CutAttn: Discovering Cognitive Transition Layers for Efficient Long-Context Prefilling

Wentao Liu[♡♣] **Xiabao Wu**[♣] **Yongchao Liu**[♣*] **Haitao Zhang**[♣]
Jiajun Zheng[♣] **Ruiting Zhou**[♡*]

♡ School of Computer Science and Engineering, Southeast University
♣ Ant Group
{liuwentao, ruitingzhou}@seu.edu.cn
{taoyang.lwt, wuxiabao.wxb, yongchao.ly, zht108229, zjj517361}@antgroup.com

Abstract

Long-context LLM inference is fundamentally a retrieval process, transitioning from diffuse to sharply focused attention at a few architecture-determined *cognitive transition layers* (CTLs). Existing acceleration methods leave significant costs unaddressed: sparse attention ignores FFNs, KV cache compression neglects prefill, and hidden-state pruning relies on fixed, heuristic rules. We introduce **CutAttn**, a training-free framework that exploits CTLs via inter-layer Jensen–Shannon divergence, pruning redundant context at chunk granularity while dynamically gated by a normalized-entropy criterion. Operating at the hidden-state level, CutAttn reduces both attention and FFN costs, composing orthogonally with existing acceleration techniques. Evaluated on RULER, InfiniteBench, and LongBench, CutAttn preserves full-attention accuracy while achieving up to a $2.02\times$ prefill speedup on 128K sequences (and $2.81\times$ when composed with sparse attention). Furthermore, our theoretical analysis provides approximation bounds, an entropy-compression limit justifying the gating, and a closed-form speedup ratio.

1 Introduction

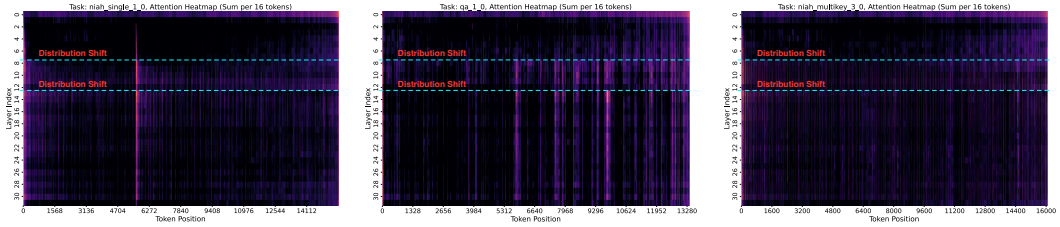
Prefilling a 128K-token sequence on Llama-3.1-8B [1] takes nearly 30 seconds even with FlashAttention-2 [2]. Two factors drive this cost: the quadratic sequence scaling of self-attention, and the linear but massive computation of the feed-forward network (FFN), which alone accounts for roughly 67% of the dense matrix operations per layer. As context windows expand beyond 128K tokens [3, 4], the prefill phase has become the critical bottleneck for long-context LLM inference.

To mitigate these bottlenecks, considerable effort has been devoted to accelerating long-context inference. *Sparse attention* methods [5, 6, 7] reduce attention computation in both prefill and decode via sparse patterns, but leave the sequence length—and thus the heavy FFN cost—unchanged. *KV cache compression* methods [8, 9, 10] evict, merge, or quantize KV pairs to reduce memory footprint, but strictly target the decode stage without accelerating prefill. *Hidden-state pruning* methods [11, 12, 13] physically remove tokens at intermediate layers, reducing both attention and FFN costs. However, they rely on heuristically chosen layers and fixed pruning ratios, blind to the dynamic nature of attention concentration. By forcing the same pruning ratio whether the model has sharply localized its focus or still relies on diffuse context, they risk severe information loss.

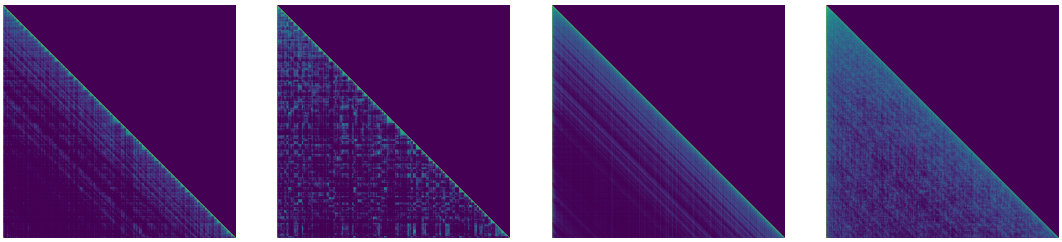
This limitation stems from the absence of a cross-layer perspective on attention evolution. Long-context inference is fundamentally a retrieval process, where the model must identify and aggregate relevant information from a large context; yet existing methods treat each layer in isolation. The

*Corresponding Author.

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



(a) Focused: single-needle retrieval. (b) Multi-focal: question answering. (c) Diffuse: multi-key retrieval.



(d) QA, Layer 12 (e) QA, Layer 13 (CTL) (f) MK, Layer 12 (g) MK, Layer 13 (CTL)

Figure 1: Attention patterns across layers on Llama-3.1-8B-Instruct. **Top**: importance distributions (summed per 16 tokens) for three retrieval modes—(a) focused, (b) multi-focal, (c) diffuse—with dashed lines marking abrupt distribution shifts. **Bottom**: attention matrices before (Layer 12) and at (Layer 13) the CTL for question answering (QA) and multi-key (MK) tasks; the pre-CTL layer shows a diagonal *slash* pattern, while the CTL shifts to *vertical* and *block* patterns, with QA concentrating on few positions and multi-key remaining diffuse.

vertical attention patterns exploited by sparse methods are not artifacts to be pruned—they are the LLM performing context retrieval. As depth increases, attention shifts from diffuse, locally-dependent patterns to sharply concentrated, globally-informed patterns—the model performs *progressive retrieval* internally through its attention layers. As shown in Figure 1, different tasks produce distinct retrieval modes—*focused* (concentrated on one position), *multi-focal* (concentrated on several), and *diffuse* (distributed across many)—yet all share a common structural signature: phase transitions at a few critical layers, where attention abruptly shifts from a diagonal *slash* pattern to *vertical* and *block* patterns. We term these layers *cognitive transition layers* (CTLs). Two properties make CTLs practically useful: (i) their locations are architecture-determined and input-invariant, enabling offline identification; (ii) the degree of concentration at the CTL reflects task difficulty, enabling adaptive pruning. Once the model has localized the relevant context at a CTL, tokens receiving low importance become redundant—removing them physically shortens the sequence, reducing both the quadratic attention cost and the linear FFN cost in all subsequent layers.

Based on these insights, we introduce **CutAttn**, a training-free acceleration framework for long-context LLM inference with three tightly integrated components: (1) **Cognitive transition layer selection**: CutAttn locates optimal pruning layers by detecting peaks in the Jensen-Shannon divergence (JSD) between adjacent layers' importance distributions; layer selection is performed offline once per model and fixed across all inputs. (2) **Chunk-based hidden-state pruning**: context is retained at fixed-length chunk granularity, with the retention ratio determined dynamically by an attention recall target; this physically shortens the sequence, reducing both attention and FFN computation while preserving local semantic coherence. (3) **Entropy-based adaptive gating**: the normalized entropy of the chunk-level importance distribution gates pruning at each CTL—activating only when attention is concentrated, skipping when it is distributed. Crucially, these mechanisms are theoretically grounded: an entropy-compression limit justifies the adaptive gate, and GQA approximation bounds guarantee pruning fidelity. Furthermore, CutAttn is *training-free* and *orthogonally compatible* with existing sparse attention and KV cache compression methods to enable compounded speedups. Our main contributions are:

- We identify *cognitive transition layers* (CTLs): architecture-determined phase transitions in LLM attention that mark the earliest safe pruning points, facilitating zero-overhead offline identification via inter-layer JSD peaks due to their input-invariant nature.

- We propose CutAttn, a training-free framework that exploits CTLs via principled layer selection, attention-recall-guided chunk pruning, and entropy-based adaptive gating—the first hidden-state pruning method to condition pruning on per-input attention concentration.
- We provide formal GQA approximation bounds, an entropy-compression limit justifying the adaptive gate, and a closed-form speedup analysis.
- On RULER, InfiniteBench, and LongBench, CutAttn maintains full-attention accuracy with up to $2.02\times$ prefill speedup, and $2.81\times$ with sparse attention—without retraining.

2 Related Work

Sparse attention. One line of work imposes predefined or learned sparse structures on the attention matrix: sink-based methods retain initial attention sinks with a sliding window [5, 14, 15], while block-routing approaches select relevant key blocks via gating or scoring [16, 17, 18, 19]. A more adaptive line selects KV pairs dynamically per query [7, 20, 21, 6, 22, 23, 24, 25, 26] or retrieves from external memory [27, 28, 29]. These methods accelerate both prefill and decode by skipping attention entries, but leave the sequence length unchanged, so FFN cost is unaffected.

KV cache compression. Token-level eviction removes unimportant KV pairs based on attention weights [8, 30, 31], while budget allocation methods assign different KV budgets across layers or heads [9, 32, 33, 34]—notably, the pyramid information funnel observed by Cai et al. [9] aligns with CutAttn's CTL discovery. Other methods reduce storage through quantization, clustering, or query-selective retention [35, 36, 10, 37, 38, 39]. All these methods operate at the KV-cache level without altering the sequence dimension, so prefill FFN overhead remains unchanged.

Hidden-state pruning. A separate line of work removes tokens from intermediate hidden states, shortening the sequence for all subsequent layers. LazyLLM [11] progressively prunes tokens using per-layer attention scores but selects pruning layers heuristically and applies a fixed pruning schedule. SlimInfer [12] introduces block-wise token-unit importance scoring and CPU KV-cache offloading for decode acceleration, yet still applies fixed retention ratios at heuristically chosen layers. FastKV [13] combines token pruning at early layers with KV cache compression for decode, yet its pruning ratio is fixed regardless of input characteristics. None of these methods accounts for cross-layer attention evolution or per-input concentration, leaving both of CutAttn's key design dimensions unaddressed.

How CutAttn differs. CutAttn is the first method to combine all three capabilities—prefill acceleration, FFN reduction, and input-adaptive pruning with principled layer selection (Table 1)—while remaining orthogonal to both sparse attention and KV cache compression, yielding compounded savings when composed. CutAttn also provides formal GQA approximation bounds and an entropy-compression limit (Theorems 2 and 3), theoretical guarantees absent from all prior hidden-state pruning methods.

Table 1: Method comparison. $\sim$ represents pattern-adaptive only / varies by method.

	Prefill	KV Mem.	FFN	Adaptive	Layer-aw.
Sparse Attn	✓	×	×	$\sim$	×
KV Cache Compr.	×	✓	×	$\sim$	$\sim$
Hidden-State Prun.	✓	✓	✓	×	×
CutAttn	✓	✓	✓	✓	✓

3 CutAttn

We present CutAttn, a training-free framework that accelerates long-context inference by physically pruning redundant context from intermediate hidden states. CutAttn comprises three components, illustrated in Figure 2: (i) a **cognitive transition layer selection** algorithm that identifies optimal pruning layers (§3.2), (ii) a **chunk-based hidden state pruning** mechanism that removes redundant context at the selected layers (§3.3), and (iii) an **entropy-based adaptive gate** that decides per-input whether pruning is safe (§3.4).

3.1 Problem Formulation and Notation

Three-zone input partition. We logically partition the input token sequence into three contiguous functional zones: instruction prefix $\mathbf{X}_{\text{pre}}$, retrieval context $\mathbf{X}_{\text{ctx}}$, and query suffix $\mathbf{X}_{\text{suf}}$, i.e., $\mathbf{X} = [\,\mathbf{X}_{\text{pre}};\ \mathbf{X}_{\text{ctx}};\ \mathbf{X}_{\text{suf}}\,]$. Both $\mathbf{X}_{\text{pre}}$ and $\mathbf{X}_{\text{suf}}$ are always preserved in full (default 256 tokens each);

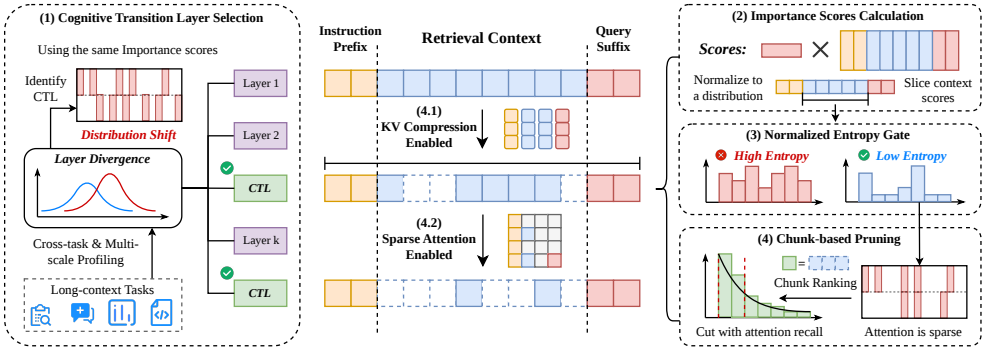


Figure 2: **Overview of CutAttn.** The input sequence is partitioned into three zones: instruction prefix, retrieval context, and query suffix. Token importance scores are computed as the average attention weight from the last w query tokens to each context position. Offline JSD peak detection identifies cognitive transition layers (CTLs). During prefill, at each CTL an entropy gate checks whether attention is sufficiently concentrated; if so, chunk-based pruning physically removes low-importance context chunks from the hidden states, reducing computation for all subsequent layers.

only $\mathbf{X}_{\text{ctx}}$ is subject to pruning. Correspondingly, the hidden states and position indices at layer l decompose as $\mathbf{H}^{(l)} = [\mathbf{H}^{(l)}_{\text{pre}};\ \mathbf{H}^{(l)}_{\text{ctx}};\ \mathbf{H}^{(l)}_{\text{suf}}]$ and $\mathbf{P} = [\mathbf{P}_{\text{pre}};\ \mathbf{P}_{\text{ctx}};\ \mathbf{P}_{\text{suf}}]$.

Token importance score. To quantify how much the model relies on each context token, we define the importance score of token j at layer l as the average attention weight assigned to position j by the last w query tokens (observation window, default w=8):

$$s_j^{(l)} = (H_q w)^{-1} \sum_{h=1}^{H_q} \sum_{i=1}^{w} \text{Softmax}\left(\mathbf{Q}^{(l)}_{w,h} (\mathbf{K}^{(l)}_{g(h)})^\top / \sqrt{d}\right)_{i,j} \tag{1}$$

where H_q is the number of query heads, $\mathbf{Q}^{(l)}_{w,h} \in \mathbb{R}^{w \times d}$ is the query matrix of head h in the observation window, $\mathbf{K}^{(l)}_{g(h)} \in \mathbb{R}^{T \times d}$ is the key matrix of the KV group assigned to head h, T is the sequence length, and d is the head dimension. All subsequent layer-selection analysis and pruning algorithms operate directly on this importance score.

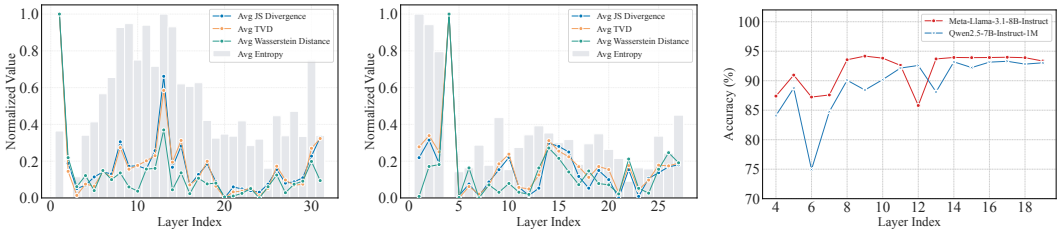
3.2 Cognitive Transition Layer Selection

As motivated in Section 1, CutAttn identifies cognitive transition layers (CTLs) by detecting peaks in the Jensen–Shannon divergence (JSD) between adjacent layers' importance distributions. Specifically, we take the per-layer importance score distribution $\mathbf{d}_l = \{s_j^{(l)}\}_j$ as a macroscopic feature of layer l. The divergence metric at layer l is defined as the JSD between the current layer and its predecessor:

$$d_l = \text{JSD}(\mathbf{d}_l,\ \mathbf{d}_{l-1}), \tag{2}$$

yielding the layer-level divergence sequence $\mathcal{D} = \{d_2, d_3, \ldots, d_N\}$ over $N-1$ layers. We select the top-k most prominent peaks in $\mathcal{D}$ among layers $l \geq l_{\min}$ (default $l_{\min}$=5), excluding shallow layers where the model is still gathering information rather than consolidating it—e.g., Qwen's layer-4 peak (Figure 3b) is skipped; see Appendix B.5 for the full peak ranking.

The selected peaks form the pruning layer set $\mathcal{L}$. Because CTLs reflect **architecture-determined** cognitive shifts, their locations are highly consistent across diverse tasks and context lengths (Appendix B.4). Leveraging this input-invariance, $\mathcal{L}$ **can be computed offline exactly once per model**. This fixed-set approach completely avoids runtime tracking overhead without compromising accuracy. Figure 3 compares multiple distance metrics (JSD, Total Variation Distance, and Wasserstein Distance) that consistently highlight the same prominent peaks. Figure 3c reveals two properties: accuracy surges sharply around each CTL, indicating a cognitive phase transition; yet pruning *before* a CTL causes a severe accuracy drop, confirming that information consolidated at CTLs must be preserved before downstream layers can safely discard redundant context.



(a) Llama-3.1-8B (32 Layers). Peaks at layers 8, 13.

(b) Qwen2.5-7B (28 Layers). Peaks at layers 4, 10, 14.

(c) RULER accuracy when pruning 50% after each layer.

Figure 3: Inter-layer divergence metrics (a, b) and pruning accuracy (c) on RULER (16k). CTLs identified by divergence peaks correspond to layers where pruning becomes safe and efficient.

3.3 Chunk-based Hidden State Pruning

Token-level pruning disrupts local textual coherence and introduces irregular memory access patterns. Instead, CutAttn evaluates and retains context at the granularity of *chunks*, balancing GPU compute efficiency with local semantic integrity.

The retrieval context $\mathbf{X}_{\text{ctx}}$ is divided into M non-overlapping chunks $\{C_1, C_2, \ldots, C_M\}$ of uniform size B (default $B = 64$). The importance score of each chunk is the sum of its constituent token scores: $S(C_i) = \sum_{j \in C_i} s_j^{(l)}$. Rather than retaining a fixed number of chunks, we use an *attention recall* target $\alpha \in (0, 1]$ (default $\alpha = 0.95$). Let $p_i = S(C_i) / \sum_{k=1}^{M} S(C_k)$ denote the normalized importance probability of chunk C_i. We determine the retention count M_r as:

$$M_r = \min\Big\{ m : \sum_{i=1}^{m} p_{\sigma(i)} \;\geq\; \alpha \Big\}, \tag{3}$$

where σ is the permutation sorting chunks by descending probability, such that $p_{\sigma(1)} \geq p_{\sigma(2)} \geq \cdots \geq p_{\sigma(M)}$. The keep-set $\mathcal{I}_{\text{keep}} = \text{Sort}_{\text{asc}}\big(\{\sigma(1), \ldots, \sigma(M_r)\}\big)$ restores the selected chunks to their original sequential order; both $\mathbf{X}_{\text{pre}}$ and $\mathbf{X}_{\text{suf}}$ are unconditionally retained (**dual-end protection**).

At each CTL, we use $\mathcal{I}_{\text{keep}}$ to physically extract the retained chunks from the hidden state and position tensors, yielding $\hat{\mathbf{H}}^{(l)} = \big[\mathbf{H}^{(l)}_{\text{pre}};\ \hat{\mathbf{H}}^{(l)}_{\text{ctx}};\ \mathbf{H}^{(l)}_{\text{suf}}\big]$ and $\hat{\mathbf{P}} = \big[\mathbf{P}_{\text{pre}};\ \hat{\mathbf{P}}_{\text{ctx}};\ \mathbf{P}_{\text{suf}}\big]$, where $\hat{\mathbf{H}}^{(l)}_{\text{ctx}}$ and $\hat{\mathbf{P}}_{\text{ctx}}$ contain only the chunks indexed by $\mathcal{I}_{\text{keep}}$. This directly reduces the sequence dimension, cutting FLOPs for all downstream self-attention and feed-forward computations while preserving the original inter-token distances for RoPE.

Compatibility. CutAttn operates on hidden states rather than KV caches or attention masks, making it orthogonal to both KV cache compression and sparse attention; hidden-state pruning shortens the sequence, directly reducing the candidate pool for downstream KV compression and shrinking the attention matrix for any sparse attention kernel. The complete forward pass is detailed in Algorithm 1 (Appendix A.2).

3.4 Entropy-based Adaptive Pruning

Different tasks exhibit drastically different attention concentration at CTLs (Figure 1). When attention is uniformly distributed, the query depends on broad context and pruning risks severe information loss; only when attention is sharply concentrated can low-scoring chunks be safely discarded. We therefore gate pruning at each CTL with a normalized-entropy criterion.

Using the same chunk-level importance scores $\{S(C_i)\}$ from §3.3, we normalize them into a probability distribution $\mathcal{P}_{\text{chunk}} = \{p_1, p_2, \ldots, p_M\}$ and compute the normalized Shannon entropy:

$$\hat{E} = -(\log M)^{-1} \sum_{i=1}^{M} p_i \log p_i \tag{4}$$

where $\log M$ is the maximum entropy of a uniform distribution over M chunks, constraining $\hat{E} \in [0, 1]$. The gating rule is straightforward: if $\hat{E} < \tau$ (default $\tau = 0.85$), the attention pattern is sparse

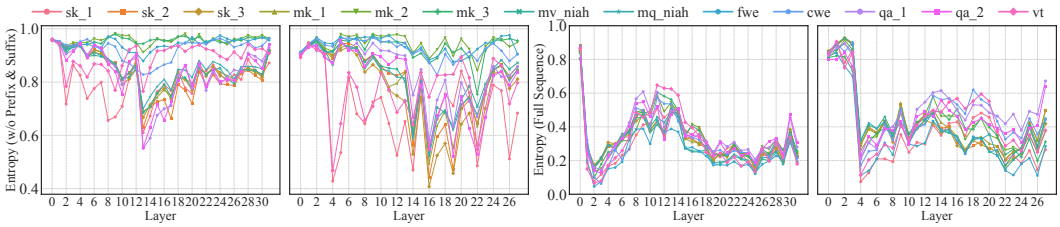


Figure 4: Per-task normalized chunk entropy across layers. Panels 1–2: entropy computed over $\mathbf{X}_{\text{ctx}}$ only; panels 3–4: entropy over the full sequence including $\mathbf{X}_{\text{pre}}$ and $\mathbf{X}_{\text{suf}}$. In each pair, left: Llama-3.1-8B-Instruct; right: Qwen2.5-7B-Instruct-1M.

and pruning is triggered following §3.3; if $\hat{E} \geq \tau$, the attention pattern is dense and pruning is skipped at this layer to avoid information loss.

Figure 4 compares entropy computed with and without $\mathbf{X}_{\text{pre}}$ and $\mathbf{X}_{\text{suf}}$. Stream heads [5, 6, 14, 23] attend predominantly to initial and recent tokens; including them inflates entropy uniformly (panels 3–4), whereas computing entropy solely over $\mathbf{X}_{\text{ctx}}$ isolates retrieval concentration—tasks with concentrated attention show sharply lower entropy after the middle CTL while those with distributed attention remain high (panels 1–2).

3.5 Theoretical Analysis

We provide formal guarantees for CutAttn's pruning mechanism (Appendix A). Theorem 2 establishes a GQA approximation bound: under attention recall α, the per-head output error is at most $2(1-\alpha)\, V_{\max}$, and the full GQA error scales as $2(1-\alpha)\, \|W_O\|_2 \sqrt{(H/G) \sum_g (V_{\max}^{(g)})^2}$. Theorem 3 proves an entropy-compression limit: the normalized entropy $\hat{E}$ lower-bounds the minimum retention ratio ρ achievable at recall α, showing that high entropy mathematically precludes effective compression—justifying the entropy gate. We also provide a window-averaged extension (Remark 1), a multi-layer error propagation analysis (Remark 2), and a closed-form speedup ratio (Proposition 5).

4 Experiments

4.1 Experimental Setup

Experimental setup. We evaluate on Llama-3.1-8B-Instruct [1] (128K context) and Qwen2.5-7B-Instruct-1M [3] (1M context) across three benchmarks: RULER [40] (4K–128K), InfiniteBench [41] (avg. >100K), and LongBench [42] (1.2K–22K). Baselines span token pruning (SlimInfer, FastKV), sparse attention (MInference, FlexPrefill, XAttention), and KV cache compression (SnapKV, PyramidKV, ChunkKV); all use default configurations from their original papers (Appendix B.1). We also evaluate combinations of sparse attention with CutAttn to assess orthogonal compatibility.

Implementation details. We implement CutAttn on top of FlashAttention-2 [2]. To ensure standardized comparisons, all methods are evaluated within the **kvpress** [43] benchmarking framework, with sparse attention baselines integrated via the MInference library [44]. We select the first two CTLs per model (Llama: layers 8, 13; Qwen: layers 10, 14), with τ=0.85, α=0.95, chunk size B=64, and prefix/suffix length 256. All experiments run on a single NVIDIA A100-80GB GPU with BFloat16 precision.

4.2 Main Results

RULER. Table 2 reports accuracy and TTFT across 4K–128K contexts. CutAttn matches or slightly exceeds Full Attention on both models (Llama: 90.07 vs. 90.02; Qwen: 91.27 vs. 91.19) with speedups scaling with context length—up to 2.02× at 128K on Llama and 1.42× on Qwen—and accelerates even at 4K (1.16× / 1.12×), unlike sparse attention methods that introduce overhead at short contexts.

Combining CutAttn with sparse attention yields compounded speedups: at 128K on Llama, XAttention + CutAttn reaches 2.81× (vs. 1.66× alone) and FlexPrefill + CutAttn reaches 2.69× (vs. 1.70× alone); on Qwen, XAttention + CutAttn reaches 2.12× (vs. 1.47×) and MInference + CutAttn reaches 1.57× (vs. 1.16×). The two approaches operate on orthogonal dimensions—sparse attention skips unimportant entries while CutAttn shortens the sequence—so their speedups compose without conflicting. Notably, most sparse methods also achieve higher accuracy when combined with CutAttn (e.g., XAttention on Qwen: 89.17 → 89.29; FlexPrefill on Llama: 89.56 → 89.61), since pruning redundant context at CTLs produces a more compact sequence on which sparse attention operates more effectively.

Table 2: RULER accuracy and TTFT across context lengths. Full Attention reports TTFT in seconds; all other methods report speedup over Full Attention.

	4k		8k		16k		32k		64k		128k		
Method	Acc.	TTFT	Acc.	TTFT	Acc.	TTFT	Acc.	TTFT	Acc.	TTFT	Acc.	TTFT	Avg.
Llama-3.1-8B-Instruct													
Full Attention	96.09	0.31	95.09	0.63	94.20	1.43	88.69	3.47	86.63	9.98	79.43	29.49	90.02
CutAttn	96.01	1.16×	**95.17**	1.23×	**94.23**	1.28×	89.17	1.37×	86.37	1.60×	**79.48**	2.02×	**90.07**
SlimInfer	93.92	1.45×	93.49	1.58×	92.45	1.73×	88.34	1.81×	**86.97**	2.03×	73.69	2.11×	88.14
FastKV	91.13	1.27×	89.84	1.30×	90.58	1.37×	89.23	1.39×	85.76	1.53×	77.70	1.55×	87.37
MInference	**96.09**	0.21×	95.03	0.33×	94.17	0.48×	88.05	0.66×	84.97	1.04×	73.57	1.77×	88.65
FlexPrefill	93.87	0.68×	94.26	0.75×	94.02	0.90×	**91.94**	1.04×	86.30	1.35×	76.97	1.70×	89.56
XAttention	95.77	0.86×	95.00	0.89×	94.08	0.98×	90.29	1.05×	84.93	1.33×	75.16	1.66×	89.20
MInference + CutAttn	96.08	0.22×	95.16	0.36×	94.11	0.54×	87.95	0.80×	85.00	1.29×	74.04	2.37×	88.72
FlexPrefill + CutAttn	94.23	0.79×	94.30	0.91×	93.76	1.08×	91.86	1.34×	86.05	1.85×	77.46	2.69×	89.61
XAttention + CutAttn	95.76	0.97×	95.00	1.08×	93.68	1.19×	89.95	1.39×	85.14	1.88×	75.95	2.81×	89.25
Qwen2.5-7B-Instruct-1M													
Full Attention	95.03	0.28	94.13	0.56	93.46	1.25	91.55	3.00	87.77	7.96	85.19	23.00	91.19
CutAttn	**94.93**	1.12×	**94.13**	1.16×	93.61	1.19×	**91.52**	1.24×	**88.05**	1.30×	**85.37**	1.42×	**91.27**
SlimInfer	87.16	1.40×	87.88	1.55×	87.79	1.65×	86.75	1.77×	84.28	1.88×	74.04	1.96×	84.65
FastKV	81.27	1.24×	80.45	1.29×	80.45	1.33×	78.10	1.40×	74.75	1.45×	73.82	1.49×	78.14
MInference	94.80	0.20×	93.90	0.28×	**93.81**	0.41×	91.40	0.58×	87.86	0.81×	84.76	1.16×	91.09
FlexPrefill	89.31	0.73×	88.77	0.76×	88.08	0.88×	85.45	1.02×	82.22	1.19×	78.54	1.47×	85.39
XAttention	93.66	0.87×	92.52	0.90×	92.02	0.96×	88.74	1.05×	85.79	1.22×	82.32	1.47×	89.17
MInference + CutAttn	94.82	0.23×	94.13	0.31×	93.70	0.46×	91.46	0.69×	88.20	1.00×	84.59	1.57×	91.15
FlexPrefill + CutAttn	90.04	0.77×	88.90	0.81×	88.37	0.97×	84.86	1.18×	81.52	1.47×	77.06	1.99×	85.13
XAttention + CutAttn	93.36	0.98×	92.65	1.02×	92.20	1.12×	89.07	1.27×	85.93	1.53×	82.52	2.12×	89.29

Table 3: InfiniteBench accuracy on nine tasks and TTFT. Full Attention reports TTFT in seconds; all other methods report speedup over Full Attention.

Method	En.QA	En.MC	En.Dia	Zh.QA	Code.D	Math.F	PassKey	Number	KV	Avg.	TTFT
Llama-3.1-8B-Instruct											
Full Attention	36.3	73.9	20.0	40.3	34.2	31.4	100.0	100.0	70.0	56.23	23.95
CutAttn	33.5	71.7	20.0	**39.4**	34.2	34.3	100.0	100.0	**73.0**	**56.23**	1.83×
SlimInfer	32.1	71.7	20.0	**39.4**	34.2	32.9	100.0	100.0	69.0	55.48	2.03×
FastKV	**36.0**	71.7	17.5	38.6	34.2	31.4	100.0	100.0	68.0	55.27	1.49×
MInference	34.2	58.7	**22.5**	26.0	**36.7**	38.6	100.0	98.3	21.0	48.44	1.59×
FlexPrefill	32.9	**73.9**	**22.5**	36.2	31.7	**40.0**	100.0	100.0	59.0	55.13	1.63×
XAttention	34.9	65.2	15.0	32.6	35.4	30.0	100.0	100.0	56.0	52.12	1.55×
MInference + CutAttn	31.9	56.5	**22.5**	26.0	**36.7**	38.6	100.0	98.3	23.0	48.17	2.07×
FlexPrefill + CutAttn	33.5	69.6	15.0	36.2	31.7	38.6	100.0	100.0	61.0	53.96	2.53×
XAttention + CutAttn	34.6	65.2	12.5	36.2	35.4	32.9	100.0	100.0	54.0	52.31	2.54×
Qwen2.5-7B-Instruct-1M											
Full Attention	12.3	78.3	17.5	14.3	39.2	40.0	100.0	100.0	69.0	52.29	20.97
CutAttn	12.3	**80.4**	27.5	**13.9**	39.2	41.4	100.0	100.0	**68.0**	**53.63**	1.37×
SlimInfer	11.3	76.1	25.0	13.1	**41.8**	38.6	100.0	100.0	62.0	51.99	1.97×
FastKV	12.3	**80.4**	25.0	11.6	39.2	42.9	100.0	100.0	31.0	49.16	1.49×
MInference	11.9	78.3	**30.0**	11.7	38.0	40.0	100.0	100.0	63.0	52.54	1.13×
FlexPrefill	9.8	30.4	17.5	12.2	27.9	22.9	100.0	100.0	49.0	41.08	1.46×
XAttention	**13.3**	78.3	15.0	13.2	32.9	44.3	100.0	100.0	61.0	50.89	1.46×
MInference + CutAttn	10.7	78.3	22.5	11.2	36.7	40.0	100.0	100.0	63.0	51.38	1.40×
FlexPrefill + CutAttn	8.4	30.4	15.0	12.0	30.4	21.4	100.0	100.0	52.0	41.07	2.15×
XAttention + CutAttn	12.3	78.3	15.0	12.8	38.0	**45.7**	100.0	100.0	60.0	51.34	1.87×

InfiniteBench & LongBench. On InfiniteBench, CutAttn matches Full Attention on Llama (56.23%) with 1.83× speedup and scores 53.63% vs. 52.29% on Qwen with 1.37×; combining with sparse attention further amplifies speedup (XAttention + CutAttn: 2.54× on Llama). We exclude Code.Run and Math.Calc from the nine-task average because both have near-zero accuracy. On LongBench [42], CutAttn preserves accuracy across all task categories on both models (Llama: 45.41 vs. 45.43; Qwen: 48.98 vs. 48.85); per-task results are in Appendix B.3.

Memory efficiency. Beyond latency, CutAttn also reduces GPU memory during prefill. Table 4 reports peak inference and KV cache memory on Llama. At 128K, inference memory drops by 33.5% (29.47→19.59 GB) and KV cache memory by 41.5% (15.42→9.02 GB); savings grow with context length as longer sequences provide more pruning opportunities, with KV cache benefiting more since it scales directly with sequence length.

Table 4: Peak memory (GB) excluding model parameters on RULER (Llama-3.1-8B).

	4K	8K	16K	32K	64K	128K
Inference Memory						
Full Attn	0.85	1.80	3.66	7.53	15.12	29.47
CutAttn	0.71	1.45	2.92	5.67	10.93	19.59
Saved	16.0%	19.3%	20.2%	24.6%	27.7%	33.5%
KV Cache Memory						
Full Attn	0.44	0.93	1.91	3.93	7.91	15.42
CutAttn	0.38	0.77	1.54	2.94	5.44	9.02
Saved	13.1%	17.9%	19.2%	25.2%	31.2%	41.5%

Task-aware retention. The entropy gate produces input-adaptive retention: concentrated-attention tasks are pruned aggressively (retention as low as 35–40%) while distributed-attention tasks retain more context, validating the gate design (§3.4); detailed per-task retention rates are in Appendix B.2.

4.3 Synergy with KV Cache Compression

CutAttn-KV. Table 5 evaluates CutAttn under extreme KV cache budgets on RULER. CutAttn-KV extends CutAttn by applying chunk-level importance scoring (chunk size 32) at every layer to compress the KV cache. Under both 10% and 20% budgets, CutAttn-KV outperforms all dedicated KV compression baselines by a wide margin: on Qwen at 10% budget, CutAttn-KV averages 85.77% vs. ChunkKV at 72.51% (↑13.26), demonstrating that the chunk-level importance scores learned from attention patterns transfer effectively to KV selection.

Table 5: RULER accuracy under 10% and 20% KV budgets across context lengths. CutAttn-KV applies the same chunk-based importance scores to KV cache pruning.

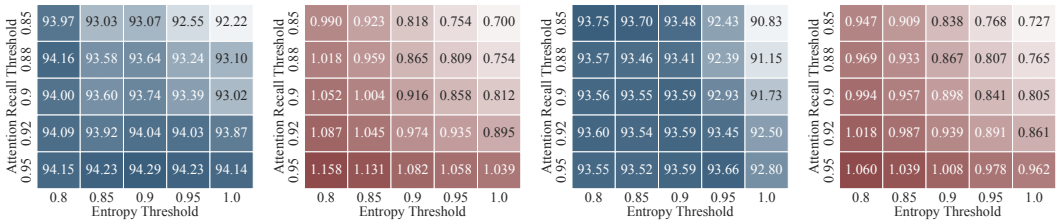
	4k		8k		16k		32k		64k		Avg.	
Method	10%	20%	10%	20%	10%	20%	10%	20%	10%	20%	10%	20%
Llama-3.1-8B-Instruct												
Full Attention (100%)	96.09	96.09	95.09	95.09	94.20	94.20	88.68	88.68	86.63	86.63	92.14	92.14
CutAttn-KV	**89.94**	**94.20**	**89.55**	**92.94**	**88.49**	**91.86**	**85.89**	**87.93**	**84.39**	**86.23**	**87.65**	**90.63**
SnapKV	79.82	81.54	74.77	78.20	76.00	82.23	73.23	81.75	71.54	79.42	75.07	80.63
ChunkKV	82.97	86.64	77.73	84.44	80.99	87.03	79.94	84.31	79.46	83.21	80.22	85.13
PyramidKV	77.87	83.68	79.15	84.54	80.82	84.97	77.21	81.38	76.29	80.56	78.27	83.03
SnapKV + CutAttn	79.61	81.58	75.15	81.64	76.57	84.71	75.11	83.04	73.79	80.96	76.05	82.39
ChunkKV + CutAttn	82.93	86.84	79.45	86.65	82.46	88.65	80.77	85.15	80.03	83.80	81.13	86.22
PyramidKV + CutAttn	78.68	85.15	81.38	86.24	81.92	85.71	77.51	81.87	77.35	80.05	79.37	83.80
Qwen2.5-7B-Instruct-1M												
Full Attention (100%)	95.03	95.03	94.12	94.12	93.46	93.46	91.55	91.55	87.77	87.77	92.39	92.39
CutAttn-KV	**82.06**	**89.90**	**86.48**	**91.23**	**88.24**	**91.30**	**86.80**	**88.89**	**85.28**	**86.53**	**85.77**	**89.57**
SnapKV	53.59	60.50	56.27	67.99	59.54	70.07	65.77	71.50	65.14	71.34	60.06	68.28
ChunkKV	65.41	74.50	70.17	78.32	72.70	80.71	77.12	83.76	77.15	82.03	72.51	79.86
PyramidKV	49.83	59.72	54.56	61.78	56.17	63.60	59.59	65.53	60.06	67.10	56.04	63.55
SnapKV + CutAttn	54.44	63.56	59.94	72.61	61.97	72.87	67.33	74.67	67.24	74.93	62.18	71.73
ChunkKV + CutAttn	67.19	76.40	72.46	82.04	74.12	82.91	78.90	84.21	78.50	82.67	74.23	81.65
PyramidKV + CutAttn	51.48	61.07	55.25	63.71	57.80	65.84	61.24	67.75	62.48	69.36	57.65	65.55

CutAttn boosts existing KV methods. Adding CutAttn consistently improves all three KV compression methods under both budget levels on both models (bottom section of Table 5). The improvement is largest under the most extreme 10% budget: ChunkKV improves from 80.22% to 81.13%

on Llama and from 72.51% to 74.23% on Qwen. By pruning redundant context at CTLs, CutAttn produces a more compact sequence on which downstream KV compression focuses its limited budget on genuinely important positions.

4.4 Ablation Studies

Entropy gate. We ablate the entropy gate by disabling it (τ=1.0, i.e., always pruning) and sweeping (τ, α) on RULER at 16K (Figure 5). Without the gate, accuracy drops sharply on both models (Qwen: $\downarrow$2.92 at α=0.85; Llama: $\downarrow$1.75), confirming that unconditional pruning damages tasks with distributed attention. With the gate enabled, accuracy varies by less than 0.2% across (τ, α) combinations at α=0.95, demonstrating robustness to hyperparameter choice. TTFT decreases as τ increases or α decreases, providing a smooth speed–accuracy trade-off. The entropy gate is thus essential: it allows CutAttn to prune aggressively when attention is concentrated while skipping when it is distributed, without manual per-task tuning.



(a) Llama Acc. (b) Llama TTFT (c) Qwen Acc. (d) Qwen TTFT

Figure 5: Joint effect of entropy threshold τ and attention recall α on RULER accuracy (a, c) and TTFT (b, d) at 16K context. τ=1.0 disables entropy gating.

CTL selection & chunk size. We ablate the choice of pruning layer by pruning 50% of tokens at each layer individually (Figure 6). Pruning before a CTL causes severe accuracy collapse, while pruning at or after a CTL preserves accuracy—confirming that CTLs mark the earliest safe pruning points. Later CTLs yield diminishing speedup since earlier CTLs have already shortened the sequence. Certain tasks are especially sensitive to premature pruning: mk2, mk3, and QA suffer severe accuracy drops when pruning occurs before a CTL, underscoring the importance of accurate CTL detection. Qwen exhibits greater sensitivity to layer selection under the same 50% pruning ratio, reflecting its lower attention sparsity. These results validate that JSD-based CTL detection is essential—without it, pruning at arbitrary layers risks catastrophic information loss. We also ablate the chunk size B on LongBench (Appendix B.6); B=64 achieves the best accuracy–speed trade-off.

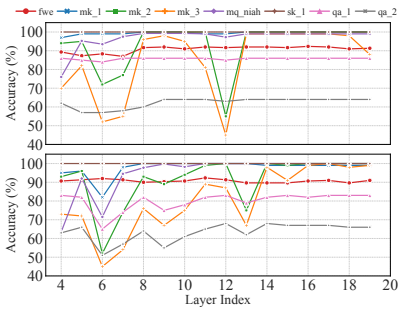


Figure 6: Per-task RULER accuracy after pruning 50% tokens per layer (top: Llama; bottom: Qwen).

4.5 Implementation and Performance on vLLM

We implement CutAttn in vLLM 0.19.1 with packed-batch support: tokens are selected per request and gathered across the batch, with sequence metadata updated after pruning. Prefill uses `torch.compile(dynamic=True)` between pruning boundaries, while scoring and pruning run in eager mode. For decoding, we adapt vLLM's CUDA Graph execution by updating per-layer KV lengths and write-slot mappings in place in fixed-address buffers before graph replay. We evaluate both models on an NVIDIA H20 at 16K/64K/128K contexts, with batch sizes 4/4/2 and 1,024 output tokens per request.

Table 6 reports gains including batching, decoding, and all CutAttn overheads. At 128K, aggregate input-plus-output throughput improves by 1.74×/1.49× on Llama/Qwen, and end-to-end latency improves by 1.67×/1.44×.

Table 6: vLLM performance on H20 with 1,024 output tokens. Metric cells show CutAttn value / speedup over Full Attention. Throughput counts input and output tokens; latency speedup is Full Attention divided by CutAttn.

Model	Context	Batch	Throughput (tok/s)	TTFT (s)	TPOT (ms)	End-to-end (s)
Llama-3.1-8B	16K	4	3057.96 / 1.03×	5.39 / 1.24×	16.2 / 0.96×	22.06 / 1.03×
	64K	4	3789.69 / 1.35×	27.83 / 1.40×	42.6 / 1.26×	71.57 / 1.31×
	128K	2	3552.37 / 1.74×	34.92 / 1.81×	39.3 / 1.54×	75.20 / 1.67×
Qwen2.5-7B	16K	4	3910.94 / 1.01×	5.17 / 1.18×	12.0 / 0.93×	17.46 / 1.01×
	64K	4	4667.85 / 1.25×	24.95 / 1.31×	31.2 / 1.17×	57.06 / 1.23×
	128K	2	4523.79 / 1.49×	33.55 / 1.54×	24.0 / 1.31×	58.23 / 1.44×

5 Conclusion

We identified *cognitive transition layers* (CTLs)—architecture-determined layers where attention undergoes phase transitions—and proposed CutAttn, a training-free framework that detects CTLs via inter-layer JSD peaks, prunes low-importance context chunks using an attention-recall target, and gates pruning with a normalized-entropy criterion. By operating at the hidden-state level, CutAttn reduces both attention and FFN computation, composes orthogonally with sparse attention and KV cache compression, and preserves full-attention accuracy on RULER, InfiniteBench, and LongBench while achieving up to $2.81\times$ prefill speedup when combined with sparse attention. Several limitations remain. First, consistent with prior work on prefill acceleration, we evaluate on 7B–8B models; the additive pruning error at each CTL (Remark 2) may become more pronounced in deeper models requiring more pruning interventions, making extension to 70B+ models an important direction. Second, dynamic sequence lengths introduced by CutAttn may cause load-balancing challenges in distributed inference with Sequence Parallelism; addressing this is left for future work.

Acknowledgments and Disclosure of Funding

This work was supported by Ant Group Research Intern Program. It was also supported in part by the National Key Research and Development Program of China (Grant No. 2024YFB2907100), the National Natural Science Foundation of China (Grant Nos. 62672111 and 62232004), the Collaborative Innovation Center of Novel Software Technology and Industrialization, and the Big Data Computing Center of Southeast University.

References

[1] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

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

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

[4] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

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

[6] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H. Abdi, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu.

Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 37, pages 52481–52515, 2024. doi: 10.52202/079017-1663.

[7] Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. In *International Conference on Machine Learning (ICML)*, pages 47901–47911, 2024.

[8] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H_2O: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

[9] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. In *International Conference on Learning Representations (ICLR)*, 2025.

[10] Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. In *International Conference on Learning Representations (ICLR)*, 2025.

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

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

[13] Dongwon Jo, Jiwon Song, Yulhwa Kim, and Jae-Joon Kim. FastKV: Decoupling of context reduction and KV cache compression for prefill-decoding acceleration. *arXiv preprint arXiv:2502.01068*, 2025.

[14] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[15] Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, Guohao Dai, Huazhong Yang, and Yu Wang. Mixture of attention spans: Optimizing llm inference efficiency with heterogeneous sliding-window lengths. In *International Conference on Learning Representations (ICLR)*, 2025.

[16] 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 *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[17] Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie, Yuxing Wei, Lean Wang, Zhiping Xiao, Yuqing Wang, Chong Ruan, Ming Zhang, Wenfeng Liang, and Wangding Zeng. Native sparse attention: Hardware-aligned and natively trainable sparse attention. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 23078–23097, 2025. doi: 10.18653/v1/2025.acl-long.1126.

[18] Ruyi Xu, Guangxuan Xiao, Haofeng Huang, Junxian Guo, and Song Han. Xattention: Block sparse attention with antidiagonal scoring. In *International Conference on Machine Learning (ICML)*, pages 69819–69831, 2025.

[19] Yu Zhang, Dong Guo, Fang Wu, Guoliang Zhu, Dian Ding, and Yiming Zhang. Anchorattention: Difference-aware sparse attention with stripe granularity. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 8537–8549, 2025. doi: 10.18653/v1/2025.emnlp-main.430.

[20] Yizhao Gao, Zhichen Zeng, DaYou Du, Shijie Cao, Hayden Kwok-Hay So, Ting Cao, Fan Yang, and Mao Yang. Seerattention: Learning intrinsic sparse attention in your llms. In *International Conference on Learning Representations (ICLR)*, 2025.

[21] 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. In *International Conference on Machine Learning (ICML)*, pages 76397–76413, 2025.

[22] Qianchao Zhu, Jiangfei Duan, Chang Chen, Siran Liu, Xiuhong Li, Guanyu Feng, Xin Lv, Xiao Chuanfu, Dahua Lin, and Chao Yang. Sampleattention: Near-lossless acceleration of long context llm inference with adaptive structured sparse attention. In *International Conference on Learning Representations (ICLR)*, 2025.

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

[24] Kan Zhu, Tian Tang, Qinyu Xu, Zhan Jin, Yile Gu, Zhichen Zeng, Rohan Kadekodi, Liangyu Zhao, Ang Li, Arvind Krishnamurthy, and Baris Kasikci. Tactic: Adaptive sparse attention with clustering and distribution fitting for long-context llms. In *International Conference on Learning Representations (ICLR)*, 2025.

[25] Yifei Wang, Yueqi Wang, Zhenrui Yue, Huimin Zeng, Yong Wang, Ismini Lourentzou, Zhengzhong Tu, Xiangxiang Chu, and Julian McAuley. Fasa: Frequency-aware sparse attention. In *International Conference on Learning Representations (ICLR)*, 2026.

[26] Aditya Desai, Kumar Krishna Agrawal, Shuo Yang, Alejandro Cuadron, Luis Gaspar Schroeder, Matei Zaharia, Joseph E. Gonzalez, and Ion Stoica. vattention: Verified sparse attention via sampling. In *International Conference on Learning Representations (ICLR)*, 2026.

[27] 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. In *International Conference on Learning Representations (ICLR)*, 2025.

[28] 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 *International Conference on Learning Representations (ICLR)*, 2025.

[29] Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. Retrievalattention: Accelerating long-context llm inference via vector retrieval. In *International Conference on Learning Representations (ICLR)*, 2025.

[30] Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant J. Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. In *MLSys Conference*, volume 6, pages 114–127, 2024.

[31] 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. In *International Conference on Learning Representations (ICLR)*, 2024.

[32] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. In *International Conference on Learning Representations (ICLR)*, 2025.

[33] Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-Wen Chang, Jiuxiang Gu, Zhen Dong, Anima Anandkumar, Abedelkadir Asi, and Junjie Hu. R-kv: Redundancy-aware kv cache compression for reasoning models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

[34] Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov. Rocketkv: Accelerating long-context llm inference via two-stage kv cache compression. In *International Conference on Machine Learning (ICML)*, 2025.

[35] 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):201:1–201:30, 2025. doi: 10.1145/3725338.

[36] Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo. Clusterkv: Manipulating llm kv cache in semantic space for recallable compression. In *Design Automation Conference (DAC)*, pages 1–7, 2025. doi: 10.1109/DAC63849.2025.11132479.

[37] 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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[38] Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. Magicpig: Lsh sampling for efficient llm generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

[39] Prajwal Singhania, Siddharth Singh, Shwai He, Soheil Feizi, and Abhinav Bhatele. Loki: Low-rank keys for efficient sparse attention. In *International Conference on Learning Representations (ICLR)*, 2025.

[40] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? In *International Conference on Learning Representations (ICLR)*, 2025.

[41] Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. ∞ bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15262–15277, 2024.

[42] Yushi Bai, Shangqing Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1942–1963, 2024. doi: 10.18653/v1/2024.acl-long.172.

[43] NVIDIA. kvpress: A library for kv cache compression methods and benchmarks. `https://github.com/NVIDIA/kvpress`, 2025.

[44] Microsoft. Minference: Million-tokens prompt inference for long-context llms. `https://github.com/microsoft/minference`, 2025.

A Theoretical Analysis

This appendix provides the complete CutAttn algorithm, extended theoretical results, and all proofs.

A.1 Peak Prominence for CTL Selection

As described in §3.2, we rank peaks in the inter-layer JSD sequence $\mathcal{D}$ by their *prominence*. The prominence of the i-th peak is defined as:

$$\pi_i = d_i - \max(V_{\text{left}},\, V_{\text{right}}), \tag{5}$$

where d_i is the divergence at peak i, and V_{left} and V_{right} represent the deepest valleys between peak i and the nearest higher peak (or the sequence boundary) to its left and right, respectively. Prominence measures how much a peak stands out relative to its surrounding baseline, making it robust to the absolute magnitude of the divergence signal.

A.2 CutAttn Forward Pass

Algorithm 1 integrates the three components of CutAttn: cognitive transition layer selection (§3.2), chunk-based hidden-state pruning (§3.3), and entropy-based adaptive gating (§3.4).

Algorithm 1 CutAttn Forward Pass with Adaptive Pruning

Require: Input tokens $\mathbf{X} = [\mathbf{X}_{\text{pre}};\, \mathbf{X}_{\text{ctx}};\, \mathbf{X}_{\text{suf}}]$, model with N layers, pruning layers $\mathcal{L}$ (selected offline via §3.2 with $l_{\min}$=5), observation window size w, chunk size B, attention recall α, entropy threshold τ

Ensure: Output logits $\mathbf{Y}$

$\mathbf{H}^{(0)} \leftarrow \text{Embed}(\mathbf{X})$; $\mathbf{P} \leftarrow \text{PositionIndices}(\mathbf{X})$
for $l = 1$ **to** N **do**
  $\mathbf{H}^{(l)} \leftarrow \text{TransformerLayer}_l(\mathbf{H}^{(l-1)}, \mathbf{P})$
  if $l \in \mathcal{L}$ **then**
    Obtain $\mathbf{Q}_w^{(l)}$ (last w queries) and $\mathbf{K}^{(l)}$ from the attention module of $\text{TransformerLayer}_l$
    Compute importance scores $\{s_j^{(l)}\}_{j \in \mathbf{X}_{\text{ctx}}}$ via Eq. (1)
    Partition $\mathbf{X}_{\text{ctx}}$ into chunks $\{C_1, \ldots, C_M\}$ of size B
    $S(C_i) \leftarrow \sum_{j \in C_i} s_j^{(l)}$ for each chunk C_i
    Normalize **context chunk scores** to a probability distribution: $p_i \leftarrow S(C_i) / \sum_{k=1}^{M} S(C_k)$
    $\hat{E} \leftarrow -\sum_i p_i \log p_i \;/\; \log M$ {Normalized entropy (Eq. 4)}
    if $\hat{E} < \tau$ **then**
      Sort chunks by p_i descending; let σ be the sorted permutation
      $M_r \leftarrow \min\{m : \sum_{i=1}^{m} p_{\sigma(i)} \geq \alpha\}$ {Attention recall (Eq. 3)}
      $\mathcal{I}_{\text{keep}} \leftarrow \text{Sort}_{\text{asc}}(\{\sigma(1), \ldots, \sigma(M_r)\})$
      $\mathbf{H}^{(l)} \leftarrow [\mathbf{H}_{\text{pre}}^{(l)};\, \hat{\mathbf{H}}_{\text{ctx}}^{(l)};\, \mathbf{H}_{\text{suf}}^{(l)}]$
      $\mathbf{P} \leftarrow [\mathbf{P}_{\text{pre}};\, \hat{\mathbf{P}}_{\text{ctx}};\, \mathbf{P}_{\text{suf}}]$
    end if
  end if
end for
return $\text{LMHead}(\mathbf{H}^{(N)})$

A.3 Per-Head Approximation Bound

Lemma 1 (Attention Output Approximation Bound)**.** *Let $V_{\max} = \max_{1 \leq j \leq T} \|v_j\|_2$. If the attention recall over the keep-set satisfies $\beta \geq \alpha$, then the per-head attention output error is bounded by*

$$\|o - \hat{o}\|_2 \;\leq\; 2(1-\alpha)\, V_{\max}\,. \tag{6}$$

Proof. Consider a single attention head at a layer following a cognitive transition layer (CTL). Let $q \in \mathbb{R}^d$ be the query vector, and let $\{k_j, v_j\}_{j=1}^{T}$ be the key-value pairs for the full sequence. The full

attention output is

$$o = \sum_{j=1}^{T} a_j \, v_j, \quad a_j = \frac{\exp(q^\top k_j/\sqrt{d})}{\sum_{j'=1}^{T} \exp(q^\top k_{j'}/\sqrt{d})} \, . \tag{7}$$

After CutAttn prunes hidden states at the preceding CTL, only tokens in the keep-set $\mathcal{K} \subset \{1, \ldots, T\}$ are retained. The pruned layer recomputes attention from scratch over the reduced token set. Crucially, for $j \in \mathcal{K}$, the hidden states (and hence k_j, v_j) are identical to the full-attention case, because CutAttn preserves retained tokens' hidden states exactly. The query q is also unchanged, since query tokens (in $\mathbf{X}_{\text{suf}}$) are always retained.

The pruned attention weights are

$$\hat{a}_j = \frac{\exp(q^\top k_j/\sqrt{d})}{\sum_{j' \in \mathcal{K}} \exp(q^\top k_{j'}/\sqrt{d})} = \frac{a_j \cdot Z}{\sum_{j' \in \mathcal{K}} a_{j'} \cdot Z} = \frac{a_j}{\beta} \, , \quad j \in \mathcal{K}, \tag{8}$$

where $Z = \sum_{j'=1}^{T} \exp(q^\top k_{j'}/\sqrt{d})$ is the full softmax normalizer, and $\beta = \sum_{j \in \mathcal{K}} a_j$ is the attention recall. Equation (8) establishes that recomputing softmax over the reduced set is *exactly equivalent* to renormalizing the original attention weights over $\mathcal{K}$.

The pruned output is therefore

$$\hat{o} = \sum_{j \in \mathcal{K}} \hat{a}_j \, v_j = \frac{1}{\beta} \sum_{j \in \mathcal{K}} a_j \, v_j \, . \tag{9}$$

If $\beta = 1$, then $\mathcal{K} = \{1, \ldots, T\}$ and $o = \hat{o}$ trivially; the bound holds with both sides equal to zero. For $\beta < 1$, define the weighted centroids of the kept and discarded value vectors:

$$\bar{v}_{\mathcal{K}} = \frac{1}{\beta} \sum_{j \in \mathcal{K}} a_j \, v_j, \qquad \bar{v}_{\mathcal{D}} = \frac{1}{1-\beta} \sum_{j \notin \mathcal{K}} a_j \, v_j \, . \tag{10}$$

Both are convex combinations of value vectors and hence lie within the convex hull of $\{v_j\}_{j=1}^{T}$.

The full output decomposes as

$$o = \beta \, \bar{v}_{\mathcal{K}} + (1-\beta) \, \bar{v}_{\mathcal{D}} \, , \tag{11}$$

and the pruned output is $\hat{o} = \bar{v}_{\mathcal{K}}$.

The approximation error is thus

$$\begin{aligned} o - \hat{o} &= \beta \, \bar{v}_{\mathcal{K}} + (1-\beta) \, \bar{v}_{\mathcal{D}} - \bar{v}_{\mathcal{K}} \\ &= (1-\beta)\big(\bar{v}_{\mathcal{D}} - \bar{v}_{\mathcal{K}}\big). \end{aligned} \tag{12}$$

Taking norms and using the triangle inequality:

$$\begin{aligned} \|o - \hat{o}\|_2 &= (1-\beta) \, \|\bar{v}_{\mathcal{D}} - \bar{v}_{\mathcal{K}}\|_2 \\ &\leq (1-\beta)\big(\|\bar{v}_{\mathcal{D}}\|_2 + \|\bar{v}_{\mathcal{K}}\|_2\big) \\ &\leq (1-\beta) \cdot 2 \, V_{\max} \\ &\leq 2(1-\alpha) \, V_{\max} \, , \end{aligned} \tag{13}$$

where the second inequality uses $\|\bar{v}_{\mathcal{K}}\|_2 \leq V_{\max}$ and $\|\bar{v}_{\mathcal{D}}\|_2 \leq V_{\max}$ (both are convex combinations of vectors with norm at most $V_{\max}$), and the last inequality uses $\beta \geq \alpha$. □

Tightness. The bound is exactly tight. Consider the case where all kept tokens have value $v_{\mathcal{K}} = V_{\max} \, e_1$ and all discarded tokens have value $v_{\mathcal{D}} = -V_{\max} \, e_1$, with $\beta = \alpha$. Then $\|o - \hat{o}\|_2 = 2(1-\alpha)V_{\max}$, achieving the bound with equality.

A.4 Window-Averaged Extension

Remark 1 (Window-Averaged Importance Scores)**.** *CutAttn selects the keep-set using importance scores averaged over the last w query tokens: $s_j = \frac{1}{w}\sum_{i=1}^{w} a_{i,j}$. The recall guarantee $\sum_{j\in\mathcal{K}} s_j \geq \alpha$ implies*

$$\frac{1}{w}\sum_{i=1}^{w}\beta_i \;\geq\; \alpha\,, \tag{14}$$

where $\beta_i = \sum_{j\in\mathcal{K}} a_{i,j}$ is the per-query recall of the i-th observation token. Applying Lemma 1 to each query and averaging yields

$$\frac{1}{w}\sum_{i=1}^{w}\|o_i - \hat{o}_i\|_2 \;\leq\; 2(1-\bar{\beta})\,V_{\max} \;\leq\; 2(1-\alpha)\,V_{\max}\,, \tag{15}$$

where $\bar{\beta} = \frac{1}{w}\sum_{i=1}^{w}\beta_i$. Thus the window-averaged pruning strategy preserves the same worst-case bound on mean *output error across the observation window. Since these w tokens sit at the prompt suffix and encode the model's retrieval intent, bounding their average error ensures that the downstream generation context remains stable.*

Derivation. CutAttn computes importance scores by averaging attention weights over the last w query tokens in the observation window:

$$s_j = \frac{1}{w}\sum_{i=1}^{w} a_{i,j}\,, \tag{16}$$

where $a_{i,j}$ is the attention weight of the i-th observation-window query on key j. The keep-set $\mathcal{K}$ is chosen to satisfy the aggregate recall constraint $\sum_{j\in\mathcal{K}} s_j \geq \alpha$.

Expanding the left-hand side by interchanging the (finite) sums:

$$\sum_{j\in\mathcal{K}} s_j = \sum_{j\in\mathcal{K}}\frac{1}{w}\sum_{i=1}^{w} a_{i,j} = \frac{1}{w}\sum_{i=1}^{w}\sum_{j\in\mathcal{K}} a_{i,j} = \frac{1}{w}\sum_{i=1}^{w}\beta_i = \bar{\beta}\,, \tag{17}$$

where $\beta_i = \sum_{j\in\mathcal{K}} a_{i,j}$ is the attention recall of the i-th query token individually, and $\bar{\beta}$ is their arithmetic mean. The recall constraint thus reads $\bar{\beta} \geq \alpha$.

By the penultimate line of Eq. (13) (before applying $\beta \geq \alpha$), each query satisfies $\|o_i - \hat{o}_i\|_2 \leq 2(1-\beta_i)\,V_{\max}$. Averaging over the observation window:

$$\begin{aligned}\frac{1}{w}\sum_{i=1}^{w}\|o_i - \hat{o}_i\|_2 &\leq \frac{1}{w}\sum_{i=1}^{w} 2(1-\beta_i)\,V_{\max} \\ &= 2\left(1 - \frac{1}{w}\sum_{i=1}^{w}\beta_i\right)V_{\max} = 2(1-\bar{\beta})\,V_{\max} \leq 2(1-\alpha)\,V_{\max}\,.\end{aligned} \tag{18}$$

The interchange of summation in Eq. (17) is valid because both sums are finite. The final inequality follows from $\bar{\beta} \geq \alpha$.

A.5 GQA Approximation Bound

Theorem 2 (Grouped-Query Attention Approximation Bound)**.** *In GQA with G key-value groups and H query heads, let $V_{\max}^{(g)}$ denote the value-vector norm bound for group g. If every query head satisfies $\beta_h \geq \alpha$, then the full GQA output error is bounded by*

$$\left\|\mathrm{GQA}(q) - \widehat{\mathrm{GQA}}(q)\right\|_2 \;\leq\; 2(1-\alpha)\,\|W_O\|_2\sqrt{\frac{H}{G}\sum_{g=1}^{G}\left(V_{\max}^{(g)}\right)^2}\,. \tag{19}$$

When $G = H$ (standard MHA), the bound reduces to $2(1-\alpha)\,\|W_O\|_2\sqrt{\sum_{h=1}^{H}(V_{\max}^{(h)})^2}$.

With default $\alpha = 0.95$, the per-head error is at most $0.1\,V_{\max}$; the bound remains valid with α replaced by $\min_h \beta_h$ when a shared keep-set causes per-head recall to differ.

Proof. In grouped-query attention (GQA), the H query heads are partitioned into G groups of $M = H/G$ heads each. Within group g, all M query heads share a single set of key-value pairs, so the value-vector norm bound $V_{\max}^{(g)} = \max_j \|v_j^{(g)}\|_2$ is common to all heads in the group.

Let $\Delta o_{g,m}$ denote the per-head output error of the m-th query head in group g. Since each head computes its own attention weights (only the keys and values are shared), Lemma 1 applies independently:

$$\|\Delta o_{g,m}\|_2 \le 2(1-\alpha)\, V_{\max}^{(g)}, \quad \forall\, g \in \{1,\dots,G\},\ m \in \{1,\dots,M\}\,. \tag{20}$$

The GQA output concatenates all $H = G \cdot M$ head outputs and applies W_O:

$$\mathrm{GQA}(q) = W_O\,[\,o_{1,1};\ \dots;\ o_{1,M};\ o_{2,1};\ \dots;\ o_{G,M}\,]\,. \tag{21}$$

Let $\Delta = [\Delta o_{1,1};\ \dots;\ \Delta o_{G,M}] \in \mathbb{R}^{d_{\mathrm{model}}}$. The concatenation norm is

$$\begin{aligned}\|\Delta\|_2 &= \sqrt{\sum_{g=1}^{G}\sum_{m=1}^{M}\|\Delta o_{g,m}\|_2^2} \le \sqrt{\sum_{g=1}^{G}\sum_{m=1}^{M}\big(2(1-\alpha)\,V_{\max}^{(g)}\big)^2} \\ &= 2(1-\alpha)\sqrt{\sum_{g=1}^{G} M\,\big(V_{\max}^{(g)}\big)^2} = 2(1-\alpha)\sqrt{\frac{H}{G}\sum_{g=1}^{G}\big(V_{\max}^{(g)}\big)^2}\,.\end{aligned} \tag{22}$$

Applying the submultiplicativity of the spectral norm:

$$\big\|\mathrm{GQA}(q) - \widehat{\mathrm{GQA}}(q)\big\|_2 = \|W_O\,\Delta\|_2 \le \|W_O\|_2\,\|\Delta\|_2 \le 2(1-\alpha)\,\|W_O\|_2\sqrt{\frac{H}{G}\sum_{g=1}^{G}\big(V_{\max}^{(g)}\big)^2}\,. \tag{23}$$

Degenerate cases.

- **Standard MHA** ($G = H$, $M = 1$): each group contains exactly one head, so $V_{\max}^{(g)} = V_{\max}^{(h)}$. The bound becomes $2(1-\alpha)\,\|W_O\|_2\sqrt{\sum_{h=1}^{H}(V_{\max}^{(h)})^2}$, recovering the standard MHA bound.
- **Multi-Query Attention** ($G = 1$, $M = H$): all heads share one KV pair with bound $V_{\max}$. The bound simplifies to $2(1-\alpha)\,\|W_O\|_2\,\sqrt{H}\,V_{\max}$.

□

A.6 Multi-Layer Error Propagation

Remark 2 (Multi-Layer Error Propagation)**.** *Within each segment between consecutive CTLs, the token set is fixed and each Transformer layer applies a residual mapping $H^{(l+1)} = H^{(l)} + g_l(H^{(l)})$. In this fixed-token regime, an approximation error ϵ at the segment boundary propagates as $\epsilon_{l_0+n} \le \epsilon\prod_{i=l_0+1}^{l_0+n}(1+\gamma_i)$, where γ_i is the Lipschitz constant of g_i and l_0 is the first layer of the segment. LayerNorm and bounded activations keep γ_i moderate. At each subsequent CTL, an additional pruning error is introduced; these per-CTL errors accumulate additively. CutAttn's CTL placement ensures that attention patterns have stabilized post-pruning (Figure 3), further limiting amplification.*

A.7 Entropy-Compression Limit

Theorem 3 (Entropy-Compression Limit)**.** *Let $\mathcal{P} = (p_1,\dots,p_M)$ be the chunk-level importance distribution (Eq. 1), $\rho = M_r/M$ the retention ratio, and α the attention recall target. If $\alpha \ge \rho$, then the normalized Shannon entropy $\hat{E} = H(\mathcal{P})/\log M$ satisfies*

$$\hat{E} \;\le\; 1 + \frac{\alpha\log\rho \;+\; (1-\alpha)\log(1-\rho) \;+\; H_b(\alpha)}{\log M}\,, \tag{24}$$

where $H_b(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$ is the binary entropy. The bound is tight: it is achieved by the piecewise-uniform distribution that assigns probability $\alpha/(\rho M)$ to each retained chunk and $(1-\alpha)/((1-\rho)M)$ to each discarded chunk.

For default $\alpha = 0.95$ and $M = 2048$, the bound yields $\tau_{\max} \approx 0.87$ at target retention $\rho_0 = 0.3$, making CutAttn's default $\tau = 0.85$ a safe choice; at the extreme $\hat{E} = 1$ (uniform), the bound forces $\rho \geq \alpha$, confirming that pruning a near-uniform distribution is futile.

The proof uses the *maximum entropy principle under constraints*: among all distributions consistent with a given compression outcome, the one with the highest entropy provides the tightest upper bound.

Proof. Let $\mathcal{P} = (p_1, \ldots, p_M)$ be a probability distribution over M chunks. Suppose a pruning procedure selects a subset $\mathcal{K} \subset \{1, \ldots, M\}$ of size $|\mathcal{K}| = M_r = \rho M$ such that

$$\sum_{j \in \mathcal{K}} p_j \;\geq\; \alpha\,. \tag{25}$$

We seek the maximum possible Shannon entropy $H(\mathcal{P}) = -\sum_{j=1}^{M} p_j \log p_j$ over all distributions $\mathcal{P}$ for which such a subset $\mathcal{K}$ exists.

Partition $\{1, \ldots, M\}$ into the retained set $\mathcal{K}$ (size ρM) and the discarded set $\mathcal{D}$ (size $(1-\rho)M$), with

$$\sum_{j \in \mathcal{K}} p_j = \alpha, \qquad \sum_{j \in \mathcal{D}} p_j = 1 - \alpha\,. \tag{26}$$

We set the retained mass to exactly α because increasing it beyond α only concentrates probability on $\mathcal{K}$, reducing entropy; thus the maximum is attained at the boundary.

Within each group, the entropy is maximized when probability is spread uniformly (by the strict concavity of $-x \log x$ under a fixed-sum constraint). The maximum-entropy distribution is therefore

$$p_j^* = \begin{cases} \dfrac{\alpha}{\rho M} & j \in \mathcal{K}, \\[2ex] \dfrac{1-\alpha}{(1-\rho)M} & j \in \mathcal{D}. \end{cases} \tag{27}$$

Substituting into the entropy formula:

$$\begin{aligned} H_{\max} &= -\rho M \cdot \frac{\alpha}{\rho M} \log \frac{\alpha}{\rho M} \;-\; (1-\rho)M \cdot \frac{1-\alpha}{(1-\rho)M} \log \frac{1-\alpha}{(1-\rho)M} \\ &= -\alpha\big(\log \alpha - \log(\rho M)\big) - (1-\alpha)\big(\log(1-\alpha) - \log((1-\rho)M)\big) \\ &= -\alpha \log \alpha + \alpha \log \rho + \alpha \log M \\ &\quad - (1-\alpha)\log(1-\alpha) + (1-\alpha)\log(1-\rho) + (1-\alpha)\log M \\ &= \log M + \alpha \log \rho + (1-\alpha)\log(1-\rho) + H_b(\alpha)\,, \end{aligned} \tag{28}$$

where $H_b(\alpha) = -\alpha \log \alpha - (1-\alpha)\log(1-\alpha)$.

For any distribution $\mathcal{P}$ satisfying the compression constraint, $H(\mathcal{P}) \leq H_{\max}$. Dividing both sides by $\log M > 0$:

$$\hat{E} = \frac{H(\mathcal{P})}{\log M} \;\leq\; 1 + \frac{\alpha \log \rho + (1-\alpha)\log(1-\rho) + H_b(\alpha)}{\log M}\,. \tag{29}$$

The distribution $\mathcal{P}^*$ in Eq. (27) achieves this bound with equality, so the bound is tight. □

Corollary 4 (Maximum Entropy for Feasible Pruning)**.** *Given target retention ratio ρ_0 and recall α, define $\tau_{\max} = 1 + [\alpha \log \rho_0 + (1-\alpha)\log(1-\rho_0) + H_b(\alpha)]/\log M$. If $\hat{E} > \tau_{\max}$, no subset of $\rho_0 M$ chunks can achieve recall α.*

Proof of Corollary 4. Fix target retention ratio ρ_0 and recall α. By Eq. (29), any distribution $\mathcal{P}$ that can be compressed to ratio $\rho \leq \rho_0$ at recall α satisfies $\hat{E} \leq \tau_{\max}(\rho_0, \alpha)$. Taking the contrapositive: if $\hat{E} > \tau_{\max}$, then no subset of size $\rho_0 M$ achieves recall α.

For default $\alpha = 0.95$ and $M = 2048$ (128K context with 64-token chunks), target retention $\rho_0 = 0.3$ yields $\tau_{\max} \approx 0.87$; CutAttn's default $\tau = 0.85 < \tau_{\max}$ leaves a safety margin.

Extreme-case verification.

- **Uniform distribution** ($\hat{E} = 1$): Setting $\hat{E} = 1$ in Eq. (29) requires $\alpha \log \rho + (1-\alpha)\log(1-\rho) + H_b(\alpha) = 0$. The function $f(\rho) = \alpha \log \rho + (1-\alpha)\log(1-\rho) + H_b(\alpha)$ achieves its unique maximum of 0 at $\rho = \alpha$ (verified by $f'(\rho) = \alpha/\rho - (1-\alpha)/(1-\rho) = 0$ iff $\rho = \alpha$). Hence $\rho \geq \alpha$: retaining 95% recall requires retaining at least 95% of chunks.
- **Concentrated distribution** ($\hat{E} \to 0$): Since $\log \rho < 0$ and $H_b(\alpha) > 0$, the bound permits $\rho \to 0$ as $\hat{E} \to 0$: a highly concentrated distribution can be compressed to a vanishing fraction of chunks.
- **Numerical example** ($\alpha = 0.95$, $M = 2048$, $\rho_0 = 0.3$): $\tau_{\max} = 1 + [0.95 \ln 0.3 + 0.05 \ln 0.7 + H_b(0.95)]/\ln 2048 = 1 + [-1.144 - 0.018 + 0.199]/7.625 = 1 - 0.126 = 0.874$. CutAttn's default $\tau = 0.85 < 0.874$ provides a safety margin.

A.8 Speedup Analysis

Let k CTLs partition the N-layer model into $k+1$ segments with effective sequence lengths $T_0 = T > T_1 > \cdots > T_k$. Each layer incurs compute $F(T') = c_a\, T'^2 + c_f\, T'$, where c_a captures attention and c_f captures linear projections and FFN.

Proposition 5 (CutAttn Speedup Ratio)**.** *Setting $l_0 = 0$ and $l_{k+1} = N$, the theoretical speedup ratio is*

$$R = \frac{N \cdot F(T)}{\sum_{i=0}^{k} (l_{i+1} - l_i)\, F(T_i)}\,. \tag{30}$$

In the attention-dominated regime ($c_a T \gg c_f$), this simplifies to $R \approx N \big/ \sum_{i=0}^{k} (l_{i+1} - l_i)(T_i/T)^2$; in the FFN-dominated regime ($c_f \gg c_a T$), $R \approx N \big/ \sum_{i=0}^{k} (l_{i+1} - l_i)(T_i/T)$.

Proof. Consider a Transformer with N layers and k CTLs at positions $l_1 < l_2 < \cdots < l_k$. Define $l_0 = 0$ and $l_{k+1} = N$. After the i-th CTL, the effective sequence length reduces from T_{i-1} to T_i, with $T_0 = T$ (the original length). Each Transformer layer processing a sequence of length T' incurs compute (counted as multiply-accumulate operations; the speedup ratio is independent of the counting convention):

$$F(T') = \underbrace{P_{\text{proj}}\, T'}_{\substack{Q,K,V,O \text{ projections}}} + \underbrace{2\, T'^2 d_{\text{model}}}_{\text{attention matrix}} + \underbrace{P_{\text{ffn}}\, T'}_{\text{FFN}} = c_a\, T'^2 + c_f\, T'\,, \tag{31}$$

where $c_a = 2\, d_{\text{model}}$, $c_f = P_{\text{proj}} + P_{\text{ffn}}$, and P_{proj}, P_{ffn} denote the per-layer parameter counts of the attention projections and the feed-forward network, respectively.

The total FLOPs for the full (unpruned) model are $\text{FLOPs}_{\text{full}} = N \cdot F(T)$.

With CutAttn, layers in the i-th segment (from $l_i + 1$ to l_{i+1}) process sequence length T_i. The total FLOPs are

$$\text{FLOPs}_{\text{CutAttn}} = \sum_{i=0}^{k} (l_{i+1} - l_i)\, F(T_i)\,. \tag{32}$$

The speedup ratio is therefore

$$R = \frac{\text{FLOPs}_{\text{full}}}{\text{FLOPs}_{\text{CutAttn}}} = \frac{N \cdot F(T)}{\sum_{i=0}^{k} (l_{i+1} - l_i)\, F(T_i)}\,. \tag{33}$$

Attention-dominated regime. When $c_a T \gg c_f$ (long sequences), $F(T') \approx c_a T'^2$, so

$$R \approx \frac{N \cdot T^2}{\sum_{i=0}^{k} (l_{i+1} - l_i)\, T_i^2} = \frac{N}{\sum_{i=0}^{k} (l_{i+1} - l_i)\, (T_i/T)^2}\,. \tag{34}$$

FFN-dominated regime. When $c_f \gg c_a T$ (moderate-length sequences where FFN cost dominates), $F(T') \approx c_f T'$, so

$$R \approx \frac{N \cdot T}{\sum_{i=0}^{k} (l_{i+1} - l_i)\, T_i} = \frac{N}{\sum_{i=0}^{k} (l_{i+1} - l_i)\, (T_i/T)}\,. \tag{35}$$

□

Numerical verification. We verify the formula on Llama-3.1-8B-Instruct ($N = 32$, $d_{\text{model}} = 4096$, $d_{\text{ff}} = 14336$, $n_{\text{ffn}} = 3$ for SwiGLU, grouped-query attention with $n_{\text{kv}} = 8$). CTLs are at layers $l_1 = 8$ and $l_2 = 13$. Using typical pruning ratios ($T_1/T \approx 0.60$, $T_2/T \approx 0.36$):

- **Attention-dominated** ($T \to \infty$): $R \approx 32/(8 \times 1.0 + 5 \times 0.36 + 19 \times 0.13) = 32/12.26 \approx 2.61$.
- **FFN-dominated** (T moderate): $R \approx 32/(8 \times 1.0 + 5 \times 0.60 + 19 \times 0.36) = 32/17.84 \approx 1.79$.
- **Full formula at** $T = 128$**K**: With $c_a = 8{,}192$ and $c_f \approx 2.18 \times 10^8$ (from per-layer parameter counts), the ratio $c_a T/c_f \approx 4.8$, indicating a mixed regime. Substituting into Eq. (33) yields a theoretical FLOP speedup of $R \approx 2.42$; the experimentally observed $2.02\times$ wall-clock speedup is lower due to memory-bandwidth effects.

B Additional Experiments

B.1 Experimental Setup Details

Benchmarks. RULER [40] is a comprehensive long-context evaluation suite that extends beyond simple needle-in-a-haystack tests, comprising diverse task categories including multi-needle insertion, multi-word tracking, variable tracking, and aggregation, with context lengths ranging from 4K to 128K tokens. InfiniteBench [41] is the first benchmark with average data length surpassing 100K tokens, comprising synthetic and realistic tasks across diverse domains in English and Chinese that require understanding long-range dependencies rather than simple passage retrieval. LongBench [42] is a multi-task benchmark covering six task categories—single-document QA, multi-document QA, summarization, few-shot learning, synthetic tasks, and code completion—with average context lengths from 1.2K to 22K tokens.

Baseline configurations. All baselines use the default configurations from their original papers. For token pruning methods: SlimInfer prunes 50% of tokens after layers 10, 20, and 30 for Llama, and after layers 9, 18, and 26 for Qwen; FastKV prunes 50% after layer 15 for Llama and after layer 13 for Qwen. For sparse attention methods: FlexPrefill uses γ=0.95 and τ=0.1, and XAttention uses S=8 and τ=0.9. For KV cache compression methods: SnapKV and ChunkKV use an attention observation window of 64 and a pooling size of 5, PyramidKV uses β=20 and α=8 for pyramidal layer-wise KV budget allocation, and ChunkKV uses a chunk size of 10.

B.2 Task-Aware Retention Analysis

Figure 7 visualizes average retention rates after CTLs across context lengths. Tasks with concentrated attention—such as single-key retrieval (sk_2) and multi-value tracking (mv_niah)—exhibit low retention rates, indicating aggressive pruning; tasks with distributed attention—such as multi-key retrieval (mk_2, mk_3)—show higher average retention, largely because the entropy gate skips pruning for many of their inputs. As context grows, concentrated tasks prune even more aggressively—retention drops further from 32K to 128K—while distributed-attention tasks show more varied retention, reflecting the entropy gate's input-adaptive behavior. This validates the entropy gate design in §3.4: pruning activates when attention concentrates and deactivates when attention distributes broadly, avoiding the information loss of a fixed-rate strategy.

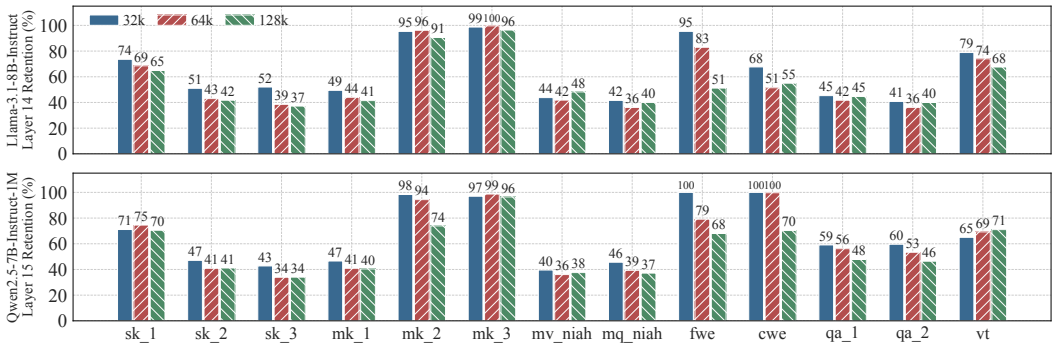


Figure 7: Task-aware retention rates after CTLs on RULER across 32K, 64K, and 128K contexts.

B.3 LongBench Detailed Results

Tables 7 and 8 report per-task results on LongBench [42] across all 21 tasks. CutAttn consistently accelerates TTFT while preserving Full Attention accuracy across all six task categories—from focused retrieval (e.g., PassageRetrieval) to diffuse-attention tasks (e.g., summarization, multi-document QA)—confirming that the entropy gate correctly adapts pruning to task structure.

Table 7: LongBench per-task results on Llama-3.1-8B-Instruct.

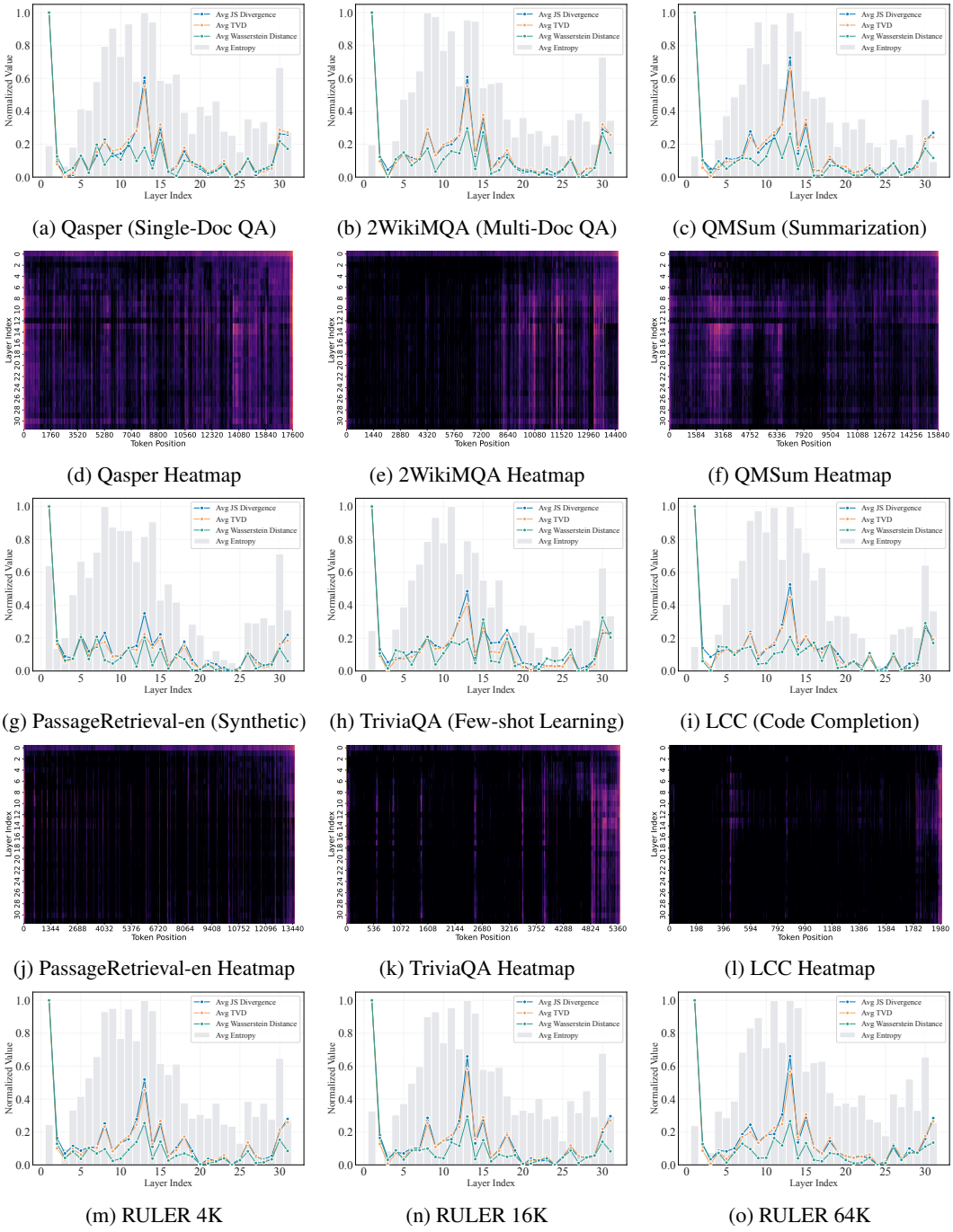
Task	Full Attn.	CutAttn	SlimInfer	FastKV	MInference	FlexPrefill	XAttention
Avg.	45.41	45.43	44.59	45.31	45.40	38.67	43.44
TTFT	0.956	1.28×	1.69×	1.34×	0.43×	0.88×	0.97×
Single-Document QA							
NarrativeQA	30.59	30.53	30.10	31.11	30.73	30.01	30.48
Qasper	47.50	47.51	46.28	47.22	46.70	45.33	45.85
MultiFieldQA-en	55.64	55.96	55.32	56.21	55.98	55.80	56.65
MultiFieldQA-zh	63.55	63.99	63.21	63.55	63.75	62.91	63.60
Multi-Document QA							
HotpotQA	59.51	59.19	59.01	58.86	57.70	58.12	58.42
2WikiMultihopQA	51.23	51.66	48.51	51.13	52.56	44.55	48.34
MuSiQue	33.67	33.76	30.96	32.71	33.36	35.44	33.19
DuReader	32.94	33.64	33.74	33.42	32.88	33.20	34.19
Summarization							
GovReport	35.13	35.15	33.80	34.47	35.04	34.71	35.23
QMSum	25.18	25.26	25.37	25.03	25.05	25.13	25.06
MultiNews	27.04	27.00	26.50	26.52	27.04	27.02	27.31
VCSUM	17.81	17.38	16.65	17.42	17.70	17.52	17.50
Few-shot Learning							
TREC	30.00	30.00	29.50	30.50	30.50	25.50	28.50
TriviaQA	92.21	92.39	92.63	91.58	92.71	90.83	92.25
SAMSum	40.94	40.79	41.00	40.48	39.82	39.82	42.08
LSHT	0.00	0.00	0.00	0.00	0.00	9.25	0.00
Synthetic							
PassageCount	11.65	12.15	9.08	11.20	12.15	2.00	10.50
PassageRetrieval-en	99.50	97.50	99.50	99.50	99.50	40.00	78.00
PassageRetrieval-zh	99.00	99.00	94.50	99.00	99.00	39.00	84.50
Code Completion							
LCC	53.31	53.61	53.62	54.28	53.85	50.00	53.10
RepoBench-P	47.31	47.62	47.13	47.27	47.46	45.92	47.43

Table 8: LongBench per-task results on Qwen2.5-7B-Instruct-1M.

Task	Full Attn.	CutAttn	SlimInfer	FastKV	MInference	FlexPrefill	XAttention
Avg.	48.98	48.85	45.76	48.36	48.66	38.88	46.92
TTFT	0.814	1.22×	1.62×	1.32×	0.36×	0.87×	0.96×
Single-Document QA							
NarrativeQA	29.44	29.23	28.44	28.92	28.52	19.31	25.48
Qasper	49.44	49.51	44.46	47.39	48.24	41.24	46.69
MultiFieldQA-en	51.62	51.95	51.35	50.92	51.94	47.36	52.31
MultiFieldQA-zh	53.99	55.04	54.92	52.28	53.78	50.07	53.29
Multi-Document QA							
HotpotQA	62.69	62.64	63.00	63.33	61.27	53.13	58.98
2WikiMultihopQA	56.62	55.72	52.75	53.79	56.64	44.82	54.70
MuSiQue	34.26	32.96	29.38	32.96	33.79	30.47	30.86
DuReader	32.21	32.44	31.61	31.14	31.28	32.96	32.97
Summarization							
GovReport	35.37	35.68	34.87	35.30	35.28	33.93	35.53
QMSum	24.81	24.89	24.80	23.88	25.08	24.23	24.46
MultiNews	26.45	25.82	24.92	25.34	25.96	26.14	26.02
VCSUM	16.65	16.02	15.83	15.97	16.58	16.57	16.30
Few-shot Learning							
TREC	68.00	68.00	64.50	66.50	67.50	64.50	68.00
TriviaQA	86.44	86.00	86.47	86.44	86.96	77.64	86.08
SAMSum	37.31	37.01	36.90	37.07	36.82	37.09	37.11
LSHT	34.25	36.00	35.75	35.00	34.75	28.00	36.50
Synthetic							
PassageCount	8.50	7.50	11.50	8.50	9.00	6.00	6.50
PassageRetrieval-en	99.50	98.00	97.00	99.50	98.00	29.00	82.50
PassageRetrieval-zh	98.50	98.50	52.00	99.00	99.00	45.00	90.00
Code Completion							
LCC	63.05	63.16	61.38	62.74	62.70	55.31	62.94
RepoBench-P	59.58	59.70	59.20	59.49	58.83	53.68	58.17

B.4 Cross-Task and Cross-Length Consistency of Cognitive Transition Layers

To verify that CTLs are an architecture-determined, input-invariant property, we conduct two complementary experiments. First, we evaluate both models on six tasks from LongBench [42], one from each of its six task categories: single-document QA (Qasper), multi-document QA (2WikiMultihopQA), summarization (QMSum), synthetic task (PassageRetrieval-en), few-shot learning (TriviaQA), and code completion (LCC). Second, we evaluate on RULER [40] at three context lengths (4K, 16K, 64K) to test whether CTL locations shift with input length. Figures 8 and 9 show that despite vastly different input distributions and context lengths, the JSD peaks—and thus the CTL locations—coincide across all conditions for both models, confirming that CTL emergence is independent of the input and depends solely on the model architecture.



(a) Qasper (Single-Doc QA) (b) 2WikiMQA (Multi-Doc QA) (c) QMSum (Summarization)

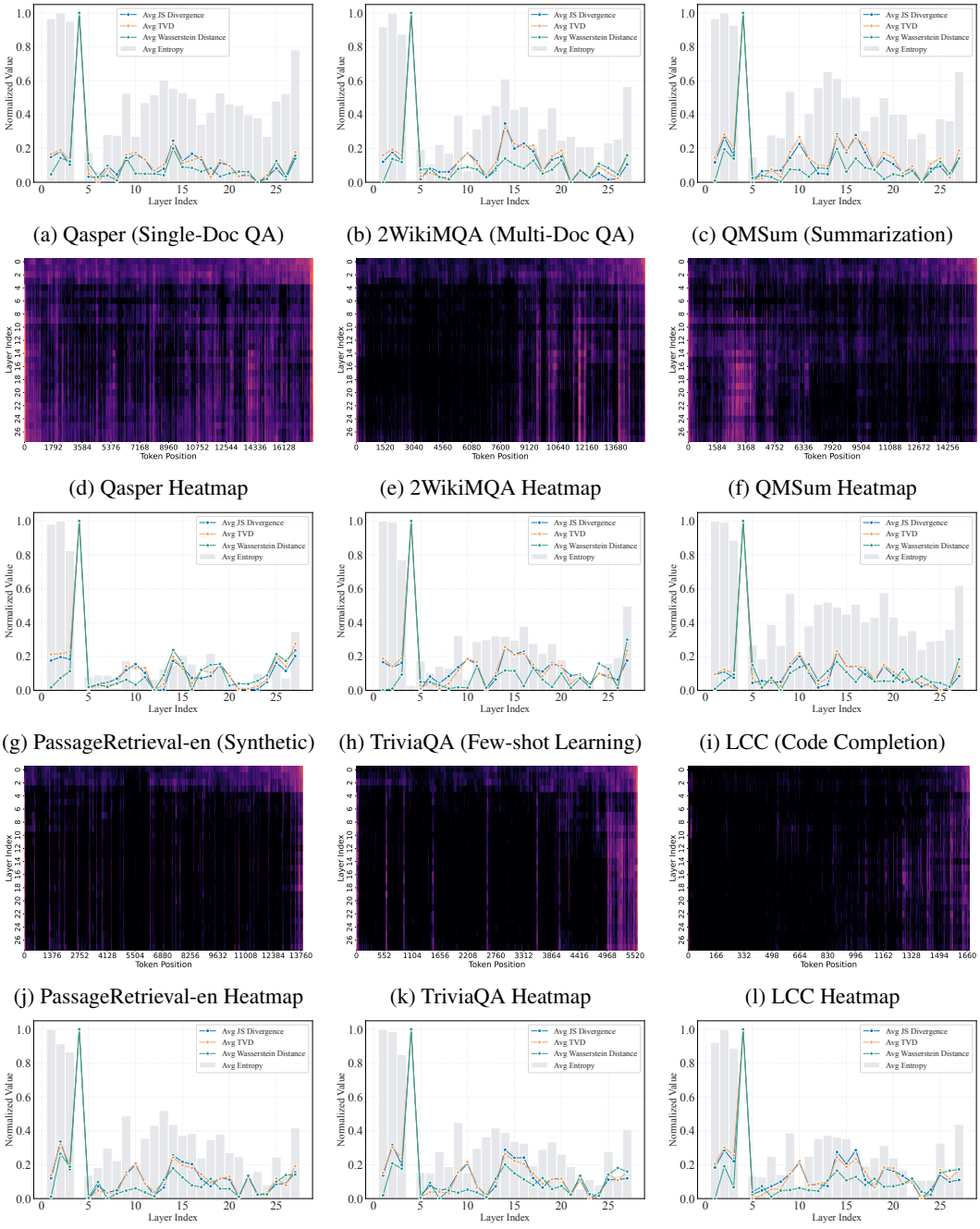
(d) Qasper Heatmap (e) 2WikiMQA Heatmap (f) QMSum Heatmap

(g) PassageRetrieval-en (Synthetic) (h) TriviaQA (Few-shot Learning) (i) LCC (Code Completion)

(j) PassageRetrieval-en Heatmap (k) TriviaQA Heatmap (l) LCC Heatmap

(m) RULER 4K (n) RULER 16K (o) RULER 64K

Figure 8: JSD curves (odd rows) and attention heatmaps (even rows) for Llama-3.1-8B-Instruct across six LongBench [42] tasks (top four rows) and three RULER [40] context lengths (bottom row). Despite diverse input distributions and context lengths, JSD peaks—and thus CTL locations—remain consistent.



(a) Qasper (Single-Doc QA) (b) 2WikiMQA (Multi-Doc QA) (c) QMSum (Summarization)

(d) Qasper Heatmap (e) 2WikiMQA Heatmap (f) QMSum Heatmap

(g) PassageRetrieval-en (Synthetic) (h) TriviaQA (Few-shot Learning) (i) LCC (Code Completion)

(j) PassageRetrieval-en Heatmap (k) TriviaQA Heatmap (l) LCC Heatmap

(m) RULER 4K (n) RULER 16K (o) RULER 64K

Figure 9: JSD curves (odd rows) and attention heatmaps (even rows) for Qwen2.5-7B-Instruct-1M across six LongBench [42] tasks (top four rows) and three RULER [40] context lengths (bottom row). The same CTL consistency pattern as in Figure 8 is observed, further confirming that CTL emergence is input-invariant and architecture-determined.

B.5 JSD Peak Ranking

Table 9 lists the top-8 JSD peaks ranked by prominence for both models on RULER at 16K. The selected CTLs (layers 8 and 13 for Llama; layers 10 and 14 for Qwen) rank among the most prominent peaks, but not all high-prominence peaks are chosen: early-layer peaks (e.g., Qwen layer 4, rank 1; Llama layer 4, rank 8) are excluded by the minimum-depth constraint $l_{\min}$=5 (§3.2), since the model is still gathering information at shallow layers and pruning there would remove context before it has been consolidated.

Table 9: Top-8 JSD peaks ranked by prominence on RULER 16K. d_l denotes the average JSD value at each peak layer. Selected CTLs are marked with ✓.

	Llama-3.1-8B				*Qwen2.5-7B-1M*			
Rank	Layer	Prominence	d_l	CTL	Layer	Prominence	d_l	CTL
1	13	0.2705	0.3619	✓	4	0.4335	0.6225	
2	8	0.0757	0.1988	✓	14	0.1386	0.2650	✓
3	18	0.0656	0.1571		10	0.0927	0.2227	✓
4	15	0.0613	0.1927		2	0.0580	0.2782	
5	26	0.0265	0.1209		22	0.0470	0.1801	
6	21	0.0166	0.0909		19	0.0274	0.1805	
7	23	0.0028	0.0868		6	0.0130	0.1581	
8	4	0.0026	0.1074		16	0.0008	0.2415	

B.6 Chunk Size Ablation

Figure 10 shows the effect of chunk size B on LongBench accuracy and TTFT for Llama-3.1-8B-Instruct. Smaller chunks (B=16, 32) prune more tokens at the same attention recall threshold, yielding lower TTFT but causing accuracy degradation due to fine-grained chunk-level pruning that removes informative tokens within otherwise important regions. Larger chunks (B=128) fail to capture the distribution of different key information within a chunk, also hurting accuracy. B=64 strikes a balance, preserving Full Attention accuracy while achieving substantial TTFT reduction.

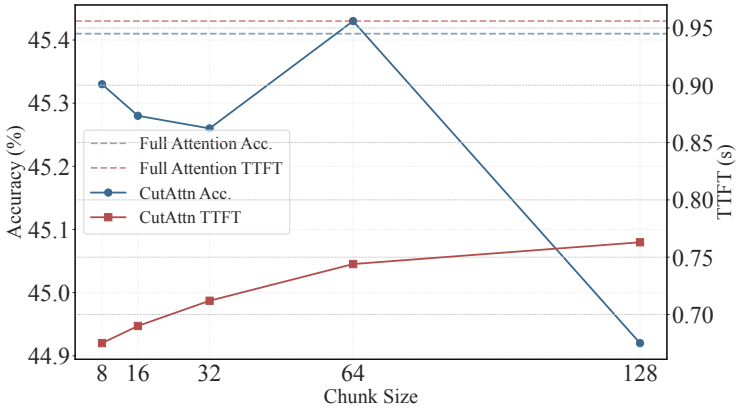


Figure 10: Effect of chunk size B on LongBench accuracy and TTFT for Llama-3.1-8B-Instruct. B=64 achieves the best accuracy–speed trade-off.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction state four contributions (CTL discovery, CutAttn framework, theoretical guarantees, and experimental validation), all of which are substantiated in Sections 3–4 and Appendix A.

   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: Section 5 discusses two limitations: (1) evaluation is limited to 7B–8B models, and the additive pruning error at each CTL may become more pronounced in deeper models requiring more pruning interventions; (2) dynamic sequence lengths introduced by CutAttn may cause load-balancing challenges in distributed inference with Sequence Parallelism.

   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: All theorems, corollaries, and propositions are stated with explicit assumptions in Section 3.5. Complete proofs are provided in Appendix A.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All 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: Section 4.1 specifies model names, benchmark versions, all hyperparameters (τ, α, chunk size B, observation window w, CTL layers), baseline configurations, evaluation frameworks, and hardware setup (NVIDIA A100-80GB). Crucially, we strictly fixed the random seed across all experiments to align the input data and ensure full reproducibility.

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 will be released upon acceptance. All benchmarks (RULER, InfiniteBench) are publicly available.

   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: CutAttn is training-free. All inference hyperparameters, baseline configurations, and evaluation details are reported in Section 4.1. Ablation studies in Section 4.4 justify the default hyperparameter choices.

   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: Following standard practice for computationally expensive long-context LLM evaluation, both accuracy and latency (TTFT) results are reported based on a single run. To ensure fair and reproducible comparisons, we strictly fixed the random seed across all

methods to guarantee identical input data sampling and evaluation conditions. Furthermore, we use greedy decoding, making the generation process deterministic. Due to the massive compute requirements of evaluating up to 128K/1M contexts, multiple runs for variance reporting were not feasible.

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: Section 4.1 reports the GPU type (NVIDIA A100-80GB), precision (BFloat16), and TTFT measurement methodology. Full Attention TTFT baselines in Tables 2 and 3 provide absolute timing references.

   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 work proposes an inference acceleration method for existing open-source models. It does not involve human subjects, private data, or dual-use concerns.

   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: CutAttn is a general-purpose inference efficiency method that reduces computational cost and energy consumption of LLM inference. It does not introduce new model capabilities or alter model behavior beyond acceleration. We do not foresee direct negative societal impacts.

    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: This work does not release new models or datasets. CutAttn is an inference optimization applied to existing publicly available models.

    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: All models (Llama-3.1-8B-Instruct, Qwen2.5-7B-Instruct-1M), benchmarks (RULER, InfiniteBench), and software frameworks (FlashAttention-2, kvpress, MInference) are cited with their original publications.

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: No new datasets or pre-trained models are released. Code will be released upon acceptance with documentation.

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: This work does not involve crowdsourcing or human subjects research.

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: This work does not involve human subjects research.

    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: LLMs are used as evaluation subjects in this work, not as a component of the proposed method. The models evaluated (Llama-3.1, Qwen2.5) are publicly available and cited.

    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.