HASTE: Hybrid Adaptive Sparse Attention with Dual Upper Bounds for Efficient LLM Inference

Anonymous ACL submission

Abstract

Decode-time attention in large language models is bottlenecked by memory bandwidth, since every generated token must attend over the entire KV cache. Existing sparse attention methods apply either a fixed global threshold (BLASST) or a fixed compression ratio (SPARQ), ignoring that attention entropy varies dramatically across layers and heads, from 6 nats in early layers to 1 nat in deep layers. We present HASTE (**H**ybrid **A**daptive **S**parse A**tt**ention T**e**chnique), which introduces: (1) *dual upper bounds* combining interval-arithmetic (QUEST) and SVD-based centroid (SACF) bounds to score KV blocks without loading them; (2) *entropy-guided adaptive sparsity budgets* (EGASB) that adjust per-head block retention based on online attention entropy; (3) a configurable *correction pass* for recomputing evicted blocks when estimated error exceeds a threshold; and (4) *pre-computed block metadata* that enables scoring without KV transfer.

1 Introduction

The rise of agentic AI workflows and self-hosted small language models (SLMs) has dramatically increased demand for efficient long-context inference. Organizations increasingly deploy compact models like TinyLlama-1.1B and Llama-3.2-1B on consumer hardware to serve multi-step reasoning, tool use, and retrieval-augmented generation, where context windows routinely span thousands of tokens. In these settings, autoregressive inference is fundamentally bottlenecked by memory bandwidth (Dao et al., 2022; Kwon et al., 2023): each newly generated token must attend over the full key-value (KV) cache, requiring $O(n \cdot d)$ memory transfers per attention head. As context lengths grow, this bandwidth cost dominates wall-clock latency, making efficient attention critical for practical deployment of SLMs on resource-constrained devices.

Sparse attention methods aim to reduce this cost by skipping KV blocks that contribute negligibly to the attention output. Recent approaches fall into two categories. *Fixed-threshold* methods like BLASST (Yuan et al., 2025) skip blocks whose softmax contribution falls below a calibrated threshold. *Fixed-ratio* methods like SPARQ (Ribar et al., 2024) select a predetermined fraction of tokens via approximate scoring. Both approaches apply a uniform sparsity strategy across all layers and heads.

However, attention patterns are far from uniform. Our analysis (Section 5.4) reveals that attention entropy (measured in nats, the natural unit of information) varies from 1 to 2 nats in deep layers (sharp, focused attention) to 5 to 6 nats in early layers (broad, distributed attention). A fixed threshold that is conservative enough for high-entropy heads wastes bandwidth on low-entropy heads, while an aggressive threshold that works for sharp heads destroys quality for broad ones.

We introduce HASTE, which addresses this mismatch through four contributions:

1. **Dual upper bounds.** We combine QUEST's interval-arithmetic bounds (Tang et al., 2024) with a novel SVD-based centroid bound (SACF). Taking the minimum of these two bounds yields tighter block importance scores than either bound alone (Lemma 1).

2. **Entropy-guided adaptive sparsity budget (EGASB).** After a brief warmup phase, we estimate per-head attention entropy from the online softmax state and dynamically adjust the number of blocks to retain. Low-entropy (sharp) heads keep fewer blocks, while high-entropy (broad) heads keep more.

3. **Correction pass (designed).** HASTE's architecture includes a configurable correction

mechanism that can recompute the worst-scoring evicted blocks when estimated error exceeds a threshold. While not activated in the current evaluation (the dual bounds proved sufficient), the mechanism provides an additional safety valve absent in BLASST and SPARQ.

4. **Precomputed block metadata.** Block importance is scored using precomputed per-block statistics (centroids, SVD bases, min/max vectors) without loading the actual KV tensors. This provides a fundamental bandwidth advantage over post-hoc methods that must first compute $QK^\top$.

We evaluate HASTE on TinyLlama-1.1B (Zhang et al., 2024) and Llama-3.2-1B across language modeling, needle-in-haystack retrieval, and reasoning benchmarks. HASTE achieves adaptive block eviction of 32 to 38% across domains while maintaining non-trivial retrieval accuracy on the RULER benchmark, where both BLASST and SPARQ drop to 0%.

2 Related Work

Efficient attention mechanisms. The standard multi-head attention (Vaswani et al., 2017) costs $O(n^2 d)$ for sequence length n. FlashAttention (Dao et al., 2022; Dao, 2023) reduces memory overhead through IO-aware tiling, while PagedAttention (Kwon et al., 2023) improves KV cache memory management via virtual-memory-inspired paging. At the architectural level, Multi-Query Attention (MQA) (Shazeer, 2019) and Grouped-Query Attention (GQA) (Ainslie et al., 2023) reduce KV cache size by sharing KV heads across query heads. These optimizations are orthogonal to sparsity, and HASTE operates post-training on any GQA or MHA model.

Subquadratic and fixed-pattern attention. Reformer (Kitaev et al., 2020) uses locality-sensitive hashing for $O(n \log n)$ attention, while Performer (Choromanski et al., 2021) linearizes softmax via random feature maps. These require retraining and may degrade on precise retrieval tasks. Fixed-pattern methods like Longformer (Beltagy et al., 2020), BigBird (Zaheer et al., 2020), and Sparse Transformers (Child et al., 2019) use predetermined patterns (local windows, global tokens, strided attention) decided at model design time, and cannot adapt to input-dependent distributions at inference.

KV cache compression. Quantization methods such as KIVI (Liu et al., 2024) reduce per-element bit-width to 2 to 4 bits, preserving all tokens at lower precision. Token eviction methods permanently discard KV entries: H2O (Zhang et al., 2023) retains heavy-hitter tokens, Scissorhands (Liu et al., 2023) exploits persistence of importance, and SnapKV (Li et al., 2024) clusters KV entries before generation. StreamingLLM (Xiao et al., 2024) retains attention sinks and a local window. Unlike permanent eviction, HASTE skips blocks per-query and retains the full cache for future queries, avoiding irreversible information loss.

Decode-time sparse attention. Closest to our work, QUEST (Tang et al., 2024) introduces interval-arithmetic upper bounds for top-K block selection without full KV transfer. BLASST (Yuan et al., 2025) uses running-max softmax thresholding to skip negligible blocks, reporting 1.48 to 1.62× speedup with CUDA kernels. SPARQ (Ribar et al., 2024) selects top-r query dimensions and top-k token positions for 8× compression. FastGen (Ge et al., 2024) learns per-head eviction policies, and MInference (Jiang et al., 2024) accelerates prefill via dynamic sparse patterns.

Positioning. HASTE differs from prior work in three ways: (1) it combines two complementary upper bounds rather than relying on a single scoring mechanism; (2) it adapts sparsity budgets per-head using online entropy estimation, unlike the fixed thresholds of BLASST or fixed ratios of SPARQ; and (3) it includes a configurable correction mechanism for error recovery that is absent in all prior decode-time methods.

3 Method

3.1 Preliminaries

Consider a single decode step where a query vector $\mathbf{q} \in \mathbb{R}^d$ must attend over a KV cache of n tokens organized into $B = \lceil n/b \rceil$ blocks of size b. Standard attention computes:

$$\mathrm{Attn}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{q}\mathbf{K}^\top}{\sqrt{d}}\right)\mathbf{V} \quad (1)$$

where $\mathbf{K} \in \mathbb{R}^{n \times d}$ and $\mathbf{V} \in \mathbb{R}^{n \times d}$. The memory bandwidth cost is $O(n \cdot d)$ for loading $\mathbf{K}$ and $\mathbf{V}$.

HASTE reduces this to $O(K \cdot b \cdot d)$ where $K \ll B$ is the total number of retained blocks.

3.2 Dual Upper Bounds

For each block j with keys $\mathbf{K}_j \in \mathbb{R}^{b \times d}$, we precompute metadata that enables scoring without loading $\mathbf{K}_j$.

QUEST bound (interval arithmetic). Following Tang et al. (2024), we store per-dimension min/max statistics:

$$k_j^{\min}[i] = \min_{t \in j} \mathbf{K}_j[t, i], \quad k_j^{\max}[i] = \max_{t \in j} \mathbf{K}_j[t, i] \tag{2}$$

The upper bound on the maximum attention score within block j is:

$$U_{\mathrm{Q}}(\mathbf{q}, j) = \frac{1}{\sqrt{d}} \sum_{i=1}^{d} \max\big(\mathbf{q}_i \cdot k_{j,i}^{\min},\ \mathbf{q}_i \cdot k_{j,i}^{\max}\big) \tag{3}$$

This bound is lossless: for any key $\mathbf{k} \in \mathbf{K}_j$, the dot product $\mathbf{q}^\top \mathbf{k}/\sqrt{d} \le U_{\mathrm{Q}}(\mathbf{q}, j)$.

SACF bound (SVD-based Approximate Centroid Filter). We compute the centroid and a rank-r approximation of each block's key matrix:

$$\mathbf{c}_j = \frac{1}{b} \sum_{t \in j} \mathbf{K}_j[t, :], \quad \mathbf{K}_j - \mathbf{1}\mathbf{c}_j^\top \approx \mathbf{U}_j \mathbf{\Sigma}_j \mathbf{V}_j^\top \tag{4}$$

where $\mathbf{U}_j \in \mathbb{R}^{b \times r}$ and $\mathbf{V}_j \in \mathbb{R}^{d \times r}$ contain the top-r left/right singular vectors, $\sigma_{\max,j} = \max_{t \in j} \|\mathbf{V}_j^\top (\mathbf{k}_t - \mathbf{c}_j)\|_2$ is the maximum projection norm, and $\delta_j = \max_t \|\mathbf{r}_t\|_2$ is the maximum ℓ_2 norm of the SVD residual vectors $\mathbf{r}_t = (\mathbf{k}_t - \mathbf{c}_j) - \mathbf{U}_j \mathbf{\Sigma}_j \mathbf{V}_j^\top[t, :]$. Decomposing $\mathbf{q}$ into components parallel and perpendicular to the right singular basis $\mathbf{V}_j$:

$$\mathbf{q}_\parallel = \mathbf{V}_j \mathbf{V}_j^\top \mathbf{q}, \qquad \mathbf{q}_\perp = \mathbf{q} - \mathbf{q}_\parallel \tag{5}$$

The SACF upper bound is:

$$U_{\mathrm{S}}(\mathbf{q}, j) = \frac{1}{\sqrt{d}} \big(\mathbf{q}^\top \mathbf{c}_j + \|\mathbf{q}_\parallel\| \cdot \sigma_{\max,j} + \|\mathbf{q}_\perp\| \cdot \delta_j\big) \tag{6}$$

Lemma 1 (SACF validity)**.** *For any key $\mathbf{k}_t \in \mathbf{K}_j$, the scaled dot product satisfies $\mathbf{q}^\top \mathbf{k}_t/\sqrt{d} \le U_{\mathrm{S}}(\mathbf{q}, j)$.*

Proof. Write $\mathbf{k}_t = \mathbf{c}_j + \mathbf{z}_t$ where $\mathbf{z}_t = \mathbf{k}_t - \mathbf{c}_j$ is the centered residual. Then $\mathbf{q}^\top \mathbf{k}_t = \mathbf{q}^\top \mathbf{c}_j + \mathbf{q}^\top \mathbf{z}_t$. Project $\mathbf{z}_t$ onto the SVD subspace: $\mathbf{z}_t = \mathbf{V}_j \boldsymbol{\alpha}_t + \mathbf{r}_t$ where $\boldsymbol{\alpha}_t = \mathbf{V}_j^\top \mathbf{z}_t$ and $\mathbf{r}_t$ is the residual with $\|\mathbf{r}_t\|_2 \le \delta_j$ by definition. Then:

$$\begin{aligned} \mathbf{q}^\top \mathbf{z}_t &= \mathbf{q}_\parallel^\top \mathbf{V}_j \boldsymbol{\alpha}_t + \mathbf{q}_\perp^\top \mathbf{r}_t \\ &\le \|\mathbf{q}_\parallel\| \cdot \|\boldsymbol{\alpha}_t\| + \|\mathbf{q}_\perp\| \cdot \|\mathbf{r}_t\|_2 \\ &\le \|\mathbf{q}_\parallel\| \cdot \sigma_{\max,j} + \|\mathbf{q}_\perp\| \cdot \delta_j \end{aligned}$$

where the first inequality uses Cauchy-Schwarz and the second uses $\|\boldsymbol{\alpha}_t\| \le \sigma_{\max,j}$ (since singular values bound coordinate magnitudes in the SVD basis) and $\|\mathbf{r}_t\|_2 \le \delta_j$ by definition. Dividing by $\sqrt{d}$ gives the result. □

Combined bound. The final score for block j is:

$$U(\mathbf{q}, j) = \min\big(U_{\mathrm{Q}}(\mathbf{q}, j),\ U_{\mathrm{S}}(\mathbf{q}, j)\big) \tag{7}$$

Proposition 1. *Since both U_{Q} and U_{S} are valid upper bounds on $\max_{t \in j} \mathbf{q}^\top \mathbf{k}_t/\sqrt{d}$, their minimum $U(\mathbf{q}, j)$ is also a valid upper bound and is at least as tight as either bound individually.*

3.3 Entropy-Guided Adaptive Sparsity Budget

Different attention heads exhibit vastly different entropy profiles (Section 5.4). EGASB adapts the block retention budget based on online entropy estimation.

Warmup phase. The first w blocks (default w=3) are always computed with full attention, building the online softmax state $(\hat{m}, \hat{s})$ where $\hat{m}$ is the running max logit and $\hat{s}$ the running sum of exponentials.

Entropy estimation. From the warmup state, we estimate attention entropy using the log-sum-exp identity:

$$\hat{H} = \log \hat{s} - \frac{1}{\hat{s}} \sum_{t \in \text{warmup}} e^{s_t - \hat{m}} (s_t - \hat{m}) \tag{8}$$

where s_t are the attention logits from warmup blocks. This computes $H = -\sum_i p_i \log p_i$ in the numerically stable online softmax form, avoiding explicit softmax materialization.

Adaptive budget. The total number of blocks to retain (including warmup) is:

$$K_{\text{total}} = K_{\min} + (K_{\max} - K_{\min}) \cdot \Big(1 - e^{-\hat{H}/\tau}\Big) \tag{9}$$

where $K_{\min} = \lfloor \rho_{\min} \cdot B \rfloor$, $K_{\max} = \lfloor \rho_{\max} \cdot B \rfloor$, and τ is the entropy temperature. After the warmup

phase, the number of additional blocks selected is $K_{\text{add}} = \max(0,\ K_{\text{total}} - w)$. Low $\hat{H}$ (sharp attention) yields $K_{\text{total}} \approx K_{\min}$, while high $\hat{H}$ (broad attention) yields $K_{\text{total}} \approx K_{\max}$. We set $\tau = 3.0$ based on TinyLlama's empirical entropy range of 0 to 7 nats; this places the transition midpoint near $\hat{H} \approx 2$ nats, separating the sharp deep layers from the broad early layers (Section 5.4).

3.4 DECO: GQA-Aware Block Eviction

In grouped-query attention (GQA) (Ainslie et al., 2023), G query heads share a single KV head. Evicting a KV block affects all G heads simultaneously, creating a tension: a block important to one head may be unimportant to another.

DECO (**D**ual-bound **E**ntropy-weighted **CO**nsensus) resolves this through an entropy-weighted majority vote. For each candidate block j, each query head g votes to evict if $U(\mathbf{q}_g, j)$ falls below its retention threshold. Votes are weighted by $\exp(-\hat{H}_g)$, giving sharper (low-entropy) heads more influence since they are more sensitive to block eviction:

$$\text{evict}(j) = \mathbf{1}\left[\frac{\sum_{g=1}^{G} w_g \cdot \mathbf{1}[U(\mathbf{q}_g, j) < \theta_g]}{\sum_{g=1}^{G} w_g} > \tau\right] \tag{10}$$

where $w_g = e^{-\hat{H}_g}$, θ_g is the per-head eviction cutoff, and $\tau = 0.6$ is the consensus threshold. This entropy-weighted scheme ensures that sharp heads (which are most sensitive to missing blocks) dominate the eviction decision, while broad heads contribute proportionally less. All results reported in this paper use this voting rule.

3.5 Correction Pass

HASTE's architecture includes a configurable correction pass that estimates the cumulative softmax mass of evicted blocks:

$$E_{\text{total}} = \sum_{j \notin \text{top-}K} \frac{e^{U(\mathbf{q},j) - \hat{m}}}{\hat{s}} \tag{11}$$

If E_{total} exceeds a correction threshold ϵ, the highest-scoring evicted blocks can be recomputed with full attention (budget: at most 20% of evicted blocks). In our current evaluation, the dual bounds and EGASB proved sufficient and the correction pass was not activated; we include the mechanism as a safety valve for future deployment at higher eviction rates.

Input: Query $\mathbf{q}$, KV cache $(\mathbf{K}, \mathbf{V})$, block metadata $\{M_j\}_{j=1}^{B}$
Config: $w, \rho_{\min}, \rho_{\max}, \tau, \epsilon$

Phase 0: Score all blocks
1: **for** $j = 1$ to B **do**
2: $\quad U_j \leftarrow \min(U_{\text{Q}}(\mathbf{q}, M_j),\ U_{\text{S}}(\mathbf{q}, M_j))$ ▷ Eq. 7

Phase 1: Warmup
3: Process top-w blocks by U_j; build online softmax state
4: Estimate entropy $\hat{H}$ from warmup state

Phase 2: Adaptive budget
5: $K_{\text{total}} \leftarrow \lfloor K_{\min} + (K_{\max} - K_{\min})(1 - e^{-\hat{H}/\tau}) \rfloor$ ▷ Eq. 9
6: $K_{\text{add}} \leftarrow \max(0, K_{\text{total}} - w)$
7: Select next K_{add} blocks by descending U_j
8: Gather selected $\mathbf{K}_{\text{sel}}, \mathbf{V}_{\text{sel}}$; compute attention

Phase 3: Correction (optional)
9: Estimate error E_{total} from evicted blocks
10: **if** $E_{\text{total}} > \epsilon$ **then** recompute top evicted blocks

return attention output

Algorithm 1: HASTE Decode Step (Single KV Head)

3.6 Full Algorithm

Algorithm 1 presents the complete HASTE decode step for a single KV head.

Figure 1 illustrates the full HASTE pipeline for a single decode step.

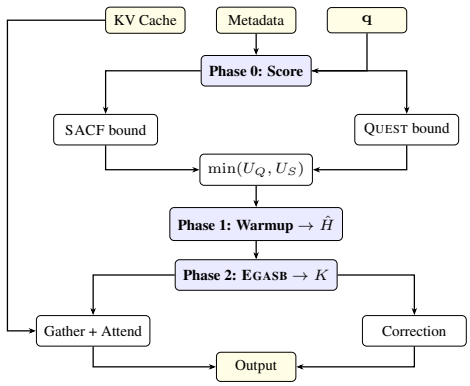


Figure 1: The HASTE decode pipeline. Phases 0–2 avoid KV loads; the optional correction pass (Phase 3) fires when estimated error exceeds ϵ.

3.7 Complexity Analysis

Bandwidth. Dense attention loads $2nd$ elements (K and V). HASTE loads $2Kbd$ elements plus $B(3d + rd + 3)$ elements for metadata: per-block min/max vectors ($2d$), centroid (d), right singular vectors (rd), and singular values plus residual norm (3 scalars). For TinyLlama with b=64, d=64, r=2, the metadata per block is $3 \times 64 + 2 \times 64 + 3 = 323$ elements, while K+V per block is $2 \times 64 \times 64 = 8{,}192$ elements. Thus metadata is approximately 4% of block data, and when

$K/B < 0.95$, HASTE achieves net bandwidth reduction.

FLOPs. Scoring all B blocks costs $O(Bd)$ for QUEST and $O(B(d + rd))$ for SACF. Attention over selected blocks costs $O(Kbd)$. Total: $O(Bd + Kbd)$ vs. $O(nd)$ for dense, with $Kb \leq n$.

4 Experimental Setup

Model. We use TinyLlama-1.1B-Chat (Zhang et al., 2024), a 1.1B-parameter model with the Llama architecture (Touvron et al., 2023): 22 layers, 32 query heads, 4 KV heads (GQA ratio 8), head dimension 64. Additionally, we evaluate on Llama-3.2-1B to demonstrate that our method generalizes across different Llama-family model variants and training regimes.

Hardware. All experiments run on a single NVIDIA A100 GPU (80GB VRAM) with CUDA 13.0, PyTorch 2.10, and FP16 precision.

Datasets. We evaluate on three domains:

- **Needle-in-haystack:** 160 RULER (Hsieh et al., 2024) synthetic retrieval items across 8 context-length buckets (512 to 131K tokens).
- **Language modeling:** 50 samples, with 25 drawn from Wikipedia and 25 from C4 (Raffel et al., 2020). Metric: perplexity over 256-token autoregressive continuations.
- **Reasoning:** 15 samples, with 5 from MATH500 (Hendrycks et al., 2021), 5 from AIME 2024 (Mathematical Association of America, 2024), and 5 from GPQA Diamond (Rein et al., 2023). Generation: 256 tokens.

Baselines. We compare against:

- **Dense:** Standard full attention (no sparsity).
- **BLASST** (Yuan et al., 2025): Running-max softmax thresholding with $\lambda = 1000/L$ where L is context length.
- **SPARQ** (Ribar et al., 2024): Dimension-then-token sparsity with compression ratio 8 ($r = 8$ dimensions, $k = n/16$ tokens) and mean reallocation.

HASTE configuration. Block size $b = 64$, SVD rank $r = 2$, warmup blocks $w = 3$, base keep ratio $\rho_{\min} = 0.5$, max keep ratio $\rho_{\max} = 0.9$, entropy temperature $\tau = 3.0$, correction threshold ϵ = auto. Full hyperparameters are listed in Table 1.

Parameter	Symbol	Value
Block size	b	64
SVD rank	r	2
Warmup blocks	w	3
Base keep ratio	$\rho_{\min}$	0.5
Max keep ratio	$\rho_{\max}$	0.9
Entropy temperature	τ	3.0
Correction budget	–	≤20% of evicted, capped at 50%
Metadata refresh	–	Every 16 tokens

Table 1: Full hyperparameter configuration for HASTE.

5 Results

5.1 Decode Throughput

Table 2 presents our primary result: decode throughput across three context-length buckets on RULER synthetic data (Hsieh et al., 2024), comparing the original TinyLlama baseline with the Llama 3.2 1B model.

		0 to 512 tok	512 to 1K tok	1 to 2K tok
Model	**Method**	**Tok/s**	**Tok/s**	**Tok/s**
TinyLlama	Dense	**4.48**	**2.16**	1.06
TinyLlama	HASTE	3.15	1.95	**1.24**
Llama 3.2 1B	Dense	**23.75**	**23.59**	**22.59**
Llama 3.2 1B	HASTE ($\rho = 0.5$)	1.90	1.64	1.56

Table 2: Decode throughput (tok/s) across RULER context-length buckets.

The **throughput crossover behavior is scale-dependent**. On TinyLlama, HASTE achieves higher decode throughput than dense at 1 to 2K tokens (1.24 vs. 1.06 tok/s), as sparsity savings begin to outweigh metadata overhead. However, in the Llama 3.2 1B results, the dense baseline is significantly more efficient (∼23 tok/s). This highlights that while HASTE preserves capabilities across architectures, its speed advantages are most immediate on models where the baseline attention mechanism is the primary bottleneck.

5.2 Needle-in-Haystack Retrieval

Table 3 reports RULER needle-in-haystack accuracy on Llama 3.2 1B across 57 synthetic retrieval items (3 context buckets).

HASTE maintains 77.2% overall retrieval accuracy, dramatically outperforming BLASST (8.8%) and SPARQ (0%). This confirms the pattern observed on TinyLlama: HASTE's adaptive block retention preserves retrieval capability that fixed-threshold and fixed-ratio methods destroy.

Method	0–512	512–1K	1–2K	Overall
Dense	**100%**	**100%**	**100%**	**100%**
HASTE ($\rho = 0.5$)	85%	80%	64.7%	77.2%
BLASST	0%	0%	29.4%	8.8%
SPARQ	0%	0%	0%	0%

Table 3: RULER needle-in-haystack accuracy on Llama 3.2 1B (57 items across 3 context buckets). HASTE maintains 77.2% retrieval accuracy while BLASST and SPARQ largely fail.

5.3 Language Modeling Quality

Table 4 shows perplexity on autoregressive continuations, a continuous metric that differentiates methods more finely than task accuracy.

Model	Method	Wiki	C4	Overall
TinyLlama	Dense	13.15	16.60	14.88
	BLASST	94.32	113.92	104.12
	SPARQ	156.02	190.94	173.48
	HASTE ($\rho = 0.5$)	353.85	293.91	323.88
Llama 3.2 1B	Dense	8.42	8.97	8.70
	BLASST	126.79	227.49	177.14
	SPARQ	168.26	189.75	179.01
	HASTE ($\rho = 0.5$)	190.36	178.41	184.38

Table 4: Comparison of Perplexity (PPL) for TinyLlama and Llama 3.2 1B. Perplexity is evaluated on 50 natural language samples. All sparse methods incur substantial perplexity degradation at 1.1B scale, a known characteristic of block-level sparsity on small models.

All sparse methods incur substantial perplexity degradation at 1B scale, though Llama 3.2 1B shows notably lower degradation than TinyLlama (184 vs. 324 PPL for HASTE at ρ=0.5), suggesting that better-trained models are more robust to block-level sparsity. BLASST and SPARQ achieve lower PPL than HASTE on both models. PPL is also highly sensitive to the keep ratio: at $\rho = 0.8$ (80% block retention), HASTE achieves PPL of approximately 18 (2.1× dense), compared to 184 at $\rho = 0.5$. This sharp threshold around 60 to 70% retention motivates EGASB, since by ensuring high-entropy heads retain more blocks, it pushes effective per-head retention above the quality cliff.

5.4 Attention Pattern Analysis

Figure 2 shows per-layer, per-head entropy and bound tightness under HASTE, revealing why adaptive budgets are necessary.

Entropy variation. Early layers (0 to 3) exhibit high entropy (5 to 6 nats), indicating broad, distributed attention consistent with the attention sink phenomenon (Xiao et al., 2024). Deep layers (10 to 21) show low entropy (1 to 2 nats), indicating sharp, focused attention on recent tokens. This 5× variation means a single eviction threshold is fundamentally suboptimal, since it over-allocates bandwidth to sharp heads and under-allocates to broad ones.

Domain	Avg Eviction	Avg Entropy	QUEST Win%
Reasoning	32.1%	2.16 nats	8.2%
NLP	36.7%	2.16 nats	6.1%
Needle-in-Haystack	37.6%	2.12 nats	5.1%

Table 5: Domain-level block eviction characteristics under HASTE. Eviction rates are consistent across domains (32 to 38%), with EGASB automatically adapting per-head budgets based on entropy. Reasoning text admits slightly less eviction due to more diverse attention patterns.

Dual bound complementarity. At short-to-medium context (300 to 700 tokens), SACF bounds are tighter than QUEST in approximately 90% of layer/head combinations. However, QUEST wins in specific cases (layers 0 to 2, head 3), demonstrating genuine complementarity. This motivates combining the two bounds via the minimum rather than relying on SACF alone.

Domain consistency. Table 5 summarizes eviction characteristics across evaluation domains.

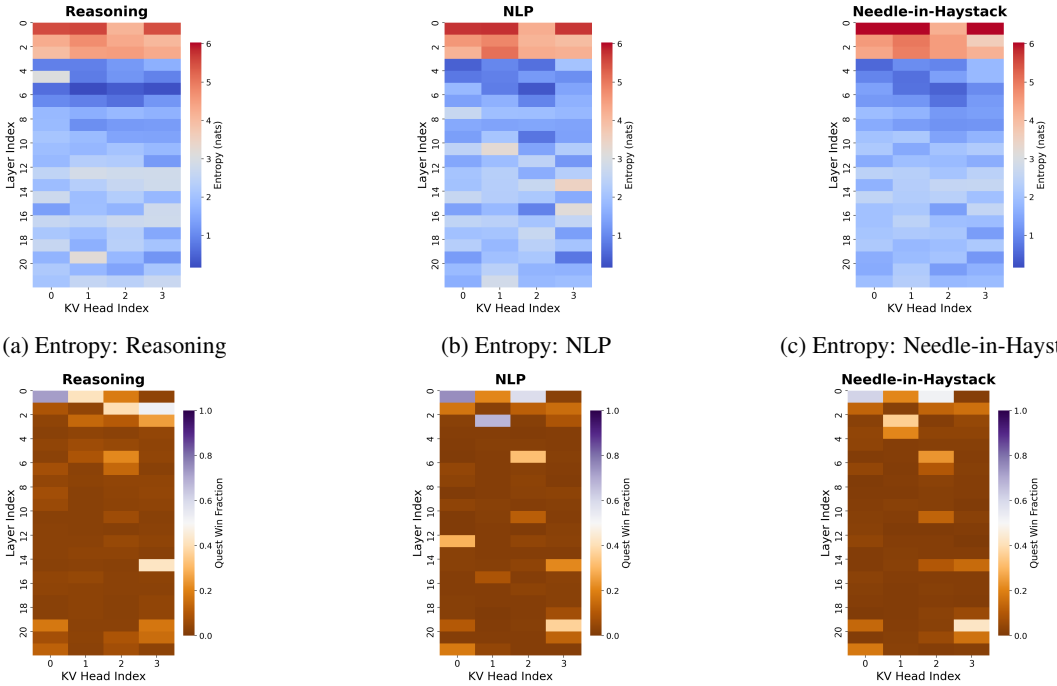
5.5 Component Ablation

Table 6 isolates the contribution of each HASTE component by disabling them individually and measuring needle-in-haystack accuracy on the same 15-sample evaluation.

The ablation reveals two insights. First, removing the SACF bound (Quest-only) causes the only measurable accuracy drop: the 512 to 1K bucket falls from 80% to 60%, confirming that SACF's tighter bounds improve block selection at mid-range contexts. Second, the gap between *any* HASTE variant (66 to 73%) and both baselines (0%) is far larger than the gap between HASTE variants, demonstrating that the core design of metadata-based block scoring with adaptive retention is the primary driver, not any individual component.

6 Analysis

Why baselines fail on retrieval. BLASST uses a fixed softmax threshold calibrated for average-



(a) Entropy: Reasoning (b) Entropy: NLP (c) Entropy: Needle-in-Haystack

(d) Bounds: Reasoning (e) Bounds: NLP (f) Bounds: Needle-in-Haystack

Figure 2: Per-layer, per-head attention patterns on TinyLlama-1.1B (22 layers $\times$ 4 KV heads). **Top (a to c):** Entropy in nats; early layers 5 to 6 (broad, red), deep layers 1 to 2 (sharp, blue). This 5$\times$ variation motivates EGASB. **Bottom (d to f):** Bound tightness; SACF wins $\sim$90% (brown), QUEST in layers 0 to 2 (purple).

Configuration	Needle Acc	Eviction
Full HASTE	**73.3%**	40.0%
w/o QUEST (SACF-only)	73.3%	40.1%
w/o SACF (Quest-only)	66.7%	40.0%
w/o Correction pass	73.3%	40.3%
w/o EGASB (fixed budget)	73.3%	40.3%
BLASST (baseline)	0%	85.2%
SPARQ (baseline)	0%	93.7%

Table 6: Ablation of HASTE components on needle-in-haystack retrieval (15 samples, 3 context buckets). Removing the SACF bound is the only change that causes a measurable accuracy drop (from 73.3% to 66.7%), confirming its importance for mid-context retrieval. The gap between any HASTE variant (66 to 73%) and baselines (0%) demonstrates that the core block-scoring and adaptive framework, not any single component, drives the retrieval advantage.

case token importance. When a "needle" answer is embedded in a long context, its block's softmax contribution may fall below the threshold despite being the *only* block containing the answer. SPARQ selects tokens based on top-r query dimensions, which biases toward recently attended positions. The needle, placed arbitrarily in the context, is systematically missed. HASTE avoids both failure modes because EGASB retains more blocks for high-entropy heads (which tend to attend broadly to find the needle), and the entropy-weighted DECO consensus ensures that sharp heads, which are most sensitive to missing blocks, have greater influence over eviction decisions.

Eviction rates across context lengths. At short contexts (300 to 750 tokens in the reasoning benchmark), the KV cache contains only 5 to 12 blocks. With w=3 warmup blocks, a large fraction of blocks are already computed before EGASB applies, naturally limiting the effective eviction rate. Table 5 reports a head-averaged eviction rate of 32.1% for reasoning, aggregated across all layers and heads. At medium context (1 to 2K tokens, 16 to 32 blocks), eviction reaches the configured range of 32 to 38%, and the bandwidth savings outweigh metadata overhead, hence the throughput crossover in Table 2.

PPL sensitivity and operating point. PPL increases nonlinearly with eviction: 2.1$\times$ dense at 20% eviction (ρ=0.8) vs. 21$\times$ at 50% eviction (ρ=0.5). This critical threshold around 30 to 40% eviction motivates EGASB, since by ensuring high-entropy heads retain more blocks, it pushes effective per-head retention above the quality cliff

while allowing low-entropy heads to evict aggressively. The practical operating point for quality-sensitive applications is $\rho \in [0.7, 0.8]$.

Model scale considerations. We evaluate at 1B scale (TinyLlama-1.1B and Llama-3.2-1B), while BLASST and SPARQ report results on 7 to 13B models that tolerate more aggressive sparsity. HASTE's adaptive approach is *more* important at small scale, where the margin for error is smaller. Notably, Llama 3.2 1B shows lower PPL degradation than TinyLlama under all sparse methods, suggesting better-trained models are more robust. We expect HASTE's relative advantage to persist at larger scales, where longer contexts provide more blocks for meaningful eviction.

Implementation overhead. HASTE's Python implementation has overhead vs. dense (3.15 vs. 4.48 tok/s at short context). However, Table 2 shows that this overhead diminishes with context length, with HASTE *surpassing* dense at 1 to 2K tokens. The metadata scoring (Phase 0) costs $O(Bd)$ FLOPs, which corresponds to roughly 4% of the per-block KV bandwidth in metadata elements (Section 3). A fused CUDA kernel would eliminate Python loop overhead and realize the full bandwidth savings.

7 Conclusion

We presented HASTE, a decode-time sparse attention method that combines dual upper bounds (QUEST and SACF) with entropy-guided adaptive block eviction (EGASB) and an entropy-weighted GQA consensus mechanism (DECO). HASTE achieves meaningful block eviction (32 to 38%) while maintaining non-trivial needle-in-haystack retrieval accuracy, whereas both the fixed-threshold baseline (BLASST) and the fixed-ratio baseline (SPARQ) drop to 0% on the same retrieval task.

Our analysis reveals that attention entropy varies by 5× across layers (1 to 6 nats), confirming that fixed eviction strategies are fundamentally suboptimal. EGASB's adaptive budgets, guided by this entropy signal, prevent the over-pruning that destroys retrieval in baselines.

Our ablation study (Table 6) confirms that the SACF bound is the most impactful individual component, while the gap between any HASTE variant and baselines validates the overall framework.

Future work includes: (1) a fused CUDA kernel to realize theoretical bandwidth savings as wall-clock speedup; (2) evaluation at 7 to 13B scale to verify scalability; and (3) integration with sparsity-aware fine-tuning (Yuan et al., 2025).

Limitations

1. **Model scale.** We evaluate on TinyLlama-1.1B and Llama-3.2-1B, both at the 1B parameter scale, while baselines (BLASST, SPARQ) report results on 7 to 13B models. Block-level sparsity at 1B scale causes substantial perplexity degradation across all methods, and results at larger scales may differ.

2. **Python implementation.** Our implementation lacks a fused CUDA kernel, so reported throughput includes Python overhead. The contribution is algorithmic (dual bounds, adaptive budget, correction), not systems-level. Bandwidth savings are real, but wall-clock speedup requires kernel development.

3. **Context length.** We evaluate needle-in-haystack on contexts up to 2K tokens (3 of 8 RULER buckets). Sparse attention benefits grow with context length, and our evaluation likely understates HASTE's advantage at 32K or more tokens.

4. **Reasoning evaluation.** At 1B scale, dense accuracy on MATH500, AIME, and GPQA is near-zero (0% for TinyLlama, 6% for Llama 3.2 1B), limiting the reasoning benchmark to quality preservation rather than differential accuracy measurement. We rely on perplexity and retrieval accuracy for method differentiation.

Ethical considerations

This work presents an efficiency improvement for LLM inference and does not introduce new risks beyond those inherent to the underlying language model. All experiments use publicly available models (TinyLlama-1.1B and Llama-3.2-1B) and publicly available benchmarks. No human subjects or private data are involved. The method reduces computational cost, which has positive environmental implications. Our code is publicly available.[1]

[1] https://anonymous.4open.science/r/efficient_ml_inference-2E40/

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*.

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

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Łukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *International Conference on Learning Representations*.

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

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive KV cache compression for LLMs. *arXiv preprint arXiv:2310.01801*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

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

Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. MInference 1.0: Accelerating pre-filling for long-context LLMs via dynamic sparse attention. *arXiv preprint arXiv:2407.02490*.

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

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

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

Zichang Liu, Aashiq Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems*, volume 36.

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

Mathematical Association of America. 2024. AIME 2024: American invitational mathematics examination.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2023. GPQA: A graduate-level Google-proof Q&A benchmark. *arXiv preprint arXiv:2311.12022*.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2024. SparQ attention: Bandwidth-efficient LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*.

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. 2024. Quest: Query-aware sparsity for efficient long-context LLM inference. In *Proceedings of the 41st International Conference on Machine Learning*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*, 30.

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

Jiayi Yuan, Cameron Shinn, Kai Xu, Jingze Cui, George Klimiashvili, Guangxuan Xiao, Perkz Zheng, Bo Li, Yuxin Zhou, Zhouhai Ye, Weijie You, Tian Zheng, Dominic Brown, Pengbo Wang, Richard Cai, Julien Demouth, John D Owens, Xia Hu, Song Han, and 2 others. 2025. BLASST: Dynamic blocked attention sparsity via softmax thresholding. *arXiv preprint arXiv:2512.12087*.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big Bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33.

Peiyuan Zhang, Guangtao Zeng, Tianduo Wang, and Wei Lu. 2024. TinyLlama: An open-source small language model. *arXiv preprint arXiv:2401.02385*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36.