Latent-Condensed Transformer for Efficient Long Context Modeling

Anonymous ACL submission

Abstract

Large language models (LLMs) face significant challenges in processing long contexts due to the linear growth of the key-value (KV) cache and quadratic complexity of self-attention. Existing approaches address these bottlenecks separately: Multi-head Latent Attention (MLA) reduces the KV cache by projecting tokens into a low-dimensional latent space, while sparse attention reduces computation. However, sparse methods cannot operate natively on MLA's compressed latent structure, missing opportunities for joint optimization. In this paper, we propose Latent-Condensed Attention (LCA), which directly condenses context within MLA's latent space, where the representation is disentangled into semantic latent vectors and positional keys. LCA separately aggregates semantic vectors via query-aware pooling and preserves positional keys via anchor selection. This approach jointly reduces both computational cost and KV cache without adding parameters. Theoretically, we prove a length-independent error bound. Experiments show LCA achieves up to **2.5**$\times$ prefilling speedup and **90%** KV cache reduction at 128K context while maintaining competitive performance.

1 Introduction

Efficient long-context modeling in large language models (LLMs) is essential for applications spanning full-document comprehension and extended multi-turn dialogues (OpenAI, 2023; Grattafiori et al., 2024; Guo et al., 2025). However, transformer-based LLMs face two challenges: 1) the linear growth of key-value (KV) cache during decoding and 2) the quadratic computational complexity of self-attention (Vaswani et al., 2017), together hindering long-context deployment.

To alleviate the memory overhead (challenge 1), Multi-head Latent Attention (MLA) (DeepSeek-AI et al., 2024) projects tokens into a shared low-dimensional latent space, significantly reducing the per-token KV cache size. During inference, MLA caches only the compressed latent vectors $\mathbf{C}^{KV}$ and the positional keys $\mathbf{K}^R$, where $\mathbf{K}^R$ encodes precise positional information via Rotary Position Embedding (RoPE) (Su et al., 2024). Owing to its favorable trade-off between memory efficiency and model capacity, MLA has been widely adopted in recent long-context models (DeepSeek-AI et al., 2024; Liu et al., 2024; Guo et al., 2025; Kimi et al., 2025a,b). Nevertheless, MLA still retains all L latent vectors and performs dense attention, preserving the quadratic computational bottleneck.

Separately, efficient attention methods mitigate quadratic computation (challenge 2) via sparsification. Early approaches used fixed sparse patterns (Beltagy et al., 2020; Zaheer et al., 2020), which lack adaptability and risk discarding important information. Recent dynamic sparse methods (Jiang et al., 2024; Lai et al., 2025; Xu et al., 2025) selectively skip computation blocks or evict less relevant tokens to reduce computation. However, these approaches remain susceptible to potentially important information loss.

Crucially, these two lines of work cannot be directly combined to address both challenges simultaneously. Most existing sparse methods operate on the computation of attention scores between original queries and keys, which require full-dimensional representations. When applied to MLA, these methods must first reconstruct full-dimensional KV matrices before performing sparsification. Consequently, these methods fail to leverage the compressed latent structure for further efficiency gains, particularly in reducing the KV cache beyond what MLA already achieves. This reveals a significant yet overlooked gap: existing methods lack a mechanism to perform efficient computation reduction natively within MLA's latent space.

In this paper, we bridge this gap by directly condensing redundant context within the MLA's latent space into a compact set of representatives,

thereby reducing the number of vectors participating in attention computation and stored in the KV cache. However, MLA's disentangled representation presents a unique challenge: the *semantic* component $\mathbf{C}^{KV}$ and the *positional* component $\mathbf{K}^{R}$ have distinct functional properties due to their different encodings. Semantic information, which captures the continuous and often smooth content representations, can be aggregated across tokens without significant loss of fidelity (Bolya et al., 2023; Feng and Zhang, 2023); while positional encoding requires careful preservation to maintain accurate relative positional relationships. Thus, an effective condensation strategy must treat these two components differently.

Based on the above insights, we propose **Latent-Condensed Attention** (**LCA**), an efficient attention mechanism that performs structured context condensation natively within latent space. LCA partitions context into groups and compresses each group into a representative latent vector via weighted pooling for the semantic component $\mathbf{C}^{KV}$, while preserving positional information by selecting the token with the highest relevance within each group for $\mathbf{K}^{R}$. In this way, LCA reduces both the KV cache size and the attention complexity, without introducing additional parameters. We further provide a theoretical guarantee that the approximation error is uniformly bounded, independent of context length. We summarize our contributions as follows:

- We propose Latent-Condensed Attention (LCA), an efficient attention mechanism that performs structured context condensation directly in MLA's latent space, reducing both KV cache and attention computations, instead of sparsification only on reconstructed KV.
- We design a latent-space condensation strategy that decouples semantic and positional processing: semantic information is adaptively aggregated via query-aware weighted pooling, while positional fidelity is preserved through hard anchor selection. This design avoids the signal-blending issue in conventional approaches.
- LCA can be easily integrated into pretrained models via lightweight fine-tuning, requiring no additional parameters. Experiments demonstrate that LCA delivers up to **2.5**× prefill speedup and **90%** KV cache reduction at 128K context length while maintaining comparable performance.

2 Related Works

Efficient Attention. To reduce the memory footprint of the KV cache, several attention variants have been proposed. Multi-Query Attention (MQA) (Shazeer, 2019) shares a single key and value head across all query heads, while Grouped-Query Attention (GQA) (Ainslie et al., 2023) groups query heads to share key-value heads, improving efficiency while maintaining quality. More recently, Multi-head Latent Attention (MLA) (DeepSeek-AI et al., 2024) compresses KV states into a low-dimensional latent space, significantly reducing per-token cache size. It has been widely adopted in state-of-the-art long-context models (DeepSeek-AI et al., 2024; Liu et al., 2024; Guo et al., 2025; Kimi et al., 2025a,b). However, these methods retain the quadratic computational complexity of standard attention.

Besides, sparse attention methods (Zaheer et al., 2020; Beltagy et al., 2020; Zhang et al., 2025; Chen et al., 2025) reduce the computational cost by skipping some tokens or computation blocks. Early methods employ fixed sparse patterns (Zaheer et al., 2020; Ding et al., 2023; Beltagy et al., 2020; Xiao et al., 2024), which lack adaptability. To introduce adaptability, DuoAttention (Xiao et al., 2025) learns head-specific static sparse patterns during an additional offline training phase. However, static patterns cannot adjust to the varying importance of tokens across different contexts. Recent dynamic sparse attention methods (Lai et al., 2025; Jiang et al., 2024; Xiao et al., 2025) selectively compute only a subset of attention blocks. However, these approaches operate on full-dimensional key-value representations. When applied to MLA, they cannot further reduce the latent cache itself.

System-Level Optimizations for Long-Context Inference. Another line of work focuses on kernel-level optimizations and cache management for long-context inference. FlashAttention (Dao et al., 2022) and its successors speed up attention by optimizing memory access patterns. PagedAttention (Kwon et al., 2023) manages the KV cache more efficiently by allowing non-contiguous storage, reducing memory fragmentation. Token-level eviction policies (Li et al., 2024; Hao et al., 2025; Liu et al., 2025) retain only the most salient tokens based on attention scores, while query-aware cache selection methods (Tang et al., 2024) learn to select a compact set of KV pairs for each query. These techniques are largely orthogonal to our approach.

3 Preliminaries

Multi-head Latent Attention (MLA) (DeepSeek-AI et al., 2024) is a core breakthrough for long-context large language models. It projects tokens into a low-dimensional latent space, drastically reducing per-token KV-cache memory while preserving modeling capability. Given an input sequence $\mathbf{X} \in \mathbb{R}^{L\times d}$, MLA computes compressed latents:

$$\mathbf{C}^{Q}=\mathbf{X}W^{DQ}, \quad \mathbf{C}^{KV}=\mathbf{X}W^{DKV}, \tag{1}$$

where $W^{DQ}, W^{DKV}\in\mathbb{R}^{d\times d_c}$ are down-projection matrices and $d_c \ll d$. For each head h, queries, keys, and values are reconstructed from these latents:

$$\mathbf{Q}^{h}=[\mathbf{C}^{Q}W^{UQ_h}, \mathbf{Q}^{R_h}] \in \mathbb{R}^{L\times d_k}, \tag{2}$$

$$\mathbf{K}^{h}=[\mathbf{C}^{KV}W^{UK_h}, \mathbf{K}^{R}] \in \mathbb{R}^{L\times d_k}, \tag{3}$$

$$\mathbf{V}^{h}=\mathbf{C}^{KV}W^{UV_h} \in \mathbb{R}^{L\times d_v}, \tag{4}$$

where $\mathbf{Q}^{R_h}, \mathbf{K}^{R} \in \mathbb{R}^{L\times d_r}$ are Rotary Position Embedding (RoPE) (Su et al., 2024) augmented components that encode positional information. $W^{UQ_h}, W^{UK_h} \in \mathbb{R}^{d_c\times d'_k}$ and $W^{UV_h} \in \mathbb{R}^{d_c\times d_v}$ are the up-projection matrices for head h, with $d'_k + d_r = d_k$. The attention is computed as:

$$\text{MLA}^{h} = \text{softmax}\left(\frac{\mathbf{Q}^{h}(\mathbf{K}^{h})^{\top}}{\sqrt{d_k}}\right)\mathbf{V}^{h}. \tag{5}$$

The outputs of all heads are then combined via concatenation. During inference, MLA caches only $\mathbf{C}^{KV}$ and $\mathbf{K}^{R}$, reducing per-token memory. Owing to its favorable efficiency-accuracy trade-off, MLA has been widely adopted as a core component in state-of-the-art long-context LLMs (Liu et al., 2024; DeepSeek-AI, 2025; Kimi et al., 2025a,b).

Limitations of MLA. While MLA substantially reduces per-token KV-cache memory by projecting tokens into a low-dimensional latent space, it still requires all L tokens to participate in attention computation, resulting in significant redundant computation and memory access and leaving the quadratic dependence on context length unchanged. Existing efficient attention methods (Xiao et al., 2024, 2025; Lai et al., 2025) can alleviate computational cost by sparsifying attention, but they operate on reconstructed full-dimensional key–value states. When applying these methods to MLA, this reconstruction step negates the efficiency benefits of its low-dimensional latent representation. As a result, current approaches **fail to exploit MLA's latent structure to jointly reduce attention computation and KV-cache growth**.

4 Latent-Condensed Attention

4.1 Motivations and Method Overview

Long context sequences often exhibit substantial redundancy: only a small subset of tokens is typically relevant for the task (Chen et al., 2025; Zhang et al., 2025; Xiao et al., 2024). This suggests that computing attention over all L tokens incurs unnecessary computational overhead. While Multi-head Latent Attention (MLA) effectively compresses the KV cache into a low-dimensional latent space, it still performs quadratic attention over all L tokens. We propose to address this bottleneck by condensing redundant context before attention computation. Crucially, unlike methods that sparsify attention on reconstructed full-dimensional states, *we operate natively within compressed latent space, leveraging its latent structure for further efficiency gains*.

Unfortunately, MLA's latent representation is naturally *disentangled*: each token is encoded by a compressed semantic latent $\mathbf{C}^{KV}$ and a residual positional key $\mathbf{K}^{R}$ with precise positional information via RoPE (Su et al., 2024). The semantic and positional components differ functionally due to their distinct encodings: semantic information can often be aggregated (Feng and Zhang, 2023; Bolya et al., 2023), whereas positional encoding is highly nonlinear and should not be arbitrarily blended. Hence, an effective condensation strategy must treat these two components differently.

Motivated by these insights, we propose **Latent-Condensed Attention (LCA)**, which employs a dual-path group-wise condensation strategy: semantic vectors are aggregated via query-aware pooling, while positional fidelity is preserved by selecting a positional anchor per group. By operating natively in MLA's latent space, LCA reduces both the number of cached vectors and the attention complexity, without introducing additional parameters. The framework of LCA is illustrated in Figure 1, with its algorithm in Algorithm 1.

4.2 Latent-Space Condensation

We perform latent-space token condensation by explicitly distinguishing semantic and positional information, and applying different reduction operators to each. This enables effective reduction of long-range context while preserving sufficient information required for attention computation.

Given the latent matrix $\mathbf{C}^{KV} \in \mathbb{R}^{L\times d_c}$ and the positional keys $\mathbf{K}^{R} \in \mathbb{R}^{L\times d_r}$, we partition the history context into $m = \left\lfloor \frac{L-w}{g} \right\rfloor$ contiguous groups

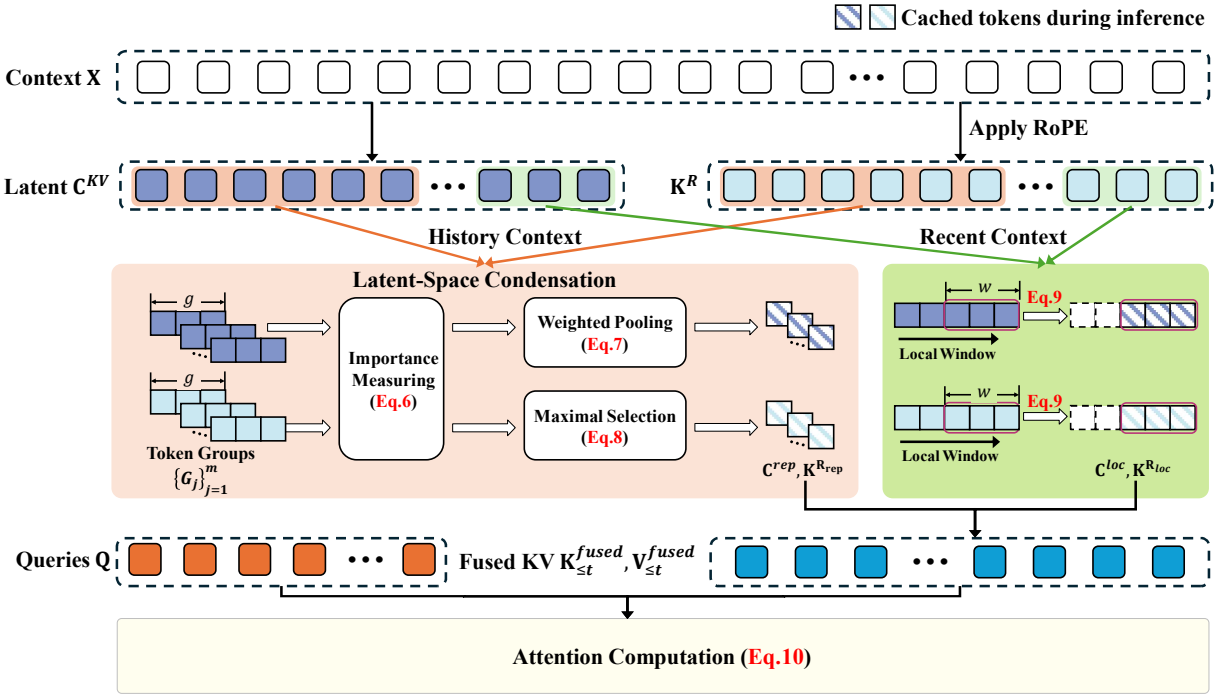

Figure 1: An overview of Latent-Condensed Attention (LCA). The history context is condensed into a compact set of representatives via group-wise condensation in the latent space: semantic information is aggregated through weighted pooling of $\mathbf{C}^{KV}$, while positional information is preserved via anchor selection from $\mathbf{K}^R$. The recent context is retained in full fidelity without condensation.

$\{G_j\}_{j=1}^m$ of size g, where w denotes a local window size that preserves fine-grained local context (detailed later). To assess token relevance within each group, we adopt a query-aware scoring mechanism that reflects the current decoding focus rather than relying on static heuristics. Following Lai et al. (2025), we compute a summary query by averaging the last g queries, *i.e.*, $\bar{\mathbf{q}} = \text{Average}(\mathbf{Q}_{[-g:]})$. For each token $i \in G_j$, we compute an importance score $s_i = \bar{\mathbf{q}}^\top \mathbf{k}_i / \sqrt{d_h}$, followed by a group-wise softmax normalization:

$$\alpha_i^{(j)} = \frac{\exp(s_i)}{\sum_{k \in G_j} \exp(s_k)}, \quad \forall i \in G_j. \tag{6}$$

We treat semantic and positional components differently based on their distinct functional properties.

Semantic condensation via weighted pooling. For the semantic latent vectors $\mathbf{c}_i^{KV}$ within each group, we compute a representative latent via weighted pooling:

$$\mathbf{c}_j^{\text{rep}} = \sum_{i \in G_j} \alpha_i^{(j)} \mathbf{c}_i^{KV} \in \mathbb{R}^{d_c}, \tag{7}$$

where $\mathbf{c}_i^{KV}$ denotes the i-th row of $\mathbf{C}^{KV}$. Unlike sparse attention methods that discard tokens or computation blocks, weighted pooling preserves information from all tokens in the group while emphasizing those most relevant to the current query. This design admits a simple yet principled interpretation. The following proposition shows that weighted pooling yields the optimal representative under an expected reconstruction criterion.

Proposition 1. ***(Optimal Condensation via Weighted Pooling)*** *Let $\{\mathbf{c}_i \in \mathbb{R}^{d_c}\}_{i=1}^g$ be a set of latent vectors and $\boldsymbol{\alpha} \in \Delta^{g-1}$ a probability distribution over them, with $\alpha_i \ge 0$ and $\sum_i \alpha_i = 1$. Consider the expected squared reconstruction error*

$$\mathcal{L}(\mathbf{c}^{rep}) = \mathbb{E}_{i \sim \boldsymbol{\alpha}} \left[\|\mathbf{c}_i - \mathbf{c}^{rep}\|_2^2 \right] = \sum_{i=1}^{g} \alpha_i \|\mathbf{c}_i - \mathbf{c}^{rep}\|_2^2.$$

The vector $\mathbf{c}^{rep}$ that minimizes $\mathcal{L}$ is uniquely given by the convex combination $\mathbf{c}^{rep} = \sum_{i=1}^g \alpha_i \mathbf{c}_i$.

Applying Proposition 1 with $\alpha_i = \alpha_i^{(j)}$ shows that our weighted pooling minimizes the expected error in representing each group's semantic content. The full proof is provided in Appendix A.1.

Positional anchoring via max-selection. The residual positional keys $\mathbf{k}_i^R$ are encoded using Rotary Position Embedding (RoPE) (Su et al., 2024), which is a nonlinear function of absolute position. Direct pooling over $\mathbf{k}_i^R$ would blend distinct posi-

Algorithm 1 Latent-Condensed Attention

Require: Sequence length L, latents $\mathbf{C}^{KV}=[\mathbf{c}_1^{KV};\ldots;\mathbf{c}_L^{KV}]$, residual keys $\mathbf{K}^R=[\mathbf{k}_1^R;\ldots;\mathbf{k}_L^R]$, queries $\mathbf{Q}=[\mathbf{q}_1;\ldots;\mathbf{q}_L]$, window size w, group size g.

$\bar{\mathbf{q}}\leftarrow\text{Avg}([\mathbf{q}_{L-g+1};\cdots;\mathbf{q}_L]), m\leftarrow\lfloor(L-w)/g\rfloor$
for $j=1$ to m **do**
  $G_j=\{(j-1)g+1,(j-1)g+2,\ldots,jg\}$
  $\alpha_i^{(j)}\leftarrow\text{softmax}_i(\bar{\mathbf{q}}^\top\mathbf{k}_i/\sqrt{d_h})$ for $i\in G_j$
  $\mathbf{c}_j^{\text{rep}}\leftarrow\sum_{i\in G_j}\alpha_i^{(j)}\mathbf{c}_i^{KV}$
  $I_j\leftarrow\arg\max_{i\in G_j}\alpha_i^{(j)}$
  $\mathbf{k}_j^{\text{rep}}\leftarrow[\mathbf{c}_j^{\text{rep}}W^{UK},\mathbf{k}_{I_j}^R]$, $\mathbf{v}_j^{\text{rep}}\leftarrow\mathbf{c}_j^{\text{rep}}W^{UV}$
end for
$\mathbf{K}^{\text{rep}}\leftarrow[\mathbf{k}_1^{\text{rep}};\ldots;\mathbf{k}_m^{\text{rep}}]$, $\mathbf{V}^{\text{rep}}\leftarrow[\mathbf{v}_1^{\text{rep}};\ldots;\mathbf{v}_m^{\text{rep}}]$
$k\leftarrow\max(w,w+((L-w)\bmod g))$
$\mathbf{K}^{\text{loc}}=[\mathbf{k}_{L-k};\ldots;\mathbf{k}_L]$, $\mathbf{V}^{\text{loc}}=[\mathbf{v}_{L-k};\ldots;\mathbf{v}_L]$
$\mathbf{K}^{\text{fused}}\leftarrow[\mathbf{K}^{\text{rep}};\mathbf{K}^{\text{loc}}]$, $\mathbf{V}^{\text{fused}}\leftarrow[\mathbf{V}^{\text{rep}};\mathbf{V}^{\text{loc}}]$
$\mathbf{Attn}\leftarrow\text{softmax}\big(\mathbf{Q}(\mathbf{K}^{\text{fused}})^\top/\sqrt{d_h}\big)\mathbf{V}^{\text{fused}}$

Ensure: Attention output $\mathbf{Attn}_t$

tional signals and distort relative positional relationships. To preserve accurate positional information, we select the token with the highest importance score within each group as the positional anchor:

$$I_j=\arg\max_{i\in G_j}\alpha_i^{(j)},\quad \mathbf{k}_j^{R_{\text{rep}}}=\mathbf{k}_{I_j}^R. \tag{8}$$

This strategy ensures that each representative's positional information corresponds to a concrete token position, preserving valid attention geometry. In contrast to semantic aggregation, positional information is preserved through hard selection to avoid blending incompatible positional signals.

Preserving Fine-Grained Local Context. While latent condensation effectively reduces redundancy in long-range context, fine-grained local information remains critical for accurate next-token prediction (Manakul and Gales, 2021; Yang et al., 2021). Accordingly, we retain a local window of the most recent w latent vectors in full fidelity. Any residual tokens that do not form a complete group are also included in this window. Specifically, the local window consists of the preceding $k=w+r$ latent vectors, where $r=(L-w)\bmod g$, ensuring that every token is accounted for.

4.3 Attention with Condensed Context

Representative Key–Value Construction. For each group G_j, the final representative key and value matrices are constructed using MLA's original up-projection matrices:

$$\mathbf{k}_j^{\text{rep}}=\left[\mathbf{c}_j^{\text{rep}}W^{UK},\mathbf{k}_j^{R_{\text{rep}}}\right],\quad \mathbf{v}_j^{\text{rep}}=\mathbf{c}_j^{\text{rep}}W^{UV}.$$

Collecting all representatives, we obtain condensed key and value matrices:

$$\mathbf{K}^{\text{rep}}=[\mathbf{k}_1^{\text{rep}};\ldots;\mathbf{k}_m^{\text{rep}}],\ \mathbf{V}^{\text{rep}}=[\mathbf{v}_1^{\text{rep}};\ldots;\mathbf{v}_m^{\text{rep}}].$$

We preserve the corresponding keys and values of the local context without condensation:

$$\mathbf{K}^{loc}=[\mathbf{k}_{L-k+1};\ldots;\mathbf{k}_L],\mathbf{V}^{loc}=[\mathbf{v}_{L-k+1};\ldots;\mathbf{v}_L], \tag{9}$$

where each $\mathbf{k}_j=[\mathbf{c}_j^{KV}W^{UK},\mathbf{k}_j^R]$ and $\mathbf{v}_j=\mathbf{c}_j^{KV}W^{UV}$. This ensures that the most recent and relevant tokens are always attended to with full fidelity, seamlessly integrating condensed long-range context with detailed local information.

Prefilling with Fused Context. During prefilling, the condensed key and value matrices $\mathbf{K}^{\text{rep}}$ and $\mathbf{V}^{\text{rep}}$ are combined with the full-fidelity local keys and values $\mathbf{K}^l$ and $\mathbf{V}^l$ to form the fused context:

$$\mathbf{K}^{\text{fused}}=\left[\mathbf{K}^{\text{rep}};\mathbf{K}^{loc}\right],\mathbf{V}^{\text{fused}}=\left[\mathbf{V}^{\text{rep}};\mathbf{V}^{loc}\right].$$

We then compute attention over this fused set:

$$\mathbf{Attn}=\text{softmax}\left(\frac{\mathbf{Q}(\mathbf{K}^{\text{fused}})^\top}{\sqrt{d_h}}\right)\mathbf{V}^{\text{fused}}. \tag{10}$$

Notably, when the total sequence length L is smaller than $w+g$, the long-range condensation provides negligible efficiency benefits. In such cases, we revert to standard MLA computation, ensuring that the model always operates in the most efficient regime without sacrificing accuracy.

Decoding and Online Cache Update. During autoregressive decoding, we only cache only the m representative latent vectors $\{\mathbf{c}_j^{\text{rep}}\}_{j=1}^m$ and their positional anchors $\{\mathbf{k}_j^{R_{\text{rep}}}\}_{j=1}^m$, together with the w most recent latent vectors in full fidelity. Once the number of newly generated tokens reaches the group size g, we use the average query of the last g tokens to compute importance scores (as in Eq. (6)) for the earliest g tokens in the buffer. These g tokens are then condensed into a single representative via Eqs. (7) and (8). This new representative is appended to the cache. This on-the-fly way ensures that the cache size remains $\mathcal{O}(m+w)$ during decoding, and each decoding step attends to only $m+w$ keys and values, thereby reducing both memory footprint and per-token decoding latency.

4.4 Further Analysis of LCA

Theoretical Analysis. We provide a formal guarantee on the approximation quality of LCA. The following theorem bounds the error between the output of LCA and MLA attention.

Theorem 1 (**Uniform Error Bound**). *Fix a query* $\mathbf{q}_t$ *with* $\|\mathbf{q}_t\|_2 \le Q$ *and assume* $\|\mathbf{v}_i\|_2 \le V$ *for all values. For each distant-history group* G_j*, let* $\mathbf{k}_j^{\text{rep}}, \mathbf{v}_j^{\text{rep}}$ *be the representative key and value that LCA uses for every token* $i \in G_j$*, and let* δ_k, δ_v *be uniform bounds on the per-token key and value deviations:* $\|\mathbf{k}_i - \mathbf{k}_j^{\text{rep}}\|_2 \le \delta_k, \|\mathbf{v}_i - \mathbf{v}_j^{\text{rep}}\|_2 \le \delta_v$*. Then the* ℓ_2 *error between the full MLA attention and LCA satisfies*

$$\left\|\mathbf{Attn}_t - \mathbf{Attn}_t^{\text{LCA}}\right\|_2 \le V\left(e^{2Q\delta_k/\sqrt{d_h}} - 1\right) + \delta_v.$$

Theorem 1 shows that the approximation error depends only on the per-token deviations δ_k, δ_v, the query/value norms Q, V, and the model dimension d_h, but not on the context length L. The deviations δ_k, δ_v are controlled by our condensation design: groups consist of contiguous, semantically similar tokens, and the representative vectors are constructed via weighted pooling to preserve salient information. Moreover, the local window ensures the most recent tokens are never compressed ($\delta_k = \delta_v = 0$ for those tokens). The full proof is provided in Appendix A.2. Experimentally, with practical group sizes, this approximation maintains performance comparable to full MLA while delivering significant efficiency gains.

Complexity Analysis. Our method achieves substantial reductions in both computational and memory complexity compared to standard MLA. By compressing the long-range context into m representative key–value pairs and retaining only w recent tokens, LCA reduces the attention computation from $\mathcal{O}(L^2)$ to $\mathcal{O}(L(m+w)) = \mathcal{O}(Lm)$, where $m \ll L$ and w is a small constant. Correspondingly, the KV cache memory footprint is reduced from $\mathcal{O}(L)$ to $\mathcal{O}(m+w) = \mathcal{O}(m)$, enabling efficient handling of long contexts with substantially lower computational and memory costs. We provide a detailed derivation in Appendix B.

5 Experiments

5.1 Experimental Setup

Evaluation Benchmarks. We evaluate our method on both long-context and short-context tasks to comprehensively assess its performance. For long-context evaluation, we adopt **LongBench-E** (Bai et al., 2023) and **RULER** (Hsieh et al., 2024), which comprehensively test retrieval, reasoning, and aggregation over sequences up to 128K tokens. To ensure no degradation on standard tasks, we further evaluate on three established short-context benchmarks: **MMLU** (Hendrycks et al., 2021), **GSM8K** (Cobbe et al., 2021), and **MBPP** (Austin et al., 2021). We put more details in Appendix C.1.

Table 1: Comparison of efficient attention methods.

Methods	Computation	Memory	No Extra Params
Minference	$< \mathcal{O}(L^2)$	$\mathcal{O}(L)$	✓
FlexPrefill	$< \mathcal{O}(L^2)$	$\mathcal{O}(L)$	✓
KDA	$< \mathcal{O}(L^2)$	$\mathcal{O}(L)$	
Ours	$\mathcal{O}(Lm)$	$\mathcal{O}(m)$	✓

Implementation Details. We replace the MLA layers of DeepSeek-V2-Lite (16B) (DeepSeek-AI et al., 2024) with our LCA, adopting it as the first publicly released MLA-based model in our study. This choice is motivated by its moderate model size, which enables comprehensive experimentation under limited GPU resources. We develop a highly optimized kernel of our LCA using Triton (Tillet et al., 2019) for high efficiency. We fine-tune the model for 1,000 steps on the SlimPajama dataset (32K length) (Fu et al., 2024). The group size g and the window size w are set to 16 and 1024, respectively. All experiments are conducted on 8×H200 GPUs. Further implementation specifics are detailed in Appendix C.2.

Compared Methods. We evaluate LCA against three representative efficient attention methods: two dynamic sparse attention approaches (**FlexPrefill** (Lai et al., 2025) and **MInference** (Jiang et al., 2024)) and a gated linear attention variant (**KDA** (Kimi et al., 2025b)). Since FlexPrefill and MInference are designed for standard self-attention, we adapt them to operate on reconstructed KV matrices from MLA's latents. For KDA, which requires integration during training from scratch, we implement a controlled adaptation: we augment the pretrained model with additional modules required by KDA and fine-tune it under the same conditions as LCA. Details are provided in Appendix C.3. We summarize the differences from existing methods in Table 1.

5.2 Performance Comparisons

Comparisons on Long Context Modeling. In Table 2, our LCA achieves an average score of 29.09 on LongBench-E, nearly matching the 29.51 of the original MLA while achieving a **1.8**× speedup at

Table 2: Comparisons on LongBench-E (Bai et al., 2023). We report the latency of DeepSeek V2-Lite within contexts of 64K on H200 GPUs.

Methods	S. QA	M. QA	Sum.	F. S.	Syn.	Code	Avg.↑	FTL (s) ↓
DeepSeek V2-Lite *(MLA)*	**23.92**	11.15	17.52	**63.19**	3.09	58.17	**29.51**	3.20
• MInference (Jiang et al., 2024)	14.54	5.61	13.80	41.15	2.17	41.01	19.71	2.99 (1.1×)
• FlexPrefill (Lai et al., 2025)	14.35	5.94	14.10	48.00	2.22	41.72	21.05	2.51 (1.3×)
• KDA (Kimi et al., 2025b)	18.81	**11.68**	14.22	56.97	**3.25**	57.97	27.15	2.47 (1.3×)
• LCA (Ours)	22.61	8.90	**23.74**	58.27	2.33	**58.67**	29.09	**1.80 (1.8×)**

Table 3: Comparisons on RULER across 4-128K context. We report the latency of DeepSeek V2-Lite within contexts of 128K on H200 GPUs.

Methods	4K	8K	16K	32K	64K	128K	Avg.↑	FTL (s) ↓
DeepSeek V2-Lite (*MLA*)	**79.51**	**78.82**	62.28	58.33	47.57	23.96	**58.91**	10.78
• MInference (Jiang et al., 2024)	76.88	69.69	44.06	20.31	10.31	4.34	37.60	5.66 (1.9×)
• FlexPrefill (Lai et al., 2025)	72.81	70.00	56.25	19.06	9.38	7.19	39.11	5.38 (2.0×)
• KDA (Kimi et al., 2025b)	72.22	73.61	**64.24**	51.39	44.10	22.22	54.63	4.96 (2.2×)
• LCA (Ours)	77.19	77.19	63.75	**59.69**	**50.63**	**24.38**	58.80	**4.40 (2.5×)**

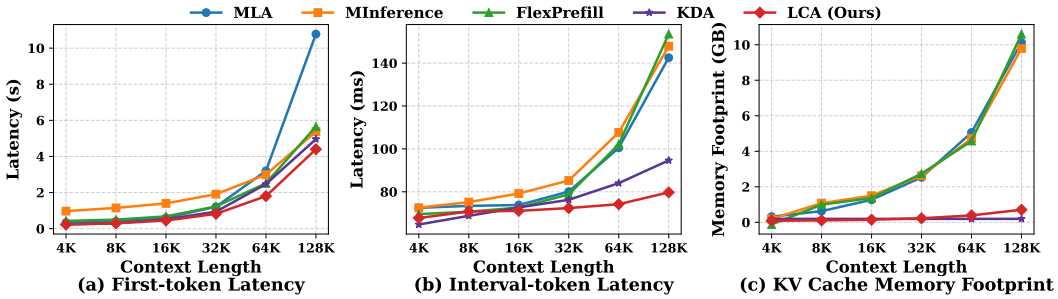

Figure 2: Efficiency comparison of different context lengths. (a) presents the first-token generation latency. (b) shows the per-token (interval) latency during decoding. (c) illustrates the GPU memory footprint of KV cache.

64K context. Notably, LCA outperforms MLA on the code generation (58.67 vs. 58.17), demonstrating its capability to preserve critical information. In contrast, existing sparse attention methods suffer significant performance degradation, with Minference and FlexPrefill dropping to 19.71 and 21.05, respectively, due to their sparsification strategy that may discard essential information.

In Table 3, LCA maintains strong performance across all lengths on RULER, achieving the highest scores at 32K, 64K, and 128K contexts. At 128K, LCA attains 24.38, surpassing MLA's 23.96 while reducing latency by **2.5**×. This demonstrates that our latent-space condensation effectively preserves long-range dependencies even at extreme lengths. Notably, while other methods show severe performance degradation at longer contexts (*e.g.*, Minference dropping to 4.34 at 128K), LCA maintains stable performance, highlighting the advantage of direct latent-space condensation over reconstruction-based sparsification.

Comparisons on Short-Context Tasks. To verify the gains do not compromise performance on standard tasks, we evaluate LCA on three widely-used short-context benchmarks. In Table 4, LCA achieves nearly identical performance to the original MLA across all three benchmarks. This demonstrates that the latent-space condensation introduced by LCA does not harm the model's capability on short-context tasks.

5.3 Comparisons on Computational Efficiency

In Figure 2, LCA achieves consistent and scalable efficiency gains across various context lengths. It reduces prefilling latency from short to long contexts, achieving **2.5**× speedup over MLA at 128K, while Minference and FlexPrefill only accelerate prefilling beyond 64K. During decoding, LCA maintains low per-token latency (reducing by **1.8**× at 128K) and reduces KV cache by **90%** (from 10.13 GB to 0.71 GB), whereas Minference and FlexPrefill cannot reduce the KV cache. KDA suffers severe performance degradation (Tables 2 and 3) despite maintaining a small cache, LCA

Table 4: Comparisons on short-context tasks.

Methods	MMLU	GSM-8K	MBPP
DeepSeek V2-Lite (*MLA*)	**57.12**	**41.47**	**54.09**
• FlexPrefill (Lai et al., 2025)	53.85	33.74	52.14
• MInference (Jiang et al., 2024)	51.01	32.90	46.69
• KDA (Kimi et al., 2025b)	56.31	37.45	50.97
• LCA (Ours)	57.04	41.17	53.31

Table 5: Ablation study on the pooling strategies for $\mathbf{c}^{\text{rep}}$ and $\mathbf{k}^{R_{\text{rep}}}$. "MaxSel" denotes max selection.

$\mathbf{c}_{\text{rep}} \setminus \mathbf{k}^{R_{\text{rep}}}$	MaxPool	MeanPool	WeightedPool	MaxSel
MaxPool	27.44	27.01	27.43	28.89
MeanPool	27.38	26.39	27.67	28.84
WeightedPool	27.93	27.04	27.83	**29.09**
MaxSel	26.94	27.65	27.79	28.93

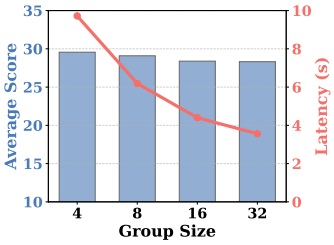

Figure 3: Ablations on g.

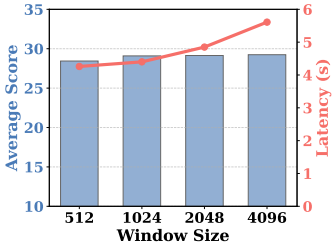

Figure 4: Ablations on w.

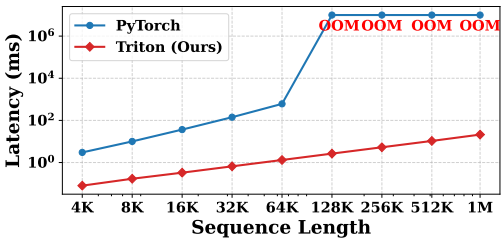

Figure 5: Latency comparison between our Triton kernel and the PyTorch implementation.

preserves competitive performance alongside its efficiency gains.

In Figures 5, we compare the latency of our custom Triton kernel with the PyTorch implementation. Our kernel achieves consistently lower latency across all context lengths, with up to **24.4**× speedup at 64K. While the PyTorch implementation encounters OOM beyond 64K, our kernel scales efficiently to **1M tokens** with feasible latency and linear memory growth. This optimization ensures that the efficiency advantages of LCA are fully realized in practice.

5.4 Ablations

All ablations are evaluated on the LongBench-E benchmark, reporting both the average score and the first-token latency at 128K context length.

Effect on Pooling Strategies. We investigate the pooling strategies for generating the representative latent $\mathbf{c}^{\text{rep}}$ and positional anchor $\mathbf{k}^{R_{\text{rep}}}$. In Table 5, the combination of **attention-weighted pooling** for $\mathbf{c}^{\text{rep}}$ and **max selection** for $\mathbf{k}^{R_{\text{rep}}}$ achieves the highest score. This validates our design: weighted pooling preserves semantic information from all tokens in the group, while max selection avoids distorting the nonlinear positional encoding.

Effect on Group Size. We investigate the impact of the group size g on model performance and efficiency by varying g in $\{4, 8, 16, 32\}$ while keeping the local window size fixed at $w = 1024$. As shown in Figure 3, smaller group sizes yield higher average scores on LongBench-E. This improvement stems from finer-grained condensation, which preserves more detailed semantic information within each group. However, this comes at the cost of increased computational latency. To balance efficiency and performance, we choose $g = 16$ as the default setting for all subsequent experiments, which provides a favorable trade-off.

Effect on Window Size. We evaluate the influence of the local window size w by varying it across $\{512, 1024, 2048, 4096\}$, with the group size fixed at $g = 16$. Larger windows preserve more local context and improve accuracy, yet also increase latency, as shown in Figure 4. We select $w = 1024$ as the default, offering a good compromise between preserving fine-grained local dependencies and maintaining inference efficiency.

6 Conclusion

In this paper, we propose LCA, an efficient attention mechanism that performs structured context condensation directly within MLA's latent space. LCA explicitly handles the disentangled semantic and positional components in MLA: aggregating semantic latents via query-aware pooling while preserving precise positional information through anchor selection. This approach jointly reduces KV cache size and attention complexity without introducing additional parameters. Theoretically, we prove that the approximation error of our method is uniformly bounded, independent of context length. Extensive experiments validate the effectiveness of LCA, demonstrating significant efficiency gains (up to **2.5**× prefilling speedup and **90%** KV cache reduction at 128K context) while maintaining competitive model performance.

Limitations

While LCA provides a general design, its implementation requires a custom kernel to achieve promising efficiency, which may involve additional engineering effort for deployment in new frameworks. The current implementation of LCA is optimized for common bfloat16/float16 precision settings, and its kernel-level optimizations have not been extensively explored for lower-precision formats (*e.g.*, int8 quantization). Additionally, the latency measurements were conducted on a homogeneous GPU cluster; performance on heterogeneous or memory-bandwidth-constrained hardware may vary. These are typical engineering and evaluation scoping considerations that do not affect the core algorithmic contribution, and they point to straightforward extensions in future work.

Information About Use Of AI Assistants

During the preparation of this work, we used large language models (LLMs) only for text polishing and grammar correction. All central research ideas, methodological designs, experimental implementations, and result interpretations are the original contributions of the authors. The AI assistants were not involved in any core intellectual content.

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. 2023. GQA: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, Singapore. Association for Computational Linguistics.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

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

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. 2023. Token merging: Your vit but faster. In *International Conference on Learning Representations*.

Yaofo Chen, Zeng You, Shuhai Zhang, Haokun Li, Yirui Li, Yaowei Wang, and Mingkui Tan. 2025. Core context aware transformers for long context language modeling. In *International Conference on Machine Learning*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*.

DeepSeek-AI. 2025. Deepseek-v3.2: Pushing the frontier of open large language models. *Preprint*, arXiv:2512.02556.

DeepSeek-AI, Aixin Liu, Bei Feng, and et.al. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *Preprint*, arXiv:2405.04434.

Jiayu Ding, Shuming Ma, Li Dong, Xingxing Zhang, Shaohan Huang, Wenhui Wang, Nanning Zheng, and Furu Wei. 2023. Longnet: Scaling transformers to 1,000,000,000 tokens. *arXiv preprint arXiv:2307.02486*.

Zhanzhou Feng and Shiliang Zhang. 2023. Efficient vision transformer via token merger. *IEEE Transactions on Image Processing*, 32:4156–4169.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. 2024. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, and 1 others. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint*.

Jitai Hao, Yuke Zhu, Tian Wang, Jun Yu, Xin Xin, Bo Zheng, Zhaochun Ren, and Sheng Guo. 2025. Omnikv: Dynamic context selection for efficient long-context llms. In *International Conference on Learning Representations*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *International Conference on Learning Representations*.

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

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

Kimi, Yifan Bai, Yiping Bao, and et.al. 2025a. Kimi k2: Open agentic intelligence. *arXiv*.

Kimi, Yu Zhang, Zongyu Lin, and et.al. 2025b. Kimi linear: An expressive, efficient attention architecture. *Preprint*, arXiv:2510.26692.

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

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

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

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, and 1 others. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*.

Xiang Liu, Zhenheng Tang, Peijie Dong, Zeyu Li, Yue Liu, Bo Li, Xuming Hu, and Xiaowen Chu. 2025. Chunkkv: Semantic-preserving kv cache compression for efficient long-context llm inference. In *Advances in Neural Information Processing Systems*.

Potsawee Manakul and Mark J. F. Gales. 2021. Longspan summarization via local attention and content selection. In *Association for Computational Linguistics*, pages 6026–6041.

OpenAI. 2023. GPT-4 technical report. *arXiv preprint*.

Noam Shazeer. 2019. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

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

Philippe Tillet, Hsiang-Tsung Kung, and David D. Cox. 2019. Triton: an intermediate language and compiler for tiled neural network computations. In *Proceedings of the International Workshop on Machine Learning and Programming Languages*, pages 10–19.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Shang Yang, Haotian Tang, Yao Fu, Song Han, and 1 others. 2025. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. In *International Conference on Learning Representations*.

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

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

Baosong Yang, Longyue Wang, Derek F Wong, Shuming Shi, and Zhaopeng Tu. 2021. Context-aware self-attention networks for natural language processing. *Neurocomputing*, 458:157–169.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and 1 others. 2020. Big bird: Transformers for longer sequences. *Advances in Neural Information Processing Systems*, 33:17283–17297.

Shuhai Zhang, Zeng You, Yaofo Chen, Zhiquan Wen, Qianyue Wang, Zhijie Qiu, Yuanqing Li, and Mingkui Tan. 2025. Curse of high dimensionality issue in transformer for long context modeling. In *International Conference on Machine Learning*.

In the supplementary, we provide a detailed proof, more details on the LCA method. We organize our supplementary as follows.

In Section A, we provide the detailed proof for Proposition 1 and Theorem 1.

In Section B, we analysis the computation and memory complexity of LCA in detail.

In Section C, we provide more experimental details of LCA.

A Theoretical Analysis

A.1 Proof of Proposition 1

We provide a complete and self-contained proof of Proposition 1, which establishes the optimality of attention-weighted pooling for latent-space condensation.

Proposition 1 (Optimal Condensation via Weighted Pooling) *Let $\{\mathbf{c}_i \in \mathbb{R}^{d_c}\}_{i=1}^{g}$ be a set of latent vectors and $\boldsymbol{\alpha} \in \mathbb{R}^g$ a probability distribution over them, with $\alpha_i \geq 0$ and $\sum_i \alpha_i = 1$. Consider the expected squared reconstruction error*

$$\mathcal{L}(\mathbf{c}^{rep}) = \mathbb{E}_{i\sim\boldsymbol{\alpha}}\left[\|\mathbf{c}_i - \mathbf{c}^{rep}\|_2^2\right] = \sum_{i=1}^{g} \alpha_i \|\mathbf{c}_i - \mathbf{c}^{rep}\|_2^2.$$

The vector $\mathbf{c}^{rep}$ that minimizes $\mathcal{L}$ is uniquely given by the convex combination $\mathbf{c}^{rep} = \sum_{i=1}^{g} \alpha_i \mathbf{c}_i$.

Proof. First, expand the squared Euclidean norm:

$$\begin{aligned}\|\mathbf{c}_i - \mathbf{c}^{\text{rep}}\|_2^2 &= (\mathbf{c}_i - \mathbf{c}^{\text{rep}})^\top (\mathbf{c}_i - \mathbf{c}^{\text{rep}}) \\ &= \mathbf{c}_i^\top \mathbf{c}_i - 2\mathbf{c}_i^\top \mathbf{c}^{\text{rep}} + \mathbf{c}^{\text{rep}\top}\mathbf{c}^{\text{rep}}.\end{aligned}$$

Substituting into the loss function yields

$$\begin{aligned}\mathcal{L}(\mathbf{c}^{\text{rep}}) = \sum_{i=1}^{g} \alpha_i \mathbf{c}_i^\top \mathbf{c}_i - 2\Big(\sum_{i=1}^{g} \alpha_i \mathbf{c}_i\Big)^\top \mathbf{c}^{\text{rep}} \\ + \Big(\sum_{i=1}^{g} \alpha_i\Big)\mathbf{c}^{\text{rep}\top}\mathbf{c}^{\text{rep}}.\end{aligned}$$

Since $\sum_i \alpha_i = 1$, the last term simplifies to $\mathbf{c}^{\text{rep}\top}\mathbf{c}^{\text{rep}}$. Define the weighted mean $\bar{\mathbf{c}} = \sum_{i=1}^{g} \alpha_i \mathbf{c}_i$. Then, we have

$$\mathcal{L}(\mathbf{c}^{\text{rep}}) = \sum_{i=1}^{g} \alpha_i \mathbf{c}_i^\top \mathbf{c}_i - 2\bar{\mathbf{c}}^\top \mathbf{c}^{\text{rep}} + \mathbf{c}^{\text{rep}\top}\mathbf{c}^{\text{rep}}.$$

To find the minimizer, we compute the gradient with respect to $\mathbf{c}^{\text{rep}}$:

$$\nabla_{\mathbf{c}^{\text{rep}}} \mathcal{L} = -2\bar{\mathbf{c}} + 2\mathbf{c}^{\text{rep}}.$$

Setting the gradient to zero gives the necessary condition:

$$\mathbf{c}^{\text{rep}} = \bar{\mathbf{c}} = \sum_{i=1}^{g} \alpha_i \mathbf{c}_i.$$

To confirm that this stationary point is indeed a minimum, we examine the Hessian matrix:

$$\nabla^2_{\mathbf{c}^{\text{rep}}} \mathcal{L} = 2\mathbf{I}_{d_c},$$

where $\mathbf{I}_{d_c}$ is the $d_c \times d_c$ identity matrix. This Hessian is positive definite for all $\mathbf{c}^{\text{rep}}$, implying that the function $\mathcal{L}$ is strictly convex. Thus, the point $\mathbf{c}^{\text{rep}} = \sum_i \alpha_i \mathbf{c}_i$ is the unique global minimizer. □

Remark 1. *In LCA, for a group G_j of size g, we treat the normalized attention scores $\alpha_i^{(j)}$ (Eq. (6)) as the probability distribution $\boldsymbol{\alpha}$. The vectors $\mathbf{c}_i^{KV}$ are the latent representations of the tokens in the group. Applying Proposition 1 demonstrates that the weighted pooling operation $\mathbf{c}_j^{rep} = \sum_{i\in G_j} \alpha_i^{(j)} \mathbf{c}_i^{KV}$ is the optimal choice for minimizing the expected reconstruction error of the group's latent contributions. This theoretical guarantee ensures that our condensation procedure maximally preserves the semantic information of the group, as measured by mean squared error.*

Comparison with Alternative Pooling Strategies. Other common pooling strategies include *mean pooling* ($\mathbf{c}^{\text{rep}} = \frac{1}{g}\sum_i \mathbf{c}_i$) and *max pooling* (selecting the vector with the highest norm or a heuristic score). Mean pooling is a special case of our weighted pooling where $\alpha_i = 1/g$ for all i, which implicitly assumes all tokens are equally important—an assumption that rarely holds in natural language. Max pooling, while preserving the most salient single vector, discards information from other tokens entirely and can be unstable. Our attention-weighted pooling generalizes mean pooling by incorporating token-specific importance, and it avoids the information-discarding drawback of max pooling. Proposition 1 shows that when the importance scores α_i accurately reflect relevance, weighted pooling is objectively optimal in the least-squares sense.

A.2 Proof of Theorem 1

We theoretically analyze the uniform error bound for LCA in Theorem 1. The argument closely follows the structure of the original theorem but employs the notation introduced in Section A that

reflects the actual grouping and condensation performed by LCA.

Notations. For a query at position t, partition the preceding tokens into the recent window $R = \{t-w+1, \dots, t\}$ and the distant history $H = \{1, \dots, t-w\}$. The distant history is further divided into m contiguous groups $G_1, \dots, G_m$, each of size g (except possibly the last). For a group G_j, LCA computes a representative key $\mathbf{k}_j^{\text{rep}}$ and a representative value $\mathbf{v}_j^{\text{rep}}$ via the condensation procedure described in Section 4.3. For a token $i \in H$, we denote by $G(i)$ the index of the group containing i.

The original MLA attention at position t is

$$\mathbf{Attn}_t = \sum_{i=1}^{t} p_i \mathbf{v}_i, \quad p_i = \frac{e^{a_i}}{Z},$$

where $a_i = \frac{\mathbf{q}_t^\top \mathbf{k}_i}{\sqrt{d_h}}$, $Z = \sum_{j=1}^{t} e^{a_j}$.

In LCA, the same query is compared with the representative keys for distant tokens and with the original keys for recent tokens. Hence the attention weights are defined by

$$\mathbf{Attn}_t^{\text{LCA}} = \sum_{i \in R} p'_i \mathbf{v}_i \; + \; \sum_{j=1}^{m} \beta_j \mathbf{v}_j^{\text{rep}},$$

where the logits for recent tokens are unchanged, *i.e.*, $b_i = a_i$ $(i \in R)$, and for a distant group G_j we use the single representative logit $b_j^{\text{rep}} = \frac{\mathbf{q}_t^\top \mathbf{k}_j^{\text{rep}}}{\sqrt{d_h}}$. The corresponding weights are

$$p'_i = \frac{e^{b_i}}{Z'} \; (i \in R), \qquad \beta_j = \frac{e^{b_j^{\text{rep}}}}{Z'}$$

$$Z' = \sum_{i \in R} e^{b_i} + \sum_{j=1}^{m} e^{b_j^{\text{rep}}}.$$

To compare the two outputs directly, it is convenient to introduce for every distant token $i \in H$ via the group-assigned key and value:

$$\tilde{\mathbf{k}}_i = \mathbf{k}_{G(i)}^{\text{rep}}, \qquad \tilde{\mathbf{v}}_i = \mathbf{v}_{G(i)}^{\text{rep}}.$$

For recent tokens $i \in R$ we simply set $\tilde{\mathbf{k}}_i = \mathbf{k}_i$, $\tilde{\mathbf{v}}_i = \mathbf{v}_i$. With this convention we can write the LCA output compactly as

$$\mathbf{Attn}_t^{\text{LCA}} = \sum_{i=1}^{t} p'_i \tilde{\mathbf{v}}_i,$$

where p'_i for $i \in H$ is defined as $p'_i = \beta_{G(i)}/|G(i)|$ (so that $\sum_{i \in G_j} p'_i = \beta_j$). The deviation hypotheses of the theorem are then

$$\|\mathbf{k}_i - \tilde{\mathbf{k}}_i\|_2 \le \delta_k, \; \|\mathbf{v}_i - \tilde{\mathbf{v}}_i\|_2 \le \delta_v, \; (i = 1, \dots, t).$$

Theorem 1 (Uniform Error Bound) *Fix a query $\mathbf{q}_t$ with $\|\mathbf{q}_t\|_2 \le Q$ and assume $\|\mathbf{v}_i\|_2 \le V$ for all values. For each distant-history group G_j, let $\mathbf{k}_j^{\text{rep}}, \mathbf{v}_j^{\text{rep}}$ be the representative key and value that LCA uses for every token $i \in G_j$, and let δ_k, δ_v be uniform bounds on the per-token key and value deviations: $\|\mathbf{k}_i - \mathbf{k}_j^{\text{rep}}\|_2 \le \delta_k$, $\|\mathbf{v}_i - \mathbf{v}_j^{\text{rep}}\|_2 \le \delta_v$. Then the ℓ_2 error between the full MLA attention and LCA satisfies*

$$\left\|\mathbf{Attn}_t - \mathbf{Attn}_t^{\text{LCA}}\right\|_2 \; \le \; V\left(e^{2Q\delta_k/\sqrt{d_h}} - 1\right) + \delta_v.$$

Proof. Define the total error $E := \mathbf{Attn}_t - \mathbf{Attn}_t^{\text{LCA}}$. A straightforward algebraic manipulation gives the decomposition:

$$E = \underbrace{\sum_{i=1}^{t} (p_i - p'_i)\mathbf{v}_i}_{\text{(I)}} \; + \; \underbrace{\sum_{i=1}^{t} p'_i (\mathbf{v}_i - \tilde{\mathbf{v}}_i)}_{\text{(II)}}.$$

We next bound the norms of (I) and (II) separately.

1) We start by bounding the first term. First, we bound the logit perturbations. Using the Cauchy–Schwarz inequality and the given bounds on the query norm and key approximations,

$$|a_i - b_i| = \frac{|\mathbf{q}_t^\top (\mathbf{k}_i - \tilde{\mathbf{k}}_i)|}{\sqrt{d_h}} \le \frac{\|\mathbf{q}_t\|_2 \|\mathbf{k}_i - \tilde{\mathbf{k}}_i\|_2}{\sqrt{d_h}}$$

$$\le \frac{Q\delta_k}{\sqrt{d_h}} =: \varepsilon,$$

where b_i denotes the logit used by LCA for token i (i.e., $b_i = a_i$ for $i \in R$ and $b_i = b_{G(i)}^{\text{rep}}$ for $i \in H$). Hence each logit changes by at most ε.

Thus, we have $e^{-\varepsilon} e^{b_i} \le e^{a_i} \le e^{\varepsilon} e^{b_i}$ for every i. Summing these inequalities over all i yields

$$e^{-\varepsilon} \sum_{j=1}^{t} e^{b_j} \; \le \; \sum_{j=1}^{t} e^{a_j} \; \le \; e^{\varepsilon} \sum_{j=1}^{t} e^{b_j}.$$

Denote $Z = \sum_j e^{a_j}$ and $Z' = \sum_j e^{b_j}$. The previous chain implies $e^{-\varepsilon} Z' \le Z \le e^{\varepsilon} Z'$. Combining this with the pointwise bound $e^{-\varepsilon} e^{b_i} \le e^{a_i}$ gives

$$p_i = \frac{e^{a_i}}{Z} \ge \frac{e^{-\varepsilon} e^{b_i}}{e^{\varepsilon} Z'} = e^{-2\varepsilon} p'_i.$$

Similarly, $p_i \le e^{2\varepsilon} p'_i$. Therefore,

$$e^{-2\varepsilon} p'_i \le p_i \le e^{2\varepsilon} p'_i \qquad \text{for all } i. \qquad (11)$$

From Eqn. (11), we obtain $|p_i - p'_i| \le (e^{2\varepsilon} - 1) p'_i$. Summing this inequality over i provides a bound on the ℓ_1 distance between the two weight vectors:

$$\sum_{i=1}^{t} |p_i - p'_i| \; \le \; (e^{2\varepsilon} - 1) \sum_{i=1}^{t} p'_i = e^{2\varepsilon} - 1. \quad (12)$$

Using the uniform bound $\|\mathbf{v}_i\|_2 \le V$ together with Eqn. (12),

$$\Big\| \sum_{i=1}^{t} (p_i - p'_i)\mathbf{v}_i \Big\|_2 \le \sum_{i=1}^{t} |p_i - p'_i| \cdot \|\mathbf{v}_i\|_2$$

$$\le V \sum_{i=1}^{t} |p_i - p'_i| \le V\big(e^{2\varepsilon} - 1\big). \tag{13}$$

2) We bound the second error component. Given $\|\mathbf{v}_i - \tilde{\mathbf{v}}_i\|_2 \le \delta_v$ and the fact that p'_i form a probability distribution, we have

$$\Big\| \sum_{i=1}^{t} p'_i(\mathbf{v}_i - \tilde{\mathbf{v}}_i) \Big\|_2 \le \sum_{i=1}^{t} p'_i \|\mathbf{v}_i - \tilde{\mathbf{v}}_i\|_2$$

$$\le \delta_v \sum_{i=1}^{t} p'_i = \delta_v. \tag{14}$$

3) Combing Eqn. (13) and (14) gives

$$\|E\|_2 \;\le\; V\big(e^{2\varepsilon} - 1\big) \;+\; \delta_v,$$

where $\varepsilon = Q\delta_k/\sqrt{d_h}$.

□

B Complexity Analysis

We analyze the computational and memory complexity of LCA against standard MLA. Recall that L is the sequence length, w is the short-range window size, and m is the number of long-range representatives.

Computational complexity. The overall computation consists of three main steps. First, importance scores are computed between a group-aware query $\bar{\mathbf{q}}$ and all L keys, requiring $\mathcal{O}(L)$ operations. Second, for each of the m groups, we compute normalized attention weights (softmax over g scores) and perform weighted pooling of g latent vectors, which together cost $\mathcal{O}(L)$. Finally, attention is computed over the condensed context of size $m + w$ for each of the L queries, resulting in $\mathcal{O}\big(L(m+w)\big)$ operations. Since w is a constant, the dominant term becomes $\mathcal{O}(Lm)$, lower than the $\mathcal{O}(L^2)$ complexity of standard MLA.

Memory complexity of the KV cache. In LCA, we cache only the m representative latent vectors $\{\mathbf{c}_j^{\text{rep}}\}_{j=1}^{m}$ and their positional anchors $\{\mathbf{k}_j^{R_{\text{rep}}}\}_{j=1}^{m}$, along with the w recent tokens stored in full. Hence the total cache size scales as $\mathcal{O}(m + w) = \mathcal{O}(m)$.

The reduction in both computation and memory enables LCA to handle long contexts with significantly lower hardware requirements while preserving essential information through structured latent-space condensation.

C More Experimental Details

C.1 More Details about Evaluation Benchmarks

We evaluate our method on both long-context and short-context tasks to comprehensively assess its performance and efficiency. For long-context understanding, we use two established benchmarks: **LongBench-E** (Bai et al., 2023), a bilingual suite with uniformly distributed context lengths that measures overall context understanding across 21 diverse tasks; and **RULER** (Hsieh et al., 2024), a synthetic benchmark with 13 subtasks (e.g., retrieval, multi-hop reasoning, aggregation) designed for fine-grained assessment of long-context capabilities.

Furthermore, to verify that our efficiency gains do not compromise performance on standard tasks, we evaluate on three widely-used short-context benchmarks: **MMLU** (Hendrycks et al., 2021), which assesses knowledge breadth and reasoning across 57 academic subjects; **GSM8K** (Cobbe et al., 2021), a dataset of grade-school math problems requiring multi-step numerical reasoning; and **MBPP** (Austin et al., 2021), which evaluates code generation ability through crowd-sourced programming tasks.

C.2 More Implementation Details

We implement and evaluate our proposed LCA on the DeepSeek-V2-Lite model (16B parameters) (DeepSeek-AI et al., 2024). We choose this model for two primary reasons: 1) it is the first publicly released model that adopts MLA, providing an ideal foundation for our method which specifically targets MLA's efficiency bottlenecks; 2) its moderate size makes it feasible to conduct extensive experiments and latency measurements within constrained GPU resources. We replace the original MLA layers in DeepSeek-V2-Lite with our LCA. To ensure high efficiency, we develop a highly optimized kernel using Triton (Tillet et al., 2019). This custom implementation minimizes memory movement and maximizes hardware utilization during both training and inference. To allow the pretrained model to adapt to our dynamic condensation mechanism, we perform continued fine-tuning for only 1,000 steps with 2.0B tokens on the per-source-length upsampled SlimPajama dataset (Fu et al., 2024). Each training sample has a sequence length of 64K tokens. We use a total batch size

of 64, implemented with a micro-batch size of 1 and gradient accumulation steps of 8. Training is conducted using the Deepspeed Zero-3 strategy for efficient memory management. The learning rate is set to 5×10^{-6}. We set the group size $g = 16$ and the window size $w = 1024$, respectively. These settings are kept fixed when inference across all experiments. All experiments and latency measurements are performed on a server equipped with 8×H200 GPUs.

C.3 More Details of Compared Methods

To comprehensively evaluate the effectiveness and efficiency of LCA, we compare it against several state-of-the-art efficient attention methods. We benchmark against two recent published methods that perform dynamic sparsification: **FlexPrefill** (Lai et al., 2025) and **MInference** (Jiang et al., 2024). Note that these methods were originally designed for and evaluated on vanilla self-attention. They cannot be directly applied to MLA's low-rank latent space. For a fair comparison, we adapt these methods to work on the *reconstructed* key and value matrices derived from MLA's latents. This ensures functional equivalence but forfeits the opportunity to exploit the compressed latent structure for further gains. **FlexPrefill** dynamically selects between a Query-Aware pattern and a Vertical-Slash pattern per attention head, adaptively determining which key-value indices are necessary for computation. We use the official implementation and adhere to the recommended hyperparameters: a score ratio $\gamma = 0.9$, a threshold $\tau = 0.1$, a minimum retained budget of 1,024 tokens, and a block size of 128. **MInference** determines optimal static sparse patterns per attention head offline, combined with online dynamic adjustment of computation regions. Following the original paper, we perform offline calibration on the DeepSeek-V2-Lite model to obtain a sparse configuration and use it for online inference.

We also compare against the recently proposed **Kimi Delta Attention (KDA)** (Kimi et al., 2025b), a gated linear attention variant that introduces a fine-grained diagonalized gate for controlling memory decay and positional awareness. A critical distinction lies in the training requirements: the original KDA is designed to be integrated into the model during pre-training from scratch, requiring substantial computational resources and large-scale pre-training data to achieve optimal performance. In contrast, our LCA requires only minimal continued fine-tuning to recover full performance on a pre-trained base. To establish a controlled and equitable comparison under constrained resources, we apply the same adaptation protocol to KDA as we do for our method. Specifically, we take the base DeepSeek-V2-Lite model, augment it with the additional parameters required by KDA, apply KDA to the reconstructed KV matrices, and fine-tune it for 1,000 steps on the same per-source-length upsampled SlimPajama dataset (Fu et al., 2024) used for LCA.