Trainable Sparse Attentions. Recent approaches [61, 62] integrate sparse attention into LLM training to reduce training–inference mismatch and improve task performance. SeerAttn [63], NSA [5], and MoBA [9] introduce trainable block-sparse attention into LLMs, using top-k selection over compressed attention scores to identify important blocks. InfLLMv2 [6] improves the short-context inference efficiency of NSA by introducing a unified kernel in the final stage, while FSA [64] further extends this design to smaller GQA group sizes. HSA [65] introduces a local encoder to summarize each chunk, but adds substantial parameters and is less compatible with pretrained softmax models during long-context continual training. Other token-wise sparse attention methods, such as DSA [66], CSA1/2 [67, 68] and HISA [69], are highly effective only at large model scales and under the MLA [70] and CED [68] architecture. The performance of these proposed methods are bounded by a fixed top-k function, which does not introduce any dynamic property [71, 72]. Another line of work seeks to accelerate α-entmax for modern hardware [12, 13] . Entmax-based models [7, 15, 73, 74] achieve inherent sparsity through a mapping from $\mathbb{R}^n$ to $\triangle_n$ that produces exact zeros, and have shown advantages in both task performance and long-context extrapolation [8]. However, they still require computing the full $\boldsymbol{QK}^\top$ tensor, drastically limiting inference acceleration. Our method DashAttention bridges these two directions by introducing α-entmax into hierarchical sparse attention while remaining easy to adapt from pretrained softmax models.

7 Conclusion

We propose DashAttention, an end-to-end differentiable and adaptively sparse hierarchical attention mechanism. When integrated into long-context continual pretraining, DashAttention outperforms existing sparse attention methods (NSA and InfLLMv2) in terms of both performance and speed. DashAttention demonstrates a favorable cost-effectiveness trade-off in high-sparsity regimes and achieves speedups of 3.36× and 1.35× over FlashAttention-3 and InfLLMv2, respectively. One limitation of this work is that the DashAttention kernels have not yet been integrated into modern LLM serving frameworks, such as vLLM and SGLang, which is left for future work.

Acknowledgments

We would like to thank the SARDINE lab team for the helpful discussions. This work is supported by the National Natural Science Foundation of China (No. 62576186), and Tsinghua University Initiative Scientific Research Program. This work was supported by the project DECOLLAGE (ERC-2022-CoG 101088763), by the Portuguese Recovery and Resilience Plan through project C645008882-00000055 (Center for Responsible AI), and by FCT/MECI through national funds and, when applicable, co-funded EU funds under DOI:10.54499/UID/50008/2025: Instituto de Telecomunicações. Edoardo M. Ponti is supported by the ERC Starting Grant AToM-FM (ERC-2025-StG 101222956).

References

[1] Omer Goldman, Alon Jacovi, Aviv Slobodkin, Aviya Maimon, Ido Dagan, and Reut Tsarfaty. Is it really long context if all you need is retrieval? towards genuinely difficult long context NLP. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16576–16586, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[2] Piotr Nawrot, Robert Li, Renjie Huang, Sebastian Ruder, Kelly Marchisio, and Edoardo M. Ponti. The sparse frontier: Sparse attention trade-offs in transformer llms. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*, 2026.

[3] 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.

[4] Petar Veličković, Christos Perivolaropoulos, Federico Barbero, and Razvan Pascanu. Softmax is not enough (for sharp size generalisation). In *Forty-second International Conference on Machine Learning*, 2025.

[5] 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 Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 23078–23097, Vienna, Austria, July 2025. Association for Computational Linguistics.

[6] Weilin Zhao, Zihan Zhou, Zhou su, Chaojun Xiao, Yuxuan Li, Yanghao Li, Yudi Zhang, Weilun Zhao, Zhen Li, Yuxiang Huang, Ao Sun, Xu Han, and Zhiyuan Liu. InfLLM-v2: Dense-sparse switchable attention for seamless short-to-long adaptation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[7] Ben Peters, Vlad Niculae, and André F. T. Martins. Sparse sequence-to-sequence models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1504–1519, Florence, Italy, July 2019. Association for Computational Linguistics.

[8] Pavlo Vasylenko, Hugo Pitorro, Andre Martins, and Marcos Vinicius Treviso. Long-context generalization with sparse attention. In *The Fourteenth International Conference on Learning Representations*, 2026.

[9] 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, Yutao Zhang, Zhilin Yang, Xinyu Zhou, Mingxing Zhang, and Jiezhong Qiu. MoBA: Mixture of block attention for long-context LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[10] Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Re. Flashattention: Fast and memory-efficient exact attention with IO-awareness. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[11] Andre Martins and Ramon Astudillo. From softmax to sparsemax: A sparse model of attention and multi-label classification. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *International Conference on Machine Learning (ICML)*, volume 48 of *Proceedings of Machine Learning Research*, pages 1614–1623, New York, New York, USA, 20–22 Jun 2016. PMLR.

[12] Nuno Gonçalves, Marcos V Treviso, and Andre Martins. Adasplash: Adaptive sparse flash attention. In *Forty-second International Conference on Machine Learning*, 2025.

[13] Nuno Gonçalves, Hugo Pitorro, Vlad Niculae, Edoardo Ponti, Lei Li, Andre Martins, and Marcos Treviso. Adasplash-2: Faster differentiable sparse attention. In *Forty-third International Conference on Machine Learning*, 2026.

[14] 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*, pages 4895–4901, 2023.

[15] Mathieu Blondel, Andre Martins, and Vlad Niculae. Learning classifiers with fenchel-young losses: Generalized entropies, margins, and algorithms. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pages 606–615. PMLR, 16–18 Apr 2019.

[16] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024.

[17] Philippe Tillet, H. T. Kung, and David Cox. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*, MAPL 2019, page 10–19, New York, NY, USA, 2019. Association for Computing Machinery.

[18] OpenBMB. Infllm-v2-data-5b dataset. https://huggingface.co/datasets/openbmb/InfLLM-V2-data-5B, 2025.

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

[20] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, 2024.

[21] Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context models effectively and thoroughly. In *The Thirteenth International Conference on Learning Representations*, 2025.

[22] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37:95266–95290, 2024.

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

[24] Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, 2019.

[25] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

[26] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pages 4791–4800, 2019.

[27] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[28] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[29] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, 2019.

[30] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[31] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[32] Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*, 2024.

[33] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[34] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

[35] Alejandro R Salamanca, Diana Abagyan, Daniel D'souza, Ammar Khairi, David Mora, Saurabh Dash, Viraat Aryabumi, Sara Rajaee, Mehrnaz Mofakhami, Ananya Sahu, et al. Tiny aya: Bridging scale and multilingual depth. *arXiv preprint arXiv:2603.11510*, 2026.

[36] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[37] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

[38] Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3258–3270, 2024.

[39] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, 2024.

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

[41] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc., 2020.

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

[43] 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. *arXiv preprint arXiv:2407.02490*, 2024.

[44] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020.

[45] 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.

[46] 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.

[47] Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.

[48] Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*, 2024.

[49] Gang Lin, Dongfang Li, Zhuoen Chen, Yukun Shi, Xuhui Chen, Baotian Hu, and Min Zhang. Lycheedecode: Accelerating long-context llm inference via hybrid-head sparse decoding. *arXiv preprint arXiv:2602.04541*, 2026.

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

[51] Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, et al. R-kv: Redundancy-aware kv cache compression for reasoning models. *arXiv preprint arXiv:2505.24133*, 2025.

[52] Yushi Bai, Qian Dong, Ting Jiang, Xin Lv, Zhengxiao Du, Aohan Zeng, Jie Tang, and Juanzi Li. Indexcache: Accelerating sparse attention via cross-layer index reuse. *arXiv preprint arXiv:2603.12201*, 2026.

[53] Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. Infllm: Training-free long-context extrapolation for llms with an efficient context memory. *Advances in neural information processing systems*, 37:119638–119661, 2024.

[54] Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

[55] Yuxiang Huang, Pengjie Wang, Jicheng Han, Weilin Zhao, Zhou Su, Ao Sun, Hongya Lyu, Hengyu Zhao, Yudong Wang, Chaojun Xiao, et al. Nosa: Native and offloadable sparse attention. *arXiv preprint arXiv:2510.13602*, 2025.

[56] Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M. Ponti. Dynamic memory compression: retrofitting llms for accelerated inference. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[57] Adrian Łańcucki, Konrad Staniszewski, Piotr Nawrot, and Edoardo Ponti. Inference-time hyper-scaling with KV cache compression. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[58] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

[59] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[60] Hailin Zhang, Xiaodong Ji, Yilin Chen, Fangcheng Fu, Xupeng Miao, Xiaonan Nie, Weipeng Chen, and Bin Cui. Pqcache: Product quantization-based kvcache for long context llm inference. *Proceedings of the ACM on Management of Data*, 3(3):1–30, 2025.

[61] Wenxuan Li, Chengruidong Zhang, Huiqiang Jiang, et al. Mtraining: Distributed dynamic sparse attention for efficient ultra-long context training. *arXiv preprint arXiv:2510.18830*, 2025.

[62] 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.

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

[64] Ran Yan, Youhe Jiang, and Binhang Yuan. Flash sparse attention: An alternative efficient implementation of native sparse attention kernel. *arXiv e-prints*, pages arXiv–2508, 2025.

[65] Jing Liu, Jianqiao Lu, Yao Luo, Yuan Yang, Chen Zheng, Deyi Liu, Mengzhao Chen, Chaoyi Zhang, Yunshui Li, Jin Ma, et al. Hsa: Head-wise sparse attention for efficient and accurate long-context inference.

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

[67] DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro/blob/main/DeepSeek_V4.pdf, 2026. Technical report, Hugging Face repository, accessed 2026-04-28.

[68] Anyi Xu, B Li, Bangcai Lin, Bing Xue, BingCheng Xian, Bingzheng Xu, Bochao Wu, Bowei Zhang, Boyi Deng, CC Yu, et al. Deepseek-v4. 1-flash: Pushing the limits of kv cache compression. *arXiv preprint arXiv:2609.19969*, 2026.

[69] Yufei Xu, Fanxu Meng, Fan Jiang, Yuxuan Wang, Ruijie Zhou, Zhaohui Wang, Jiexi Wu, Zhixin Pan, Xiaojuan Tang, Wenjie Pei, et al. Hisa: Efficient hierarchical indexing for fine-grained sparse attention. *arXiv preprint arXiv:2603.28458*, 2026.

[70] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[71] Jintao Zhang, Kai Jiang, Chendong Xiang, Weiqi Feng, Yuezhou Hu, Haocheng Xi, Jianfei Chen, and Jun Zhu. Spargeattention2: Trainable sparse attention via hybrid top-k+ top-p masking and distillation fine-tuning. *arXiv preprint arXiv:2602.13515*, 2026.

[72] Wentao Ni, Kangqi Zhang, Zhongming Yu, Oren Nelson, Mingu Lee, Hong Cai, Fatih Porikli, Jongryool Kim, Zhijian Liu, and Jishen Zhao. Double-p: Hierarchical top-p sparse attention for long-context llms. *arXiv preprint arXiv:2602.05191*, 2026.

[73] Xingyue Huang, Xueying Ding, Mingxuan Ju, Yozen Liu, Neil Shah, and Tong Zhao. Threshold differential attention for sink-free, ultra-sparse, and non-dispersive language modeling. *arXiv preprint arXiv:2601.12145*, 2026.

[74] Gonçalo M Correia, Vlad Niculae, and André FT Martins. Adaptively sparse transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2174–2184, 2019.

[75] Constantino Tsallis. Possible generalization of boltzmann-gibbs statistics. *Journal of statistical physics*, 52(1):479–487, 1988.

[76] Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, dahai li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024.

[77] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

[78] Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. Olmes: A standard for language model evaluations. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 5005–5033, 2025.

[79] 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 2024.

[80] Xiang Hu, Jiaqi Leng, Jun Zhao, Kewei Tu, and Wei Wu. Hardware-aligned hierarchical sparse attention for efficient long-term memory access. *arXiv preprint arXiv:2504.16795*, 2025.

[81] Xiang Hu, Zhanchao Zhou, Ruiqi Liang, Zehuan Li, Wei Wu, and Jianguo Li. Every token counts: Generalizing 16m ultra-long context in large language models. *arXiv preprint arXiv:2511.23319*, 2025.

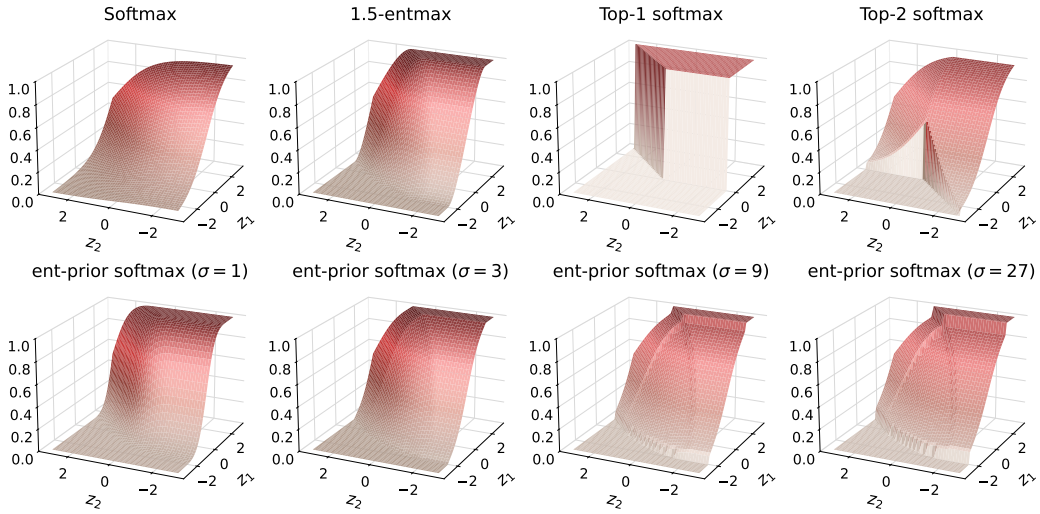


Figure 4: Visualization of mappings for different values of α, and top-k softmax with $k = 1, 2$. Each panel shows how p_0 varies for the input $\boldsymbol{z} = [0, z_1, z_2]$. Ent-prior softmax denotes a variant of softmax in which the softmax scores are scaled by an entmax prior. The prior is also generated from $\boldsymbol{z}$ and normalized according to Proposition 1.

A The α-entmax Mapping

This appendix briefly reviews the α-entmax transformation used in our coarse routing stage (Stage 1), and includes basic properties and calculation of α-entmax.

Variational View of Mappings to Probability Simplex. Given a vector of scores $\boldsymbol{z} \in \mathbb{R}^n$, α-entmax maps $\boldsymbol{z}$ to a probability vector by solving a regularized prediction problem over the simplex [7]:

$$\alpha\text{-entmax}(\boldsymbol{z}) = \underset{\boldsymbol{p} \in \triangle_n}{\arg\max}\, \boldsymbol{p}^\top \boldsymbol{z} + H_\alpha(\boldsymbol{p}), \qquad \triangle_n = \left\{\boldsymbol{p} \in \mathbb{R}^n : \boldsymbol{p} \geq \boldsymbol{0},\ \boldsymbol{1}^\top \boldsymbol{p} = 1\right\}, \tag{11}$$

where H_α is the Tsallis α-entropy [75]:

$$H_\alpha(\boldsymbol{p}) = \begin{cases} \dfrac{1}{\alpha(\alpha-1)} \displaystyle\sum_{j=1}^{n} \left(p_j - p_j^\alpha\right), & \alpha \neq 1, \\ -\displaystyle\sum_{j=1}^{n} p_j \log p_j, & \alpha = 1. \end{cases} \tag{12}$$

The parameter α controls the shape of the regularizer and, consequently, the sparsity of the resulting distribution. The optimal solution has a thresholding form. There exists a scalar threshold τ such that

$$\alpha\text{-entmax}(\boldsymbol{z})_i = \left[(\alpha-1)z_i - \tau\right]_+^{\frac{1}{\alpha-1}}, \tag{13}$$

where τ is chosen so that the entries sum to one. Therefore, any coordinate satisfying $(\alpha-1)z_i \leq \tau$ is assigned exactly zero mass. This property is central to our use of α-entmax: the router does not merely downweight irrelevant chunks, but can remove them from the routed set altogether.

Special Cases. The α-entmax family contains softmax and sparsemax as limiting or special cases. As $\alpha \to 1^+$, the Tsallis entropy converges to Shannon entropy, and Eq. (11) recovers the standard softmax transformation. At $\alpha = 2$, the mapping becomes sparsemax [11], which is the Euclidean projection of the score vector onto the simplex, i.e., $\text{sparsemax}(\boldsymbol{z}) = \arg\min_{\boldsymbol{p} \in \triangle_n} \|\boldsymbol{z} - \boldsymbol{p}\|_2$. Intermediate values, such as $\alpha = 1.5$, provide a useful compromise: they preserve differentiability while allowing exact zeros in the output distribution [7].

Computing the threshold. Evaluating α-entmax reduces to finding the scalar threshold τ in Eq. (13). For $\alpha = 2$ and $\alpha = 1.5$, specialized algorithms are available [11, 7]. For general α, τ can be obtained by root finding, such as bisection over a bounded interval [15]. Recent GPU implementations further accelerate this step with safeguarded higher-order updates [12, 13].

Why it is useful for routing. In our setting, α-entmax is applied at the chunk-routing level, where exact zeros have a direct computational interpretation: chunks assigned zero probability are not passed to the expensive second-stage attention computation. This differs from top-k softmax in two important ways. First, top-k fixes the number of selected chunks in advance, so the routing budget is an external hyperparameter rather than a consequence of the score distribution. By contrast, α-entmax determines the active support from the scores themselves, allowing different queries and sequences to route to different numbers of chunks. Second, top-k changes support through a discrete selection operator; within a region where the selected set is fixed, gradients do not describe how an unselected chunk should change its score to enter the routed set. In α-entmax, the routing probabilities arise from a continuous sparse transformation, so score changes near the boundary are reflected in the probability map rather than handled by a separate hard truncation step. Figure 4 illustrates that α-entmax functions are smooth but cannot accelerate inference, since the full $\boldsymbol{QK}^\top$ matrix must still be computed. In contrast, top-k softmax can accelerate inference but is non-smooth and stiff. With a suitable choice of σ, ent-prior softmax, similar to DashAttention, can be smoother and more dynamic than top-k softmax while still enabling inference acceleration through hierarchical attention.

Differentiability of DashAttention. Entmax priors provide a native mechanism for enabling differentiability in hierarchical sparse attention. Unlike NSA, which relies on compressed attention to pass gradients to the first stage, or InfLLMv2, where the first stage is not trainable, our approach allows gradients to flow back through the routed chunks via 1.5-entmax. While 1.5-entmax can produce exact zeros, it admits a closed-form Jacobian within regions where the support remains fixed. At support-change boundaries, it is piecewise differentiable, analogous to sparsemax or ReLU. In practice, automatic differentiation applies the corresponding subgradient or Jacobian, enabling end-to-end gradient-based training. Therefore, from a computational perspective, DashAttention inherits the differentiability of α-entmax and thus can be viewed as a fully end-to-end differentiable mechanism.

B Reproducibility Settings

B.1 Performance Evaluation

B.1.1 Training

We choose the 1B, 3B, and 8B base versions of MiniCPM-4 [19] as our backbone models, motivated by their relatively large GQA group sizes. Specifically, the 1B model uses a group size of 8, while the 3B and 8B models adopt a group size of 16. We train all three models on InfLLM-5B [18] with a context length of 16K, followed by a short supervised fine-tuning (SFT) stage using the original MiniCPM-4 [19] training data. We employ the WSD learning rate scheduler [76], and conduct training using the Megatron framework [77]. All experiments are performed on 32 NVIDIA A800 GPUs with Intel® Xeon® Platinum 8470 CPUs running CentOS Linux 7 (Core), using `bfloat16` precision. For long-context continual pretraining, we train on 2.0B, 3.4B, and 5.0B tokens for the 1B, 3B, and 8B models, respectively. The subsequent SFT stage uses 0.6B, 1.3B, and 1.9B tokens correspondingly. Notably, long-context continual pretraining is a standard practice for modern LLM development; therefore, our method introduces neither additional training stages nor extra training overhead compared with full attention and other baselines.

The DashAttention and baseline models are trained under the following settings. For NSA and InfLLMv2, we set the attention sink size to 64, the sliding window size to 1024, the block size to 64, and the total number of selected tokens to 4096. For DashAttention, the block size is also set to 64. We do not enforce attention to the sink tokens; instead, the sliding window consists of the diagonal block and the last block, resulting in a total size of 64–127 tokens. During training, we set $t_c = 1$ for the 1B and 3B models, and $t_c = 0.5$ for the 8B model. The parameter α is gradually increased from 1.25 to 1.5 throughout training. We set $\sigma = 10^6$ for both 1B and 3B model, and $\sigma = 10^8$ for the 8B model in both training and inference.

B.1.2 Benchmarks

We evaluate DashAttention against all baselines on RULER [20] and HELMET [21]. We use the official RULER benchmark suite to assess long-context performance, and Olmes [78] for HELMET evaluation. For each RULER task, we evaluate 50 samples, and for each HELMET task, 100 samples, using greedy generation. All reported metrics and averaging methods follow the official settings. The reported sparsity is computed by excluding the first 4096 tokens, as these tokens use full attention in NSA and InfLLMv2. For general tasks, we evaluate the trained models on MMLU [22], MMLU-Pro [23], CSQA [24], IFEval [25], HellaSwag [26], GSM8K [27], MATH [28], DROP [29], MBPP [30], and HumanEval [31], by using the LM-Evaluation-Harness [79] framework. For full attention, InfLLMv2, and DashAttention, we use softmax full attention during inference. For NSA, since removing the compressed attention during inference would cause training-inference mismatch, we inference the model with compressed attention scores but without sparsity in Stage 2.

B.2 Efficiency Evaluation

Setup. All measurements were taken on a single NVIDIA GH200 GPU with CUDA 13.2, using PyTorch 2.10.0 and Triton 3.6.0 at `bf16` precision. Each kernel is wrapped with `torch.compile` and timed with Triton's `do_bench`; we measure the attention kernels themselves, with query/key/value projections and token-level KV-cache management out of scope. The GQA configuration is fixed across methods: H_q=32 query heads, H_{kv}=2 key/value heads, head dimension D=128, and a chunk size of 64 for all three sparse methods. We sweep $N \in \{16, 32, 64, 96\}$K and chunk sparsity $s \in \{75, 87.5, 93.75\}\%$, with routing top-$k = (1 - s) \cdot N/64$ for the top-k baselines.

Batch sizes. We run prefill at batch size 1 and decoding at batch size 24. Prefill is set to batch size 1 to keep the Stage 1 chunk-score tensor within memory at N=96K. For decoding, batch size 1 latency is dominated by kernel-launch overhead, which CUDA graphs can mitigate, but we instead use 24 to operate outside that regime.

Why we randomize the active-block bitmask. Stages 0 and 1 do not depend on the sparsity level s, so to sweep s we inject at Stage 2 a random bit-packed mask at the target sparsity rather than a router-produced one. This isolates the cost of Stage 2 at a controlled sparsity and matches the effective sparsity used by the top-k baselines.

Scope. These are isolated attention-kernel benchmarks (i.e., no request scheduler, no paged KV cache, etc.) so they do not translate one-for-one to end-to-end TTFT or token throughput in a production serving stack. Integration into a serving framework such as vLLM or SGLang is the natural next step and we leave it to future work.

B.3 Performance and Efficiency Trade-Offs

We report the Pareto frontier at the in-distribution context length of 16K in the main text. DashAttention's effective sparsity is input-dependent and controlled at inference by the scaling factor t_c; we sweep $t_c \in \{0.50, 0.30, 0.15, 0.10\}$, with $t_c = 0.5$ recovering the training operating point of the 8B checkpoint and the smaller values probing denser regimes. NSA and InfLLMv2 instead use a deterministic per-query top-k budget, with average density top-$k \times B/L$ ($B = 64$, $L = 16384$). Targeting densities of 10%, 20%, 30%, and 40% of the prefix to span roughly the same range as DashAttention, we pick top-$k \in \{26, 51, 77, 102\}$. Each point in the figure is plotted at its empirically measured density.

C Extended Related Works

In this section, we introduce the major related hierarchical trainable sparse attention and show the difference in the design between our method and existing methods. Since we position our work as a hierarchical, block-wise trainable sparse attention method, we empirically compare it with the state-of-the-art approaches, NSA and InfLLMv2. For other lines of work, such as element-wise sparse attention, we discuss their design principles to highlight their orthogonality to key-novelties of our proposed method.

SeerAttention and MoBA. SeerAttention [63] and MoBA [9] are representative early works on trainable sparse attention. By compressing both Q and K vectors, these methods introduce dynamic block-sparse patterns through a top-k selection function. Both methods are integrated into model training, allowing the model to adapt to sparse attention patterns and mitigating the train–inference mismatch that arises when attention sparsity is applied. However, because Q is also compressed within each chunk along the sequence dimension, these methods are difficult to apply during decoding. In contrast, DashAttention selects KV chunks independently for each query, enabling exact alignment between training and inference as well as between prefill and decoding. This design improves long-context performance while accelerating both prefill and decoding.

NSA and InfLLMv2. NSA [5] first introduces a separate KV selection mechanism by leveraging the group size in GQA to satisfy the Tensor Core GEMM shape constraints. It adopts three complementary attention patterns: sliding-window attention, block-sparse attention, and compressed attention. The compressed KV representations are produced by an MLP with additional parameters introduced during long-context continual pretraining. However, since compressed attention is integrated into the model architecture, it is difficult to remove even when performing inference on short-context inputs, which can slow down short-context inference. To address this issue, InfLLMv2 [6] removes the compressed attention output from the final attention computation and replaces the MLP with mean pooling, thereby avoiding additional parameters. InfLLMv2 achieves both better performance and higher efficiency, and further provides an official efficient CUDA implementation. Nevertheless, both NSA and InfLLMv2 rely on top-k selection, which lacks dynamic adaptability when selecting the essential KV cache entries to attend to. DashAttention alleviates this limitation by introducing an entmax-based Stage 1 mechanism. Moreover, since DashAttention weakens the influence of prior scores through a large hyperparameter σ, it can also perform inference without Stage 1 scores and sparsity, aligning with the inference behavior of InfLLMv2. DashAttention also removes the overlapped block compression strategy introduced in NSA and InfLLMv2, resulting in fewer chunk summarizations and accelerating Stage 1 without causing substantial performance degradation. In this paper, we choose these methods as baselines to demonstrate the effectiveness of the dynamicity introduced by entmax-based Stage 1.

Element-wise Sparse Attention. Adopted in DeepSeek-V3.2 [66] and DeepSeek-V4 [67], element-wise hierarchical sparse attention methods, such as DSA [66], CSA, and HCA [67], have emerged as alternatives to block-selection methods by enabling finer-grained token selection. These methods typically train a token indexer to predict the importance of each KV position with respect to a given query input, supervised by an additional loss. Although they can achieve better performance than block-level sparse attention, their efficiency relies on large model scales and the use of the MLA architecture, which reduces the relative overhead introduced by the indexer. As a result, they are not directly compatible with the commonly adopted GQA architecture. In addition, these methods adopt a top-k selection function. Notably, DashAttention is not fundamentally incompatible with element-wise sparse attention: a coarse-level token indexer could also be combined with entmax-based selection techniques and our prior-based training method. However, because training extremely large LLMs with element-wise sparse attention is currently infeasible for us, we leave the integration of DashAttention with these methods to future work.

HSA. HSA [80, 81] has demonstrated the long-context performance and extrapolation capabilities. By using a local encoder and incorporating a `[CLS]` landmark token into the input sequence, HSA achieves strong long-context performance. However, the local encoder relies on a large number of newly introduced parameters and requires substantial training. Moreover, HSA cannot be directly adapted from existing pretrained softmax full-attention models, which limits its generalizability. In contrast, DashAttention's local attention mechanism degenerates to mean pooling at the beginning of training, providing a smooth transition from an existing full-attention model to a DashAttention sparse-attention model. Since our comparison is conducted under a long-context continual pretraining setup, we exclude HSA as a baseline because it cannot be adapted from pretrained models.

D Equivalent Attention Bias Form of Diagonal Estimation

We start by re-stating the original proposition.

Proposition 1. *Calculating the attention output $\boldsymbol{o}_i$ with Eq. (7) and Eq. (9) is equivalent to first computing $\mu_i = \mathrm{mean}_{j\in\mathcal{R}_i}\{\log w_{i,c(j)}\}$, followed by*

$$d_{i,j} = \begin{cases} \dfrac{\log w_{i,c(j)} - \mu_i}{\sigma}, & j \in \mathcal{R}_i, \\ 0, & j \in \mathcal{D}_i, \end{cases} \qquad \boldsymbol{o}_i = \sum_{j\in\mathcal{R}_i\cup\mathcal{D}_i} \frac{\exp(z_{i,j} + d_{i,j})\,\boldsymbol{v}_j}{\sum_{t\in\mathcal{R}_i\cup\mathcal{D}_i}\exp(z_{i,t} + d_{i,t})}.$$

Proof. We start the proof by expanding λ_i:

$$\lambda_i = \mathrm{sigmoid}\left(\mathrm{KL}(\boldsymbol{u}_{\mathcal{R}_i}\|\boldsymbol{w}'_{\mathcal{R}_i}) + \log\frac{|\mathcal{R}_i|}{|\mathcal{D}_i|}\right) = \frac{1}{1+\exp\left(-\mathrm{KL}(\boldsymbol{u}_{\mathcal{R}_i}\|\boldsymbol{w}'_{\mathcal{R}_i}) - \log\frac{|\mathcal{R}_i|}{|\mathcal{D}_i|}\right)}.$$

Since $\boldsymbol{u}_{\mathcal{R}_i}$ is uniform on $\mathcal{R}_i$, the KL term simplifies as

$$\mathrm{KL}(\boldsymbol{u}_{\mathcal{R}_i}\|\boldsymbol{w}'_{\mathcal{R}_i}) = \sum_{j\in\mathcal{R}_i}\frac{1}{|\mathcal{R}_i|}\log\frac{1/|\mathcal{R}_i|}{w'_{i,j}} = -\log|\mathcal{R}_i| - \frac{1}{|\mathcal{R}_i|}\sum_{j\in\mathcal{R}_i}\log w'_{i,j}.$$

Denote the geometric mean of $\boldsymbol{w}'_i$ on the routed support by

$$\eta_i = \exp\Big(\tfrac{1}{|\mathcal{R}_i|}\textstyle\sum_{j\in\mathcal{R}_i}\log w'_{i,j}\Big) = \Big(\prod_{j\in\mathcal{R}_i} w'_{i,j}\Big)^{1/|\mathcal{R}_i|}.$$

Plugging back, we have

$$\lambda_i = \frac{1}{1+|\mathcal{D}_i|\,\eta_i}, \qquad 1-\lambda_i = \frac{|\mathcal{D}_i|\,\eta_i}{1+|\mathcal{D}_i|\,\eta_i}.$$

Therefore, for all $j \in \mathcal{R}_i$, we have

$$\log g_{i,j} = \log\lambda_i + \log w'_{i,j} = \log w'_{i,j} - \log(1+|\mathcal{D}_i|\,\eta_i),$$

and for all $j \in \mathcal{D}_i$, we have

$$\log g_{i,j} = \log\Big(\tfrac{1-\lambda_i}{|\mathcal{D}_i|}\Big) = \log\eta_i - \log(1+|\mathcal{D}_i|\,\eta_i).$$

Denote

$$A = \log\eta_i - \log(1+|\mathcal{D}_i|\,\eta_i),$$

then $\forall j\in\mathcal{D}_i,\ \log g_{i,j} - A = 0 = d_{i,j}$, and

$$\forall j\in\mathcal{R}_i,\ \log g_{i,j} - A = \log w'_{i,j} - \log\eta_i.$$

Recalling $w'_{i,j} = w_{i,c(j)}^{1/\sigma}/(B\,\mathbf{1}^\top\boldsymbol{w}_i^{1/\sigma})$, the normalizer $B\,\mathbf{1}^\top\boldsymbol{w}_i^{1/\sigma}$ does not depend on j and so cancels between $\log w'_{i,j}$ and $\log\eta_i$, giving

$$\log g_{i,j} - A = \log w_{i,c(j)}^{1/\sigma} - \mathrm{mean}\{\log\boldsymbol{w}_i^{1/\sigma}\} = \frac{\log w_{i,c(j)} - \mu_i}{\sigma} = d_{i,j}.$$

Therefore, by plugging the results above into Eq. (7), we can obtain that

$$\begin{aligned}\boldsymbol{o}_i &= \sum_{j\in\mathcal{R}_i\cup\mathcal{D}_i}\frac{g_{i,j}\exp(z_{i,j})\cdot\boldsymbol{v}_j}{\sum_{t\in\mathcal{R}_i\cup\mathcal{D}_i} g_{i,t}\exp(z_{i,t})} = \sum_{j\in\mathcal{R}_i\cup\mathcal{D}_i}\frac{\exp(z_{i,j}+\log g_{i,j} - A)\cdot\boldsymbol{v}_j}{\sum_{t\in\mathcal{R}_i\cup\mathcal{D}_i}\exp(z_{i,t}+\log g_{i,t} - A)} \\ &= \sum_{j\in\mathcal{R}_i\cup\mathcal{D}_i}\frac{\exp(z_{i,j}+d_{i,j})\cdot\boldsymbol{v}_j}{\sum_{t\in\mathcal{R}_i\cup\mathcal{D}_i}\exp(z_{i,t}+d_{i,t})}.\end{aligned}$$

□

E Non-Dispersion Property

In this section, we focus on the dispersion property with respect to the Stage 1 design and the head aggregation strategy in hierarchical sparse attention. We begin by restating the definition of dispersion following [8]. Dispersion describes the phenomenon in which the entropy of attention probabilities grows at $\sim \log n$, introducing greater uncertainty as the sequence length increases and thereby making long-context modeling more challenging.

Definition 2 (Dispersion). *Given a bounded sequence $\boldsymbol{z}_n \in \mathbb{R}^n$ and a mapping $f : \mathbb{R}^n \to \triangle_n$, we call the mapping f a dispersive function if*

$$\lim_{n\to\infty} \frac{\mathcal{H}(f(\boldsymbol{z}_n))}{\log n} = 1,$$

where $\mathcal{H}$ is the Shannon's entropy. If there exists a constant $0 \le c < 1$,

$$\limsup_{n\to\infty} \frac{\mathcal{H}(f(\boldsymbol{z}_n))}{\log n} \le c < 1,$$

then f is a non-dispersive function.

Existing top-k based hierarchical sparse attention methods, such as NSA [5] and InfLLMv2 [6], are non-dispersive in Stage 2. In particular, applying softmax only over the top-k selected attention logits constitutes a non-dispersive operation.

Theorem 2 (Top-k-Softmax is non-dispersive). $\mathrm{softmax}(\text{top-}k(\cdot))$ *is a non-dispersive function, i.e., given a bounded sequence $\boldsymbol{z} \in \mathbb{R}^n$, there is always*

$$\lim_{n\to\infty} \frac{\mathcal{H}(\mathrm{softmax}(\text{top-}k(\boldsymbol{z})))}{\log n} = 0.$$

Proof. $\mathcal{H}(\mathrm{softmax}(\text{top-}k(\boldsymbol{z}))) \le \log k$, thus

$$\lim_{n\to\infty} \frac{\mathcal{H}(\mathrm{softmax}(\text{top-}k(\boldsymbol{z})))}{\log n} \le \lim_{n\to\infty} \frac{\log k}{\log n} = 0.$$

□

This shows that top-k sparse attention mechanisms are non-dispersive and therefore can potentially perform well in long-context modeling.

However, when applying sparse attention to GQA models, existing methods commonly require all query heads within the same group to share an identical sparse pattern. To achieve this, a softmax-based head aggregation is typically applied before top-k sparse attention, where the Stage 1 scores are merged on the probability simplex to prevent the aggregated scores from being overly dominated by a few strong heads. For DashAttention, entmax head-aggregation is applied to achieve similar effect.

Next, we first formally define the head aggregation mechanism, followed by proving the dispersion property of softmax head aggregagation and non-dispersion property of entmax head disaggregation.

Definition 1 (Head aggregation). *Given a mapping $f : \mathbb{R}^n \to \triangle_n$, H bounded sequence $\{\boldsymbol{z}^{(h)}\}_{h=1}^H \subset \mathbb{R}^n$, and the aggregation weights $\theta = (\theta_1, \cdots, \theta_H)^\top \in \triangle_H$, the aggregated probability is*

$$\mathrm{aggr}_f\left(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(H)}; \boldsymbol{\theta}\right) = \sum_{h=1}^{H} \theta_h \cdot f\left(\boldsymbol{z}^{(h)}\right) \in \triangle_n.$$

Theorem 1 (Dispersion in head aggregation)**.** *Under any finite H and $\theta \in \triangle_H$, the following always hold:*

1. Softmax head aggregation is dispersive, i.e.

$$\lim_{n\to\infty} \frac{\mathcal{H}\left(\text{aggr}_{\text{softmax}}\left(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(H)}; \boldsymbol{\theta}\right)\right)}{\log n} = 1,$$

2. Denote $\boldsymbol{p}^{(h)} = \alpha\text{-entmax}(\boldsymbol{z}^{(h)})$, if there are $\|\boldsymbol{p}^{(h)}\|_0 = \mathcal{O}\left(n^{\beta_h}\right), \beta_h \in (0,1)$, then entmax head aggregation is not dispersive, and

$$\limsup_{n\to\infty} \frac{\mathcal{H}\left(\text{aggr}_{\alpha\text{-entmax}}\left(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(H)}; \boldsymbol{\theta}\right)\right)}{\log n} \leq \max_{h\in[H]} \beta_h < 1.$$

Proof. We first prove that softmax head aggregation is dispersive.

Denote $\boldsymbol{p}^{(h)} = \text{softmax}\left(\boldsymbol{z}^{(h)}\right)$.

First, we find the lower bound of the numerator by using $\mathcal{H}(\cdot)$'s concavity.

$$\begin{aligned}\mathcal{H}\left(\text{aggr}_{\text{softmax}}\left(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(H)}; \boldsymbol{\theta}\right)\right) &= \mathcal{H}(\theta_1 \boldsymbol{p}^{(1)} + \cdots + \theta_H \boldsymbol{p}^{(H)}) \\ &\geq \theta_1 \mathcal{H}\left(\boldsymbol{p}^{(1)}\right) + \cdots + \theta_H \mathcal{H}\left(\boldsymbol{p}^{(H)}\right).\end{aligned}$$

Then, since softmax is a dispersive function, we have

$$\begin{aligned}\lim_{n\to\infty} \frac{\mathcal{H}\left(\text{aggr}_{\text{softmax}}\left(\boldsymbol{z}^{(1)}, \cdots, \boldsymbol{z}^{(H)}; \boldsymbol{\theta}\right)\right)}{\log n} &\geq \theta_1 \lim_{n\to\infty} \frac{\mathcal{H}\left(\boldsymbol{p}^{(1)}\right)}{\log_n} + \cdots + \theta_H \lim_{n\to\infty} \frac{\mathcal{H}\left(\boldsymbol{p}^{(H)}\right)}{\log_n} \\ &= \theta_1 + \cdots + \theta_H = 1.\end{aligned}$$

Then, since $\forall \boldsymbol{p} \in \triangle_n$, $\mathcal{H}(\boldsymbol{p}) \leq \log n$, we also have $\lim_{n\to\infty} \frac{\mathcal{H}(\boldsymbol{p})}{\log n} \leq 1$. Therefore, we can obtain the following limitation:

$$\lim_{n\to\infty} \frac{\mathcal{H}\left(\text{softaggr}\left(\boldsymbol{z}^{(1)}, \boldsymbol{z}^{(2)}, \cdots, \boldsymbol{z}^{(H)}; \boldsymbol{\theta}\right)\right)}{\log n} = 1.$$

Then, we prove that the entmax head aggregation under given assumption is not dispersive.

Since $\forall h \in \{1, 2, \cdots, H\}$, $\|\boldsymbol{p}^{(h)}\|_0 = \mathcal{O}(n^{\beta_h})$, then

$$\|\boldsymbol{p}\|_0 = \left\|\frac{1}{H}\sum_{h=1}^{H} \boldsymbol{p}^{(h)}\right\|_0 = \left\|\sum_{h=1}^{H} \boldsymbol{p}^{(h)}\right\|_0 \leq \sum_{h=1}^{H} \left\|\boldsymbol{p}^{(h)}\right\|_0 = \sum_{h=1}^{H} \mathcal{O}\left(n^{\beta_h}\right) = \mathcal{O}\left(n^{\max_{h\in[H]} \beta_h}\right),$$

therefore we can obtain that $\|\boldsymbol{p}\|_0 = \mathcal{O}\left(n^{\max_{h\in[H]} \beta_h}\right)$.

By using [8]'s Proposition 1, the theorem can be simply proved.

□

F Additional Experimental Results

F.1 Full Benchmark Results of HELMET

Here, we list the detailed results of DashAttention compared with baselines on HELMET in Table 8.

F.2 Ablations on Chunk Sizes

DashAttention trained with one chunk size can be used for inference with different chunk sizes. A smaller chunk size enables finer-grained Stage 1 selection, leading to better performance under the

Table 8: ***HELMET*** scores (%) of DashAttention compared with all baselines for sparse adaptation (detailed).

Task		MiniCPM-4-1B				MiniCPM-4-3B				MiniCPM-4-8B			
		FullAttn	NSA	InfLLMv2	DashAttention	FullAttn	NSA	InfLLMv2	DashAttention	FullAttn	NSA	InfLLMv2	DashAttention
Recall	*JKV*	48.0	20.0	21.0	**23.0**	55.0	24.0	45.0	**53.0**	96.0	25.0	**91.0**	73.0
	MK2	82.0	16.0	48.0	**61.0**	93.0	20.0	51.0	**77.0**	99.0	28.0	73.0	**98.0**
	MK3	43.0	5.0	19.0	**33.0**	70.0	9.0	38.0	**45.0**	99.0	14.0	62.0	**82.0**
	MV	93.5	52.5	86.8	**90.3**	96.3	66.5	**98.0**	92.5	99.0	73.0	99.5	**100.0**
	Avg.	66.6	23.4	43.7	**51.8**	78.6	29.9	58.0	**66.9**	98.3	35.0	81.4	**88.3**
ICL	*Ban*	47.0	36.0	**54.0**	51.0	41.0	**57.0**	51.0	50.0	72.0	73.0	**75.0**	71.0
	Cli	64.0	36.0	60.0	**64.0**	45.0	46.0	46.0	**63.0**	64.0	79.0	**83.0**	73.0
	NLU	63.0	46.0	**62.0**	60.0	61.0	55.0	48.0	**64.0**	77.0	75.0	76.0	**78.0**
	TrC	80.0	78.0	84.0	**88.0**	54.0	44.0	**54.0**	53.0	53.0	52.0	60.0	**77.0**
	TrF	41.0	34.0	36.0	**42.0**	23.0	24.0	22.0	**26.0**	14.0	20.0	**24.0**	15.0
	Avg.	59.0	46.0	59.2	**61.0**	44.8	45.2	44.2	**51.2**	56.0	59.8	**63.6**	62.8
Rerank	Avg.	13.9	15.0	15.5	**16.5**	25.3	20.5	**24.8**	20.0	37.8	27.9	36.8	**39.7**
RAG	*Hot*	20.0	**19.0**	17.0	14.0	28.0	30.0	**35.0**	**35.0**	33.0	28.0	**36.0**	33.0
	NQ	25.0	32.0	31.0	**35.0**	40.0	**42.0**	41.0	41.0	45.0	43.0	44.0	**46.0**
	Pop	32.0	32.0	**38.0**	35.0	46.0	37.0	43.0	**46.0**	45.0	38.0	**48.0**	**48.0**
	Tri	61.0	57.0	62.0	**70.0**	80.0	73.0	**78.0**	69.0	90.0	84.0	88.0	**89.0**
	Avg.	34.5	35.0	37.0	**38.5**	48.5	45.5	**49.3**	47.8	53.3	48.3	**54.0**	**54.0**
LongQA	*InC*	45.0	38.0	**40.0**	34.0	39.0	42.0	**46.0**	38.0	49.0	46.0	**50.0**	**50.0**
	InQ	14.3	13.9	**15.2**	13.3	20.5	18.2	**22.1**	14.6	27.2	25.1	**26.5**	26.0
	Nar	61.0	52.0	61.0	**62.0**	68.0	59.0	**75.0**	55.0	81.0	82.0	**83.0**	81.0
	Avg.	40.1	34.6	**38.7**	36.4	42.5	39.7	**47.7**	35.9	52.4	51.0	**53.2**	52.3
Cite	*ASQ*	17.4	12.8	**14.7**	14.7	23.2	14.1	10.8	**19.6**	36.0	24.0	**32.3**	32.1
	Qam	0.4	0.1	**0.4**	0.1	1.4	0.5	**1.1**	1.1	4.0	3.3	**4.3**	2.8
	Avg.	8.9	6.4	**7.6**	7.4	12.3	7.3	6.0	**10.3**	20.0	13.7	**18.3**	17.4
Summ.	*InS*	1.0	0.8	**0.9**	0.3	2.4	1.0	**1.1**	1.0	5.3	3.7	**4.1**	3.8
	Mul	7.4	8.1	8.9	**13.2**	16.9	13.3	**18.5**	15.6	27.6	**25.6**	23.3	23.5
	Avg.	4.2	4.4	4.9	**6.7**	9.7	7.1	**9.8**	8.3	16.5	**14.7**	13.7	13.7
Overall ↑ (%)		32.5	23.6	29.5	**31.2**	37.4	27.9	34.2	**34.3**	47.7	35.8	45.9	**46.9**
Sparsity ↑ (%)		0.0	75.0	75.0	**75.4**	0.0	75.0	75.0	**75.4**	0.0	75.0	75.0	**75.4**

Table 9: DashAttention's performance under different chunk sizes (trained with 64).

Chunk Size	RULER-16K													Avg. ↑ (%)	Sparsity. ↑ (%)
	SG1	SG2	SG3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2		
16	100.0	100.0	100.0	100.0	98.0	100.0	100.0	99.0	57.2	50.6	76.0	70.0	50.0	84.7	77.0
32	100.0	100.0	100.0	98.0	96.0	86.0	100.0	99.0	60.0	49.6	72.0	72.0	54.0	83.6	75.7
64	100.0	100.0	96.0	96.0	92.0	68.0	96.0	95.0	54.0	45.0	66.7	70.0	54.0	79.4	75.1
128	100.0	92.0	98.0	90.0	80.0	58.0	83.5	88.5	50.0	48.8	59.3	74.0	46.0	74.5	75.0

same sparsity constraint. To verify this, we evaluate our 8B model, trained with a chunk size of 64, on RULER using test-time chunk sizes of $\{16, 32, 64, 128\}$. We adjust the entmax temperature to achieve $\sim 75\%$ sparsity. As shown in Table 9, smaller inference chunk sizes yield better task performance, demonstrating that the inference chunk size is not constrained by the chunk size used during training.

F.3 Full Results of DashAttention Inference in Full Attention

Table 10: RULER performance of DashAttention models inference with softmax full attention (detailed).

	Method	RULER													
		SG1	SG2	SG3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	Avg.
1B	FullAttn	100.0	100.0	100.0	**98.0**	78.0	48.0	87.0	**89.0**	17.6	**1.8**	66.7	44.0	36.0	66.6
	DashAttention-full	**100.0**	**100.0**	**100.0**	92.0	**92.0**	**46.0**	**88.0**	88.5	**62.0**	0.8	**71.3**	38.0	**36.0**	**70.4**
3B	FullAttn	**100.0**	**100.0**	98.0	**100.0**	94.0	**60.0**	**96.0**	**96.5**	20.4	**5.8**	19.3	**60.0**	**50.0**	69.2
	DashAttention-full	**100.0**	**100.0**	**100.0**	**100.0**	**96.0**	56.0	89.0	94.5	**21.6**	2.8	**82.7**	50.0	42.0	**71.9**
8B	FullAttn	**100.0**	**100.0**	**100.0**	98.0	**100.0**	96.0	**100.0**	**99.5**	58.4	49.2	**78.0**	**78.0**	**52.0**	85.3
	DashAttention-full	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**	99.5	99.0	**77.2**	**56.2**	74.7	70.0	50.0	**86.7**

Table 11: HELMET performance of DashAttention models inference with softmax full attention (detailed).

	Method	HELMET							
		Recall	ICL	Rerank	RAG	LongQA	Cite	Summ.	Overall
1B	FullAttn	**66.6**	59.0	13.9	34.5	**40.1**	**8.9**	4.2	32.5
	DashAttention-full	66.2	**61.2**	**17.5**	**39.8**	37.5	8.6	**5.7**	**33.8**
3B	FullAttn	78.6	44.8	**25.3**	**48.5**	**42.5**	**12.3**	9.7	**37.4**
	DashAttention-full	**80.3**	**48.2**	23.1	46.8	40.2	10.5	**9.7**	37.0
8B	FullAttn	98.3	56.0	37.8	**53.3**	52.4	20.0	**16.5**	47.7
	DashAttention-full	**99.8**	**59.0**	**40.5**	**53.3**	**53.7**	**20.9**	13.6	**48.7**

Similar to InfLLMv2 [6], models trained with DashAttention can perform inference with vanilla softmax full attention without notable performance degradation. As shown in Table 10 and Table 11, DashAttention-full outperforms FullAttn in most settings, demonstrating that models trained with sparse attention can even achieve better long-context performance than those trained with full attention when evaluated using full-attention inference.

F.4 Cost-Effectiveness Analysis

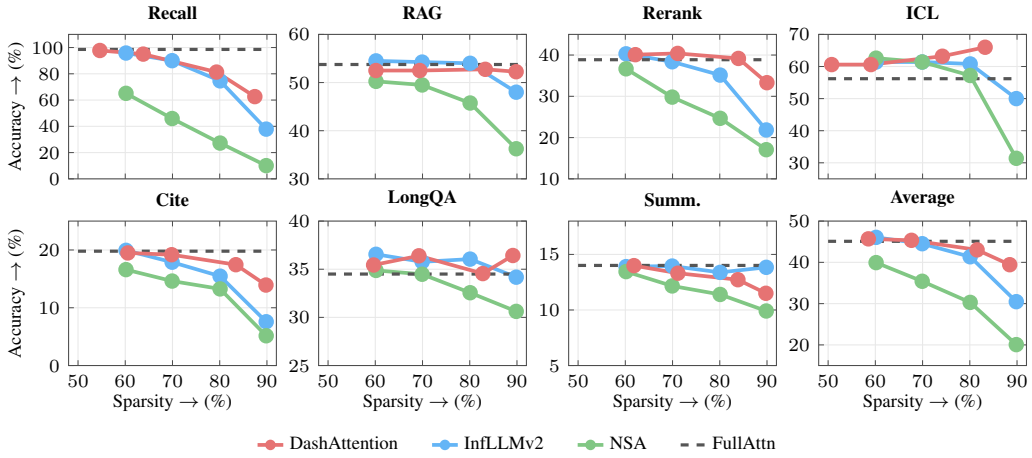


Figure 5: Accuracy–Sparsity Pareto frontiers on HELMET at 16K context length.

We present detailed results for the cost-effectiveness analysis across the seven HELMET tasks in Fig. 5. The results show that DashAttention is generally more robust under sparse settings than NSA and InfLLMv2, especially on tasks such as Recall, RAG, Rerank, and Cite. By using an α-entmax-guided chunk routing strategy, DashAttention allocates the sparsity budget more effectively across layers and heads under high sparsity, thereby minimizing performance degradation even in extreme configurations.

F.5 Ablations on Local Attention and σ

Table 12: Performance under different local attention and σ settings.

Local Attn	σ	RULER-16K													Avg. ↑ (%)
		SG1	SG2	SG3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	
✗	10^8	**100.0**	**100.0**	**100.0**	**100.0**	**96.0**	84.0	**100.0**	98.0	59.6	47.4	71.3	70.0	**56.0**	83.3
✓	10^1	**100.0**	**100.0**	**100.0**	98.0	94.0	82.0	99.5	90.0	**60.4**	46.4	69.3	**72.0**	54.0	82.0
✓	10^4	**100.0**	**100.0**	**100.0**	96.0	**96.0**	82.0	99.5	98.5	60.0	48.4	71.3	**72.0**	**56.0**	83.1
✓	10^{12}	**100.0**	**100.0**	**100.0**	**100.0**	**96.0**	84.0	99.5	**99.0**	60.0	45.6	**72.0**	70.0	54.0	83.1
✓	10^8	**100.0**	**100.0**	**100.0**	98.0	**96.0**	**86.0**	**100.0**	**99.0**	60.0	**49.6**	**72.0**	**72.0**	54.0	**83.6**

Here, we study the effectiveness of local attention and the role of σ in controlling prior strength through ablations. To assess local attention, we replace it with mean pooling, following prior work [6, 9]. We also evaluate different prior scaling factors, $\sigma \in \{10^1, 10^4, 10^8, 10^{12}\}$, where larger σ weakens the entmax priors. As σ increases, the priors gradually diminish; in the limit $\sigma \to +\infty$, they degenerate into non-differentiable binary masks, making the summarization query head $\bar{\boldsymbol{q}}$ untrainable and reducing local attention to mean pooling. We benchmark all settings on our 8B model using RULER at a 16K context length.

Table 12 shows that removing local attention degrades performance. A small σ makes the prior overly strong and leads to suboptimal results, while σ becomes relatively stable once sufficiently large. We therefore set $\sigma = 10^8$, which weakens the priors while preserving the differentiability.

G Full Algorithms

Algorithms 1–3 give the overall view for the three stages of DashAttention. We write $\boldsymbol{X}_i^{\mathcal{G}_r,j}$ for the slab of a tensor $\boldsymbol{X}$ obtained by fixing token i, restricting the head axis to $\mathcal{G}_r$, and taking the j-th block along the chunk axis; e.g. $\boldsymbol{Z}_i^{\mathcal{G}_r,j} \in \mathbb{R}^{g_q \times 32}$ is the Stage 1 SRAM tile. The chunk-axis block width is fixed at 32 because the mask is bit-packed into $\mathrm{int}32$ words.

Algorithm 1 Stage 0: Local Chunk Summarization

Require: Per-head summary queries $\bar{\boldsymbol{q}} \in \mathbb{R}^{T_c \times h_{kv} \times d_h}$ (RoPE pre-applied), keys $\boldsymbol{K} \in \mathbb{R}^{n \times h_{kv} \times d_h}$, chunk size B, sequence length n.
Ensure: Chunk summaries $\bar{\boldsymbol{K}} \in \mathbb{R}^{T_c \times h_{kv} \times d_h}$.

1: Divide $\boldsymbol{K}$ along the sequence axis into $T_c = \lfloor n/B \rfloor$ blocks $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_{T_c}$, each of size $B \times h_{kv} \times d_h$.
2: **for** $c = 1, \ldots, T_c$ and $r = 1, \ldots, h_{kv}$ in parallel **do**
3: Load $\bar{\boldsymbol{q}}_c^{(r)} \in \mathbb{R}^{d_h}$ and $\boldsymbol{K}_c^{(r)} \in \mathbb{R}^{B \times d_h}$ from HBM to SRAM.
4: On chip, compute $\boldsymbol{s}_c^{(r)} = \boldsymbol{K}_c^{(r)} \bar{\boldsymbol{q}}_c^{(r)} / \sqrt{d_h} \in \mathbb{R}^B$.
5: On chip, compute $\bar{\boldsymbol{k}}_c^{(r)} = \mathrm{softmax}(\boldsymbol{s}_c^{(r)})^\top \boldsymbol{K}_c^{(r)} \in \mathbb{R}^{d_h}$.
6: Write $\bar{\boldsymbol{k}}_c^{(r)}$ to HBM.
7: **end for**

Algorithm 2 Stage 1: α-entmax Block Routing

Require: $\boldsymbol{Q} \in \mathbb{R}^{n \times h_q \times d_h}$, $\bar{\boldsymbol{K}} \in \mathbb{R}^{T_c \times h_{kv} \times d_h}$; $\alpha > 1$, scaling γ, prior strength σ.
Ensure: Bit-packed mask $\boldsymbol{\mathcal{M}} \in \{0,1\}^{n \times h_{kv} \times T_c}$ (stored as $\mathrm{int}32$ words, one bit per chunk), routing bias $\boldsymbol{d} \in \mathbb{R}^{n \times h_{kv} \times T_c}$.

Materialize $\bar{\boldsymbol{Z}} \in \mathbb{R}^{n \times h_q \times T_c}$ in HBM with $\bar{z}_{i,c}^{(h)} = \gamma(\alpha - 1)\,\langle \boldsymbol{q}_i^{(h)}, \bar{\boldsymbol{k}}_c^{(r(h))} \rangle / \sqrt{d_h}$.
Divide $\bar{\boldsymbol{Z}}$ along the chunk axis into $T_{tc} = \lceil T_c/32 \rceil$ blocks of width 32 (one $\mathrm{int}32$ mask word each), so that $\bar{\boldsymbol{Z}}_i^{\mathcal{G}_r,j} \in \mathbb{R}^{g_q \times 32}$ denotes the j-th chunk-block at row i, restricted to query-head group $\mathcal{G}_r$. Likewise define block slices $\boldsymbol{\mathcal{M}}_i^{(r),j} \in \{0,1\}^{32}$ and $\boldsymbol{d}_i^{(r),j} \in \mathbb{R}^{32}$.
for $1 \le i \le n$ and $1 \le r \le h_{kv}$ in parallel **do**
  Let $J_i = \lceil c_i/32 \rceil$ (causal blocks only).
  For each $h \in \mathcal{G}_r$, solve $f(\tau) = \sum_{c<c_i} \max(0, \bar{z}_{i,c}^{(h)} - \tau)^{1/\alpha - 1} - 1 = 0$ for $\tau_i^{(h)}$ via AdaSplash-2's algorithm [12].
  Initialize $S_i^{(r)} \leftarrow 0,\ N_i^{(r)} \leftarrow 0$ on SRAM.
  for $j = 1$ to J_i **do**
    Load $\bar{\boldsymbol{Z}}_i^{\mathcal{G}_r,j} \in \mathbb{R}^{g_q \times 32}$ from HBM to SRAM.
    On chip, compute $\boldsymbol{w}_i^{(r),j} = \frac{1}{g_q} \sum_{h \in \mathcal{G}_r} \max(0, \bar{\boldsymbol{z}}_i^{(h),j} - \tau_i^{(h)})^{1/\alpha - 1} \in \mathbb{R}^{32}$.
    On chip, $\boldsymbol{\mathcal{M}}_i^{(r),j} \leftarrow \mathbb{1}[\boldsymbol{w}_i^{(r),j} > 0]$.
    $S_i^{(r)} \mathrel{+}= \sum_{c : w_{i,c}^{(r),j} > 0} \log w_{i,c}^{(r),j}, \quad N_i^{(r)} \mathrel{+}= \|\boldsymbol{\mathcal{M}}_i^{(r),j}\|_1$.
    Bit-pack $\boldsymbol{\mathcal{M}}_i^{(r),j}$ into one $\mathrm{int}32$ word and write to HBM.
  end for
  $\mu_i^{(r)} \leftarrow S_i^{(r)} / N_i^{(r)}$.
  for $j = 1$ to J_i **do**
    Reload $\bar{\boldsymbol{Z}}_i^{\mathcal{G}_r,j}$ and recompute $\boldsymbol{w}_i^{(r),j}$ on chip.
    On chip, $\boldsymbol{d}_i^{(r),j} \leftarrow (\log \boldsymbol{w}_i^{(r),j} - \mu_i^{(r)})/\sigma$ on active entries, else 0.
    Write $\boldsymbol{d}_i^{(r),j}$ to HBM.
  end for
  Set $\mathcal{M}_{i,c_i}^{(r)} \leftarrow 1,\ d_{i,c_i}^{(r)} \leftarrow 0$ (diagonal chunk).
end for

Algorithm 3 Stage 2: Prior-Induced Sparse Softmax Attention

Require: $\boldsymbol{Q} \in \mathbb{R}^{n \times h_q \times d_h}$, $\boldsymbol{K}, \boldsymbol{V} \in \mathbb{R}^{n \times h_{kv} \times d_h}$, per-token mask $\boldsymbol{\mathcal{M}} \in \{0,1\}^{n \times h_{kv} \times T_c}$ and routing bias $\boldsymbol{d} \in \mathbb{R}^{n \times h_{kv} \times T_c}$ from Stage 1, chunk size B, GQA group size g_q.
Ensure: $\boldsymbol{O} \in \mathbb{R}^{n \times h_q \times d_h}$.

Divide $\boldsymbol{K}$ and $\boldsymbol{V}$ along the sequence axis into $T_c = \lceil n/B \rceil$ blocks $\boldsymbol{K}_1, \ldots, \boldsymbol{K}_{T_c}$, $\boldsymbol{V}_1, \ldots, \boldsymbol{V}_{T_c}$, each of size $B \times h_{kv} \times d_h$.
for $i = 1, \ldots, n$ and $r = 1, \ldots, h_{kv}$ in parallel **do**
  Load $\boldsymbol{Q}_i^{\mathcal{G}_r} \in \mathbb{R}^{g_q \times d_h}$ from HBM to SRAM.
  Initialize $\boldsymbol{O}_i^{\mathcal{G}_r} \leftarrow \boldsymbol{0}_{g_q \times d_h}$, $\boldsymbol{\ell}_i \leftarrow \boldsymbol{0}_{g_q}$, $\boldsymbol{m}_i \leftarrow -\infty\, \boldsymbol{1}_{g_q}$ on SRAM.
  for j such that $\mathcal{M}_{i,j}^{(r)} = 1$ **do**
    Load $\boldsymbol{K}_j^{(r)}, \boldsymbol{V}_j^{(r)} \in \mathbb{R}^{B \times d_h}$ and $d_{i,j}^{(r)} \in \mathbb{R}$ from HBM to SRAM.
    On chip, compute $\boldsymbol{S}_i^{\mathcal{G}_r,j} = \boldsymbol{Q}_i^{\mathcal{G}_r} \boldsymbol{K}_j^{(r)\top} / \sqrt{d_h} + d_{i,j}^{(r)}\, \boldsymbol{1}_{g_q \times B} \in \mathbb{R}^{g_q \times B}$.
    If $j = c_i$: apply causal mask to $\boldsymbol{S}_i^{\mathcal{G}_r,j}$.
    $\boldsymbol{m}_i' \leftarrow \max(\boldsymbol{m}_i, \mathrm{rowmax}(\boldsymbol{S}_i^{\mathcal{G}_r,j}))$, $\boldsymbol{P}_i^{\mathcal{G}_r,j} \leftarrow \exp(\boldsymbol{S}_i^{\mathcal{G}_r,j} - \boldsymbol{m}_i' \boldsymbol{1}^\top)$.
    $\boldsymbol{\ell}_i \leftarrow e^{\boldsymbol{m}_i - \boldsymbol{m}_i'} \odot \boldsymbol{\ell}_i + \mathrm{rowsum}(\boldsymbol{P}_i^{\mathcal{G}_r,j})$.
    $\boldsymbol{O}_i^{\mathcal{G}_r} \leftarrow \mathrm{diag}(e^{\boldsymbol{m}_i - \boldsymbol{m}_i'})\, \boldsymbol{O}_i^{\mathcal{G}_r} + \boldsymbol{P}_i^{\mathcal{G}_r,j} \boldsymbol{V}_j^{(r)}$.
    $\boldsymbol{m}_i \leftarrow \boldsymbol{m}_i'$.
  end for
  $\boldsymbol{O}_i^{\mathcal{G}_r} \leftarrow \mathrm{diag}(\boldsymbol{\ell}_i)^{-1} \boldsymbol{O}_i^{\mathcal{G}_r}$; write $\boldsymbol{O}_i^{\mathcal{G}_r}$ to HBM.
end for

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We claim that DashAttention perform better and is more efficient than existing hierarchical attention methods, e.g., NSA and InfLLMv2. This is proven in the experiment section Sec. 5. We also claim that existing methods have dispersion problem while DashAttention does not, which is proven in App. E.

   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: Limitations are included in Sec. 7.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

Answer: [Yes]

Justification: Assumptions and proofs are given in App. D and E.

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: Reproducibility settings are provided in App. B.

   Guidelines:

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

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

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

Answer: [No]

Justification: Code cannot be release at the time of submission. However, we will release the code later.

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: Experimental settings and details are provided in Sec.5 and App. B.

   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: [No]

   Justification: All benchmark results are based on greedy decoding. Therefore, there is no randomness. We are aligning the reported metrics with the related works in this area (efficient attentions) [6, 5, 12, 13].

   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: Compute resources are included in App. B.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: This research is conducted 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: [N/A]

    Justification: As far as we concern, this paper does not have negative societal impact. This paper introduces DashAttention, an efficient attention mechansism, that can only enhance the efficiency of long-context processing, and does not have negative impacts or ethic issues.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: We are only altering the attention mechanisms with DashAttention. No large datasets or powerful LLMs are released. There is no safety risks imposed by DashAttention.

    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 datasets, base models, and code bases included in this work are all referenced with official papers or websites.

    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: This 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: No crowdsourcing and research with human subjects are involved.

    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: No crowdsourcing and research with human subjects are involved.

    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: The core method of this paper does not involve LLMs as any important, original, or non-standard components.

    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.