DAM: Dynamic Attention Mask for Long-Context Large Language Model Inference Acceleration

Hanzhi Zhang[1], Heng Fan[1], Kewei Sha[2], Yan Huang[1], Yunhe Feng[1]

[1]LLaVi Lab, Department of Computer Science & Engineering; [2]Department of Data Science
University of North Texas

{hanzhi.zhang, heng.fan, kewei.sha, yan.huang, yunhe.feng}@unt.edu

Abstract

Long-context understanding is crucial for many NLP applications, yet transformers struggle with efficiency due to the quadratic complexity of self-attention. Sparse attention methods alleviate this cost but often impose static, predefined masks, failing to capture heterogeneous attention patterns. This results in suboptimal token interactions, limiting adaptability and retrieval accuracy in long-sequence tasks. This work introduces a dynamic sparse attention mechanism that assigns adaptive masks at the attention-map level, preserving heterogeneous patterns across layers and heads. Unlike existing approaches, our method eliminates the need for fine-tuning and predefined mask structures while maintaining computational efficiency. By learning context-aware attention structures, it achieves high alignment with full-attention models, ensuring minimal performance degradation while reducing memory and compute overhead. This approach provides a scalable alternative to full attention, enabling the practical deployment of large-scale Large Language Models (LLMs) without sacrificing retrieval performance. *DAM* is available at: https://github.com/HanzhiZhang-Ulrica/DAM.

1 Introduction

Understanding long contexts is essential for document summarization, question answering, and retrieval-augmented generation. Long-context NLP applications power legal analysis, financial reporting, and knowledge graph construction, where maintaining coherence across tokens is critical. However, existing LLMs struggle with long sequences due to inefficiency in self-attention.

LLMs leverage the Transformer architecture, which models long-range token dependencies through self-attention, enabling direct interactions between all tokens. Multi-head attention enhances expressivity by capturing multiple token relationships, while positional embeddings preserve order information. Dynamic token representations allow contextual meaning to evolve across layers, ensuring coherent and accurate long-term recall.

However, transformers process long contexts inefficiently. Quadratic complexity makes full attention computationally prohibitive, forcing models to truncate inputs, leading to information loss in long-document tasks. Attempts to mitigate this through fixed context windows or uniform attention fail to distinguish between critical and redundant information. Meanwhile, streaming applications suffer from recomputing at every step, as each newly generated token attends to all previous tokens. This results in redundant computation, growing memory usage, and increased latency, making real-time processing impractical.

The inability to process long contexts also directly impacts businesses and researchers. Legal and financial institutions rely on AI to analyze contracts and reports(Pingili, 2025), but truncated inputs cause critical details to be lost. AI assistants in customer service forgetting past interactions may fail to maintain coherent conversations. Researchers pushing transformer efficiency face skyrocketing computational costs, making large-scale deployment unsustainable. Without an efficient solution, enterprises must rely on costly and ineffective workarounds like document chunking, which may destroy contextual coherence.

Sparse attention is an efficiency-driven approach to mitigating long-context inefficiency in generative LLMs by enforcing structured sparsity. Figure 1(b) illustrates static sparse attention methods that reduce computational cost by enforcing fixed-span sliding window and global masks across all heads and input lengths. This approach improves efficiency but sacrifices flexibility, forcing models to rely only on local interactions. As a result, these models fail to adapt to long-range dependencies, reducing accuracy in complex retrieval and reasoning tasks. Figure 1(c) improves flexibility by assigning

Findings of the Association for Computational Linguistics: ACL 2025, pages 4663–4676

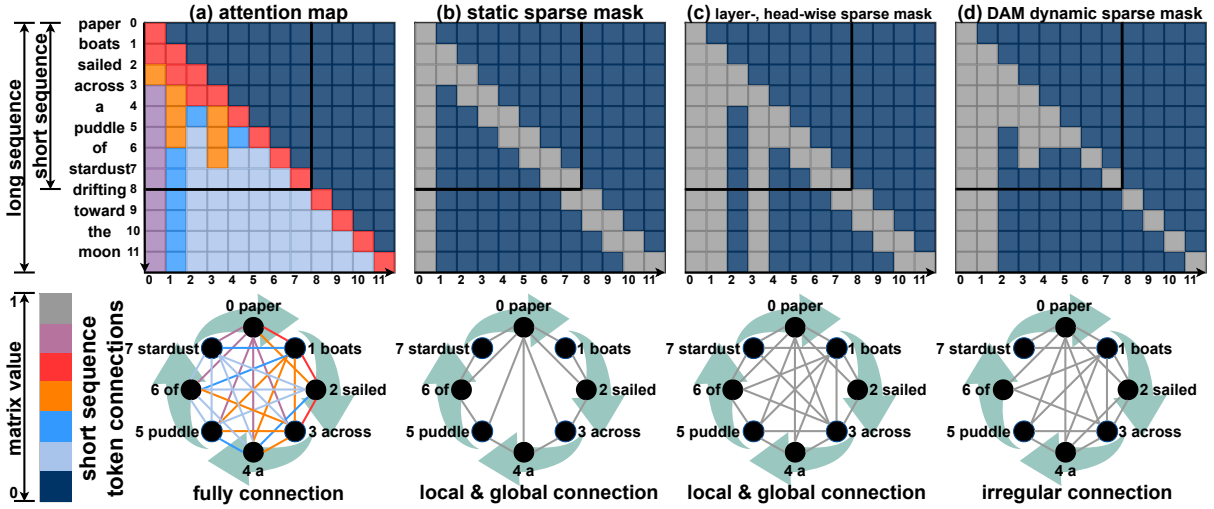

Figure 1: Attention patterns from $queries \times keys$. The short input sequence "paper boats sailed across a puddle of stardust" is a subset of the longer input sequence "paper boats sailed across a puddle of stardust drifting toward the moon". (a) Each query attends to all keys, and longer attention patterns are extensions of short patterns. (b) Static attention map captures same classical global and sliding-window patterns to attention maps from every layers and heads. (c) Different masks are assigned to corresponding maps with predefined patterns. (d) Heterogeneous map remains feature patterns from each attention map.

different masks to layers and heads, removing the need for fine-tuning. However, it assumes attention structures can be predefined, failing to capture heterogeneous token interactions that emerge dynamically. The result is a rigid sparsity pattern that still requires processing all sequence lengths, making it resource-intensive for long-context applications.

This work introduces *DAM*, a novel framework for dynamic sparse attention, as illustrated in Figure 1(d). *DAM* generates adaptive sparse attention masks at the granularity of individual attention maps, thereby capturing both layer-specific structural patterns and input-dependent variations in attention. In contrast to prior approaches that often rely on fixed or globally-defined sparsity patterns, *DAM* preserves the heterogeneity of attention patterns across different layers and heads, leading to improved expressiveness. Furthermore, the method eliminates the need for manual, task-specific fine-tuning of the sparsity structure, while maintaining the computational benefits of sparse attention.

Our contributions are summarized as follows:

- We propose a dynamic sparse attention framework that assigns distinct, adaptive sparse masks to each attention map, preserving heterogeneous patterns across heads and layers.
- Our approach is fine-tuning-free and generalizes seamlessly to varying input lengths, eliminating the need for manual sparsity pattern design.
- We incorporate a flexible "true mask" mechanism to focus attention on relevant regions, reducing unnecessary computations on padding tokens or less informative areas.
- We demonstrate that *DAM* achieves performance comparable to full-attention models while improving computational efficiency.

2 Related Work

Attention mechanisms enable transformers to model dependencies across sequences but introduce computational challenges at scale. Researchers have explored multiple strategies to address these inefficiencies, including KV-caching for faster inference, sparse and hierarchical attention for memory reduction, state-space models for efficient streaming, and hybrid architectures for improved long-term memory tracking. While these approaches enhance scalability, each introduces trade-offs that limit their applicability to long-sequence processing.

KV-cache enhances autoregressive decoding by storing key and value representations from previous steps, allowing reuse instead of recomputing attention scores for all tokens (Ge et al., 2023; Li et al., 2024; Zhang et al., 2024b; Zhao et al., 2024; Chen et al., 2024; Liu et al., 2024; Adnan et al., 2024; Ge et al., 2023). This reduces redundant computation and accelerates inference but increases memory usage, limiting scalability for long sequences (Zhang et al., 2024a; Ye et al., 2024; Hu et al., 2024). Cache management adds complexity, and performance gains depend on reuse efficiency (Zheng et al., 2024b,a; Xiong et al., 2024; Gao et al., 2024). While KV-cache mitigates inefficiencies in autoregressive generation, it does not reduce the fundamental complexity of self-attention.

Sparse attention reduces token interactions to improve efficiency (Child et al., 2019; Yun et al., 2020; Ho et al., 2019). Static sparse attention applies predefined masks across all processed sentences to lower computational cost and improve hardware utilization (Roy et al., 2021; Kitaev et al., 2020; Tay et al., 2019; Choromanski et al., 2020). Common approaches include global, sliding window, and random masks, with local attention patterns enabling KV-cache eviction beyond the attention span to reduce memory usage (Beltagy et al., 2020a; Ainslie et al., 2004; Zaheer et al., 2020). However, static masks remain uniform across layers and heads, ignoring token-specific dependencies. This rigidity leads to information loss in long-sequence tasks, where retrieval accuracy relies on adapting attention spans dynamically.

Other strategies generate distinct masks by leveraging statistical information, defining role-specific constraints for attention heads, or introducing context-dependent sparsity (Wang et al., 2020; Fu et al., 2024; Correia et al., 2019). While these approaches increase flexibility, they fail to dynamically capture heterogeneous attention within individual maps and still process all sequence lengths, raising resource costs.

To introduce flexibility, some approaches assign different predefined sparse patterns to layers and heads (Fu et al., 2024; Wang et al., 2020; Fu et al.; Correia et al., 2019). They select masks based on input length, improving adaptability without requiring fine-tuning. However, it assumes optimal attention structures are predefined, missing dynamic token interactions, and require evaluating multiple sequence lengths, thereby increasing computational overhead. This limitation motivates methods to infer sparse structures without exhaustive manual design or repeated inference.

3 Preliminaries

Transformer models adopt the scaled dot-product attention mechanism, a core component for capturing relationships between tokens in a sequence (Vaswani, 2017). Attention scores are calculated as $S = \frac{QK^\top}{\sqrt{d_k}}$, where $Q \in \mathbb{R}^{n\times d_k}$ and $K \in \mathbb{R}^{m\times d_k}$ denote the query and key matrices, respectively. Here, n and m denote the number of query and key/value vectors, while d_k represents the dimensionality of each key/query vector. The resulting matrix $S \in \mathbb{R}^{n\times m}$ contains the unnormalized attention logits, representing the pairwise similarities between queries and keys. The scaling factor $\frac{1}{\sqrt{d_k}}$ is crucial for maintaining numerical stability during training, preventing the dot products from growing excessively large, which can lead to vanishing gradients during backpropagation. This scaling mitigates issues caused by large variances in the logits, particularly when applying a masking operation.

The computation of attention scores for all pairs of tokens has a quadratic time complexity of $\mathcal{O}(n^2)$ with respect to the sequence length, which becomes computationally expensive for long sequences. Sparse attention mechanisms address this computational bottleneck by imposing structured sparsity on the attention matrix. This is achieved through a binary mask $M_{\ell,h} \in \{0,1\}^{n\times m}$ for each layer ℓ and head h, defined as:

$$M_{\ell,h,i,j} = \begin{cases} 1, & \text{if token } i \text{ attends to token } j \\ & \text{in layer } \ell \text{ and head } h, \\ 0, & \text{otherwise.} \end{cases}$$

The mask $M_{\ell,h}$ is applied element-wise to the attention logits $S' = S \odot M_{\ell,h}$, where $\odot$ denotes the Hadamard product (element-wise multiplication). This effectively prevents attention between specific token pairs. The masked attention logits are then normalized using the softmax function:

$$A_{ij} = \frac{\exp(S'_{ij})}{\sum_{k=1}^{m} \exp(S'_{ik})}.$$

The output of the attention mechanism is computed as a weighted sum of the values, where $V \in \mathbb{R}^{m\times d_v}$ is the value matrix as $O = AV$.

While sparse attention mechanisms substantially improve computational efficiency, they inherently restrict the model's ability to learn long-range dependencies by limiting token interactions. A key limitation of many sparse attention approaches is their reliance on *fixed* sparsity patterns. Such patterns are unable to adapt to the dynamic nature of attention, including variations in sequence length and the diversity of attention distributions across different inputs. This rigidity can result in a significant reduction in performance, especially when dealing with long sequences or tasks requiring the modeling of intricate relationships. Moreover, predefined sparse attention structures like sliding window (Beltagy et al., 2020b) or global attention (Liu et al., 2021) often overlook the critical variations in attention patterns that occur across different layers and heads within the network. The optimal set of token interactions evolves across layers, rendering fixed sparsity patterns a bottleneck. This motivates

the need for a dynamic, structure-aware sparsity mechanism that adapts to position-wise attention patterns while maintaining compatibility with pre-trained transformer architectures.

4 Dynamic Attention Masks (*DAM*)

This section introduces our proposed Dynamic Attention Mask (*DAM*) mechanism. We first motivate the design by illustrating the dynamic nature of attention patterns across layers and heads in Transformer models. Then, we detail the architecture of *DAM*, and finally, we describe its integration into the standard Transformer framework.

4.1 Dynamic Attention Patterns

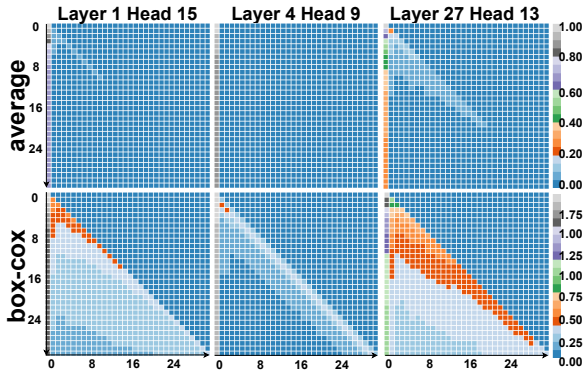

Figure 2: Visualization of dynamic attention patterns across layers and heads. The figure compares the effect of averaging (top row) and applying the Box-Cox transformation (bottom row) to attention values from a LLaMA 3.2 3B Instruct model on the Multi-News dataset. The Box-Cox transformation enhances the visibility of dynamic patterns.

Prior research has investigated and validated the existence of dynamic attention patterns across attention heads and layers in Transformer models (Goindani and Shrivastava, 2021; Xiao et al., 2024a). To visualize these patterns, we analyze attention maps obtained from a LLaMA 3.2 3B Instruct model (AI, 2024) evaluated on the Multi-News summarization benchmark (Fabbri et al., 2019). Figure 2 presents these attention maps, revealing inconsistencies in the underlying sparse structures across different heads and layers. The top row of Figure 2 displays the average attention values across the dataset. While these average maps suggest the presence of dynamic patterns, the patterns themselves are not readily discernible, hindering a deeper understanding and impeding the design of effective sparsity-inducing techniques.

We posit that enhancing the contrast between significant and less significant attention values can reveal these dynamic patterns more clearly. Specifically, we aim to preserve the largest attention values (e.g., those corresponding to the leftmost column in each attention map), while simultaneously accentuating the intermediate values (e.g., those distributed along the diagonal) and differentiating them from the smallest values (e.g., those in the bottom-left regions). To achieve this, we evaluated **nine different transformation methods** (more details in the Appendix B) and found the Box-Cox transformation (Box and Cox, 1964) consistently yielded the most informative visualizations, as shown in the bottom row of Figure 2.

The Box-Cox transformation enhances the visualization clarity of attention maps and **amplifies small and medium attention values**, making subtle yet important structural patterns more discernible, while **preserving the scale of larger values without introducing distortion**. It directly facilitates the intuitive selection and tuning of the threshold parameters (τ in Section 4.2.3 and μ in Section 4.2.4).

4.2 Two-Stage Dynamic Attention Masks

DAM enhances the efficiency of Transformer models by learning adaptive sparse attention masks. It addresses the limitation of predefined sparsity patterns, which can discard valuable, low-magnitude connections, by dynamically adjusting the attention mask based on observed attention patterns. The framework operates in two stages in Figure 3. First, a frozen pre-trained model processes input sequences (truncated to a manageable Pattern Capture Length, PCL) to extract full attention maps. These maps undergo a Box-Cox transformation for normalization, followed by thresholding to generate "true masks" representing key dependencies. Structural pattern analysis (identifying vertical and diagonal patterns) then enables the extrapolation of these masks to lengths exceeding the PCL, creating "extended masks". The Appendix A describes the detailed extension observation.

The second stage applies these generated, adaptive sparse attention masks (either true or extended, depending on sequence length) to a sparse Transformer model. This application occurs *before* the softmax operation within the attention mechanism, effectively limiting computations to the unmasked connections. This significantly reduces both memory and computational overhead compared to full attention, while preserving the crucial dependencies identified in the first stage. By focusing on

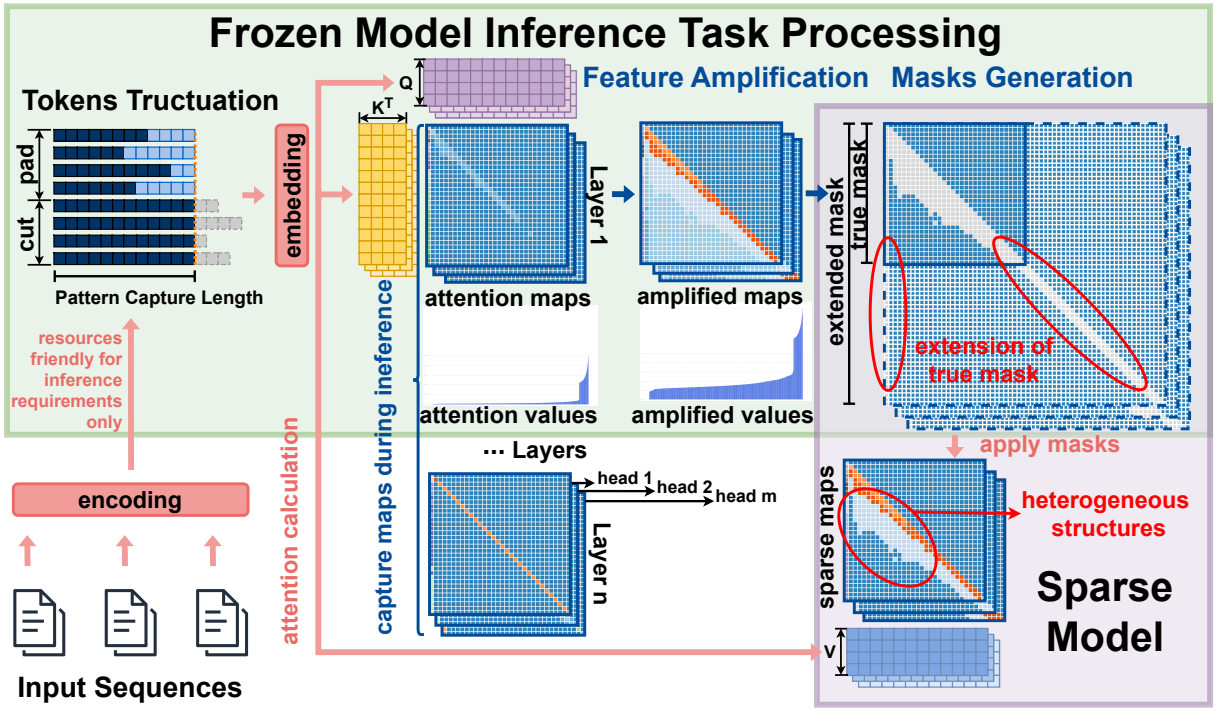

Figure 3: Two-stage *DAM* overview. The first stage extracts full attention patterns from sequences of varying lengths, applies a Box-Cox transformation, and generates masks that capture essential dependencies. The second stage applies these masks to a sparse model, enabling efficient inference while preserving key attention structures.

observed attention patterns and extrapolating structural regularities, *DAM* achieves a balance between efficiency and the ability to capture nuanced, long-range dependencies in input sequences, making it suitable for processing long sequences that would otherwise be computationally prohibitive.

4.2.1 Pattern Capture Length (PCL)

The Pattern Capture Length (PCL), denoted as L, represents a critical parameter within the *DAM* framework. It defines the maximum sequence length processed by the *frozen* model to extract the initial, full attention distributions. This constraint is essential for maintaining computational feasibility, particularly given the quadratic complexity of full attention mechanisms.

Let S represent the length of an input sequence. The PCL, L, is determined as $L = \min(S, L_{\max})$ where $L_{\max}$ is the maximum sequence length for which full attention computation remains computationally tractable given the available resources (e.g., GPU memory). As shown in Table 1, the original LLaMA 3.2 3B model runs out of memory (OOM) when processing sequences longer than 8k tokens on an A100 GPU (40GB). Based on this, we **select the longest sequence length that the hardware can stably support**, and then adjust downward only if necessary. Unlike tuning from small to large values, which is inefficient and error-prone, starting from the maximum supported length and adjusting downward as needed makes PCL tuning both simple and reliable.

In essence, the PCL acts as a truncation point, ensuring that the initial attention map extraction is performed on sequences of a manageable length, while still capturing representative attention patterns. The choice of $L_{\max}$ is a hyperparameter that depends on the specific hardware and model architecture.

4.2.2 Feature Amplification via Box-Cox

This section details the process of amplifying and normalizing attention scores using the Box-Cox transformation. This step aims to address the often-observed skewness in attention distributions, where a few connections dominate while many others have very low values. By amplifying smaller attention values, we reveal potentially significant connections that might otherwise remain masked.

First, mean attention scores are computed across all valid position pairs within the dataset. Let $A_{\ell,h,i,j}$ denote the accumulated attention value at layer ℓ, head h, from token position i to token position j, summed across multiple batches. A binary mask $m_{i,j}^{(b)} \in \{0, 1\}$ indicates whether the attention weight for position pair (i, j) was computed in batch b. The count matrix $C_{\ell,h,i,j}$ records the number of times each position pair (i, j) appears across

all batches, we have $C_{\ell,h,i,j} = \sum_b m_{i,j}^{(b)}$. The mean attention score, $\bar{A}_{\ell,h,i,j}$, is then calculated as:

$$\bar{A}_{\ell,h,i,j} = \frac{A_{\ell,h,i,j}}{C_{\ell,h,i,j} + \epsilon},$$

where ϵ is a small constant (e.g., 10^{-8}) added to the denominator to prevent division by zero and ensure numerical stability.

To mitigate the skewness of the attention scores and emphasize smaller values, a Box-Cox transformation is applied. To ensure the input to the transformation is strictly positive, a small constant ϵ is added to the mean attention scores as $X_{\ell,h,i,j} = \max(\bar{A}_{\ell,h,i,j}, \epsilon)$. The Box-Cox transformation is then applied to $X_{\ell,h,i,j}$ as follows:

$$B_{\ell,h,i,j} = \begin{cases} \frac{X_{\ell,h,i,j}^{\lambda} - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \ln(X_{\ell,h,i,j}), & \text{if } \lambda = 0 \end{cases}$$

where λ is the transformation parameter. In practice, we find that $\lambda = 0.5$ improves visualization, which does not require to be tuned in the future.

To ensure the transformed values $B_{\ell,h,i,j}$ remain non-negative, we subtract the minimum value across all heads and layers:

$$B^*_{\ell,h,i,j} = B_{\ell,h,i,j} - \min_{\ell',h',i',j'}(B_{\ell',h',i',j'}).$$

Finally, the normalized attention map $\tilde{A}_{\ell,h,i,j}$ is defined as $\tilde{A}_{\ell,h,i,j} = B^*_{\ell,h,i,j}$. With amplified smaller values, $\tilde{A}_{\ell,h,i,j}$ is then used for subsequent mask generation.

4.2.3 True Mask Generation

We detail the process of generating "true masks", denoted as $M_{\ell,h}$, which represent the binarized and thresholded version of the normalized attention maps. These masks serve as the basis for identifying structural patterns and subsequently constructing the extended, sparse attention masks.

A binary thresholding operation is applied to the normalized attention maps, $\tilde{A}_{\ell,h}$ (obtained as described in Section 4.2.2), to produce the true masks. Each true mask $M_{\ell,h}$ has the same dimensions as the corresponding attention map: $M_{\ell,h} = [m_{i,j}] \in \{0,1\}^{L \times L}$, where L is the Pattern Capture Length (PCL). The elements of the true mask, $m_{i,j}$, are determined by comparing the corresponding normalized attention values, $\tilde{A}_{\ell,h,i,j}$, to a predefined threshold, τ:

$$m_{i,j} = \begin{cases} 1, & \text{if } \tilde{A}_{\ell,h,i,j} \geq \tau, \\ 0, & \text{if } \tilde{A}_{\ell,h,i,j} < \tau. \end{cases}$$

This thresholding operation is applied independently to each layer ℓ and attention head h. The threshold, τ, acts as a hyperparameter controlling the sparsity of the true masks. A higher value of τ results in a sparser mask, retaining only the strongest attention connections.

4.2.4 Dynamic Mask Generation via Structural Pattern Matching

We construct *DAM* by identifying and combining predefined structural patterns within the true attention masks. A pattern pool, $\mathcal{P}$, is defined, consisting of a set of predefined attention patterns. Each pattern is represented as a binary matrix $P_k = [p_{i,j}] \in \{0,1\}^{L \times L}$, where L denotes the PCL and P_k represents the k-th pattern in the pool. The pattern pool, in this work, includes diagonal and vertical patterns, reflecting common attention structures observed in Transformer models.

A diagonal pattern, $P_{\text{diag},r}$, starts at row index r and extends diagonally downwards:

$$p_{i,j} = \begin{cases} 1, & \text{if } j = i - r, \\ 0, & \text{otherwise.} \end{cases}$$

for $r \in \{0, 1, \ldots, L-1\}$. A vertical pattern, $P_{\text{vert},c}$, captures column-wise attention (i.e., tokens attending to a specific column c):

$$p_{i,j} = \begin{cases} 1, & \text{if } j = c \text{ and } i \geq c, \\ 0, & \text{otherwise.} \end{cases}$$

for $c \in \{0, 1, \ldots, L-1\}$. The complete pattern pool is the union of these sets:

$$\mathcal{P} = \{P_{\text{diag},r}\} \cup \{P_{\text{vert},c}\}.$$

Each true mask $M_{\ell,h}$ is compared against patterns in $\mathcal{P}$. The match score γ_k for a pattern P_k is computed as:

$$\gamma_k = \frac{\sum_{i,j} M_{\ell,h}^{(i,j)} \cdot P_k^{(i,j)}}{\sum_{i,j} P_k^{(i,j)}}.$$

A pattern P_k is considered a valid match if its match score γ_k exceeds a predefined threshold μ, where $\mu \in [0,1]$ is a hyperparameter controlling the sensitivity of the pattern matching. Higher values of μ lead to fewer patterns being matched, resulting in sparser masks. It is robust across a relatively wide range, from 0.7 to 1.0, while still preserving the model's language understanding capabilities.

The extended mask, $\tilde{M}_{\ell,h}$, is constructed by extrapolating the structural patterns identified in the true masks, which are initially computed using sequences up to the PCL. These patterns—such as diagonal and vertical structures—are selected from a predefined pattern pool. Because patterns may overlap, the extended mask is formed by summing

all matched patterns whose match score exceeds a threshold:

$$\tilde{M}_{\ell,h} = \sum_{P_k \in \mathcal{P}, \gamma_k \geq \mu} P_k.$$

Finally, to ensure the extended mask is binary, a thresholding operation is applied:

$$\tilde{M}_{\ell,h}^{(i,j)} = \begin{cases} 1, & \text{if } \sum_{P_k \in \mathcal{P}, \gamma_k \geq \mu} P_k^{(i,j)} \geq 1, \\ 0, & \text{otherwise.} \end{cases}$$

4.3 Applying Dynamic Attention Masks

Case 1: If the input sequence length S satisfies $S \leq L$, $M_{\ell,h}$ will be applied as the attention mask. **Case 2:** If $S > L$, the method constructs an extended mask $\tilde{M}_{\ell,h}$ of size $S \times S$. The first $L \times L$ region remains unchanged:

$$\tilde{M}_{\ell,h}^{(i,j)} = M_{\ell,h}^{(i,j)}, \quad \text{for } i, j \leq L.$$

For $i, j > L$, attention is allowed if the token pair (i, j) is in the stored matched positions $\mathcal{P}_{\ell,h}$:

$$\tilde{M}_{\ell,h}^{(i,j)} = \begin{cases} 1, & \text{if } (i,j) \in \mathcal{P}_{\ell,h}, \\ 0, & \text{otherwise.} \end{cases}$$

The attention mask applies before softmax. The modified attention score matrix is:

$$A'_{\ell,h} = \frac{Q_{\ell,h} K_{\ell,h}^T}{\sqrt{d_k}} \odot \tilde{M}_{\ell,h}.$$

The model sets masked positions $\tilde{M}_{\ell,h}^{(i,j)} = 0$ to $-\infty$ before softmax, ensuring a probability of zero. The final output is: $O'_{\ell,h} = \text{softmax}(A'_{\ell,h}) V_{\ell,h}$.

5 Experiment

5.1 Experiment Setup

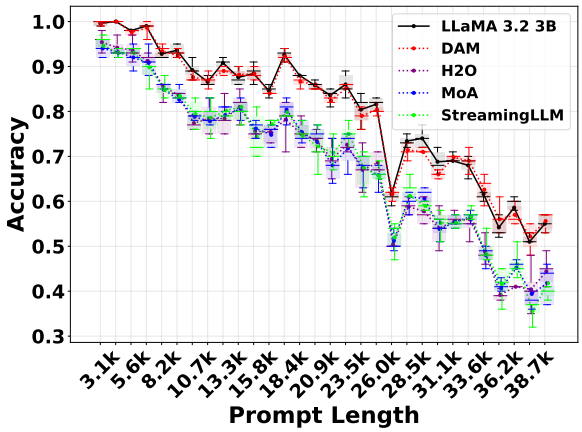

Figure 4: Line-level retrieval accuracy on the LongEval benchmark using the base LLaMA 3.2 3B model. Each sequence contains a predefined target line to be retrieved. Input lengths range from 200 to 3100 lines (3.1k to 38.7k tokens). DAM maintains high retrieval accuracy across all lengths with minimal degradation.

We evaluate DAM on long-context retrieval and QA tasks, comparing against full attention and structured sparsity baselines across multiple sequence lengths and model scales.

Baselines. We compare *DAM* against FlashAttention (Dao, 2023), MoA (Fu et al., 2024), StreamingLLM (Xiao et al., 2024b), and H2O (Zhang et al., 2023). MoA uses predefined sparse attention patterns per layer and head, while StreamingLLM and H2O enhance efficiency during autoregressive decoding.

Base Models. The experiments use LLaMA-3.2-1B-Instruct and LLaMA-3.2-3B-Instruct to analyze scalability across different parameter sizes.

Benchmarks. The evaluation uses LongEval (Krishna et al., 2023) and LV-Eval (Yuan et al., 2024) to assess long-context understanding. LongEval measures key-value retrieval accuracy with 100 data items per sequence length level, offering insights into contextual recall performance.

Hardware. The experiments run on multiple GPU configurations: 4 × A100 (40GB) for LongEval, 2 × H100 (80GB) for LV-Eval, and 1 × A100 (40GB) for efficiency evaluations.

DAM Configuration:

- **Dataset for attention map capture:** MultiNews (Fabbri et al., 2019), a large-scale multi-document dataset that captures diverse attention patterns for general language capability.
- **Pattern Capture Length:** 512, balancing feasibility with attention pattern extraction.
- **Threshold for true masks:** 0.3, determined through attention sparsity analysis.
- **Threshold for approximate masks:** 0.8, ensuring effective structural alignment while minimizing unnecessary attention connections.

5.2 Performance Evaluation

Long-Context Retrieval. The LongEval lines task evaluates retrieval accuracy across different sequence lengths by measuring a model's ability to extract predefined tokens embedded within input sequences ranging from 3K to 104K tokens (illustration ends with base model accuracy smaller than 0.5). Figure 4 shows that DAM maintains an average accuracy of 0.7966, closely matching full attention (0.8011). The accuracy gap remains minimal across all tested lengths, confirming DAM's ability to preserve long-range dependencies. MoA and StreamingLLM experience sharp performance declines beyond 20K tokens, with accuracy dropping to 0.394 and 0.356, respectively. These models fail to capture heterogeneous attention patterns

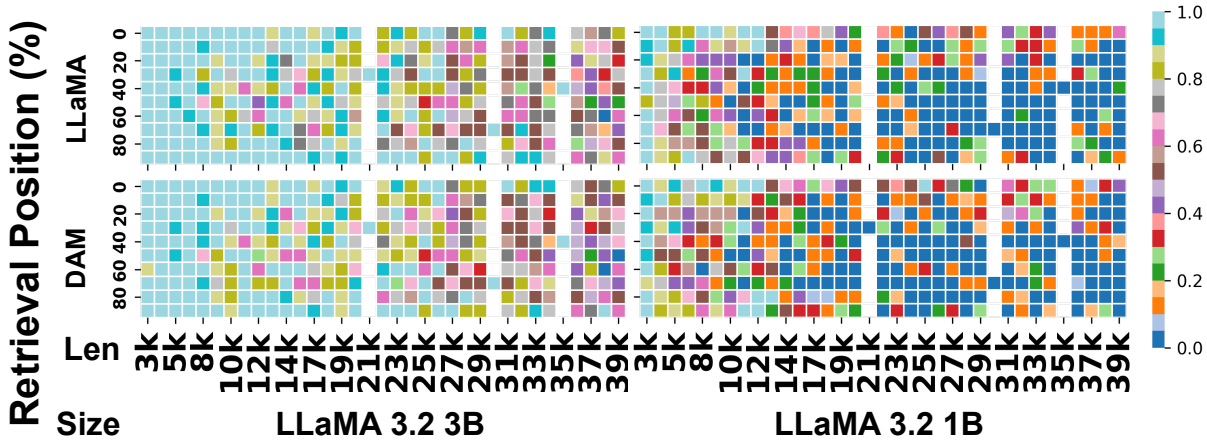

Figure 5: Retrieval accuracy on LongEval for LLaMA 3.2 3B and 1B models. Even at fixed token lengths, performance varies based on the target keyword's position within the sequence. DAM closely matches the dense model's retrieval accuracy across all settings.

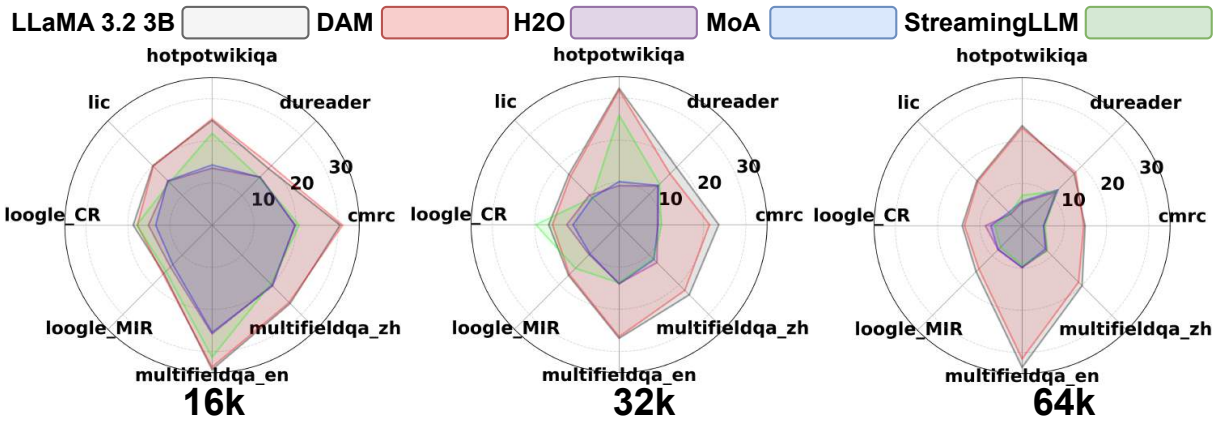

Figure 6: LV-Eval retrieval score across long-context QA tasks. DAM closely matches full attention, achieving 18.61 score at 64K tokens. MoA, StreamingLLM, and H2O lose performance as sequence length increases, with DAM outperforming alternative sparse attention methods.

dynamically, leading to reduced retrieval accuracy.

The retrieval accuracy task evaluates the ability to locate predefined tokens across different sequence lengths. Figure 5 illustrates the performance of DAM compared to the full attention baseline on LLaMA 3.2 1B and 3B models. DAM consistently aligns with the dense model's performance across all evaluated sequence lengths and keyword positions. Notably, even at the same token length, retrieval accuracy varies depending on the keyword's relative position, highlighting the importance of modeling position-sensitive dependencies. DAM preserves such fine-grained retrieval capabilities, demonstrating its effectiveness at retaining long-range and position-sensitive attention patterns without incurring the full computational cost of dense attention. The Appendix C shows the full comparison of other methods.

Long-Context Tasks. The LV-Eval benchmark evaluates retrieval performance in long-context question-answering tasks. This experiment examines sequence lengths from 16K to 256K tokens, with results showing up to 64K tokens. Beyond 128K tokens, base model performance remains stable, while retrieval score declines at 256K tokens. The benchmark includes single-hop and multi-hop QA tasks that require retrieving relevant information from long input contexts. Single-hop QA datasets include cmrc-mixup, multifieldqa-en-mixup, and multifieldqa-zh-mixup. Multi-hop QA datasets include dureader-mixup, loogle-CR-mixup, loogle-MR-mixup, hotpotwikiqa-mixup, and lic-mixup.

Figure 6 shows that DAM closely follows full attention across all datasets. At 64K tokens, DAM reaches an average score of 18.61, compared to 19.29 for full attention. The small gap confirms DAM's ability to retain retrieval scores without quadratic attention costs. MoA and StreamingLLM lose scores as sequence length increases. At 64K tokens, MoA reaches 7.56 and StreamingLLM 7.47, both lower than DAM and full attention.

Mdl	Method	Len	Mem	Tupt	AvgTim
LLaMA 3.2 1B	Original	1k	5.21	1677.63	9766.18
		2k	12.13	1025.84	31942.5
		4k	38.10	928.18	70607
		8k	OOM	—	—
	FlashAttn	1k	3.82	1763.55	9290.38
		2k	5.32	1099.35	29806.8
		4k	8.33	633.456	103458
		8k	10.21	69823.4	1877.19
	DAM	1k	3.84	2574.84	6363.11
		2k	5.35	1656.09	19786.4
		4k	8.35	941.22	69628.8
		8k	10.64	639.653	204911
LLaMA 3.2 3B	Original	1k	9.89	689.20	23772.5
		2k	16.70	403.58	81194.1
		4k	OOM	—	—
		8k	OOM	—	—
	FlashAttn	1k	9.88	710.65	23055.1
		2k	13.76	419.17	78173.8
		4k	21.52	232.15	282298
		8k	21.15	25796.4	5081.03
	DAM	1k	9.90	1095.52	14955.4
		2k	13.79	651.14	50324.3
		4k	21.53	354.96	184631
		8k	31.71	238.08	550541
Vicuna 7B	Original	1k	28.81	451.42	36294.7
		2k	OOM	—	—
	DAM	1k	28.84	738.97	22171.5
		2k	39.14	437.05	74976

Table 1: Comparison of GPU memory (GB), throughput (tokens/s), and average latency (ms) for Original, FlashAttention, and DAM across LLaMA 3.2 (1B, 3B) and Vicuna 7B models at various sequence lengths.

These models fail to retain long-range dependencies, reducing effectiveness in multi-hop retrieval. H2O holds performance better than MoA and StreamingLLM but scores lower than DAM. At 64K tokens, H2O records 7.59, slightly above MoA and StreamingLLM but below DAM.

5.3 Efficiency

We benchmark DAM against the original dense attention implementation and FlashAttention across LLaMA 3.2 1B, 3B, and Vicuna 7B models. Table 1 reports GPU memory usage (GB), average decoding latency (ms), and throughput (tokens/sec) at varying sequence lengths.

DAM consistently maintains a low memory footprint across all models and sequence lengths. For LLaMA 3.2 1B, DAM uses only 10.64 GB at 8K tokens, compared to 38.1 GB for the dense model (which fails beyond 8K due to OOM). Similarly, for LLaMA 3.2 3B and Vicuna 7B, DAM enables 8K inference while full attention is infeasible. DAM's structured sparsity yields memory savings comparable to FlashAttention, and enables longer context handling without modifying model weights or kernel implementations.

DAM achieves favorable throughput–latency trade-offs compared to FlashAttention. For LLaMA 1B at 4K tokens, DAM reaches 941 tokens/sec (vs. 633 for FlashAttention) with 33% lower latency. At 8K, DAM maintains 639 tokens/sec with 204 ms latency, whereas FlashAttention achieves a higher 69K tokens/sec spike but only due to GPU tiling effects. This discrepancy stems from FlashAttention's recompute-based kernel optimization, which exhibits nonlinear scaling when the input length aligns with block sizes (e.g., 8192 = 64×128). In contrast, DAM's performance scales more predictably across lengths, without relying on such alignment.

FlashAttention and DAM operate at different layers of optimization. FlashAttention optimizes kernel-level memory access for dense attention, while DAM applies structured sparsity at the model level by preselecting important attention positions. Although we compute full attention maps initially (for mask extraction), DAM avoids attending to most token pairs during inference. This reduces FLOPs complexity from $\mathcal{O}(L^2)$ to $\mathcal{O}(sL)$, where s is the average number of retained keys per query ($s \ll L$). Importantly, DAM's sparse attention layout is compatible with tile-based GPU execution, allowing further fusion with memory-efficient kernels like FlashAttention in future implementations.

6 Conclusion

We introduce *DAM*, a sparse attention method that dynamically captures heterogeneous token interactions, overcoming the limitations of static and predefined sparsity patterns. *DAM* learns adaptive attention masks that retain crucial dependencies, improving retrieval accuracy while significantly reducing computational cost. Experiments across long-context benchmarks demonstrate DAM's effectiveness in maintaining full-attention performance with lower memory and compute requirements. By bridging the gap between efficiency and expressivity in sparse attention, DAM provides a scalable solution for long-context processing.

7 Limitations

While reducing attention computation at runtime, dynamic sparse masks add preprocessing overhead versus fixed masks. Optimizing mask generation to minimize overhead while maintaining adaptability remains a challenge. Additionally, the approach assumes that structured sparsity in attention patterns can be effectively learned and generalized, but this may not always align with optimal information flow in every task. Future work could explore adaptive learning mechanisms that refine sparsity patterns based on downstream task performance. Though this method scales more efficiently than full attention, handling extremely long sequences, such as multi-million-token documents or continuous streaming inputs, remains a challenge due to memory constraints in mask storage and extension. Exploring hybrid models that integrate retrieval-based or memory-augmented techniques could improve efficiency for such cases.

References

Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. 2024. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. *Proceedings of Machine Learning and Systems*, 6:114–127.

Meta AI. 2024. Llama 3.2 model card.

Joshua Ainslie, Santiago Ontañón, Chris Alberti, Philip Pham, Anirudh Ravula, and Sumit Sanghai. 2004. Etc: encoding long and structured data in transformers. *CoRR, abs*.

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

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

G. E. P. Box and D. R. Cox. 1964. An analysis of transformations. *Journal of the Royal Statistical Society: Series B (Methodological)*, 26(2):211–252.

Yilong Chen, Guoxia Wang, Junyuan Shang, Shiyao Cui, Zhenyu Zhang, Tingwen Liu, Shuohuan Wang, Yu Sun, Dianhai Yu, and Hua Wu. 2024. Nacl: A general and effective kv cache eviction framework for llms at inference time. *arXiv preprint arXiv:2408.03675*.

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

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, David Belanger, Lucy Colwell, et al. 2020. Masked language modeling for proteins via linearly scalable long-context transformers. *arXiv preprint arXiv:2006.03555*.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. *Preprint*, arXiv:1909.00015.

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

Alexander R. Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R. Radev. 2019. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model. *CoRR*, abs/1906.01749.

Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. 2024. Moa: Mixture of sparse attention for automatic large language model compression. *arXiv preprint arXiv:2406.14909*.

Tianyu Fu, Xuefei Ning, Boju Chen, Tianqi Wu, Genghan Zhang, Guohao Dai, Huazhong Yang, and Yu Wang. Semsa: Semantic sparse attention is hidden in large language models.

Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. 2024. {Cost-Efficient} large language model serving for multi-turn conversations with {CachedAttention}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 111–126.

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

Akshay Goindani and Manish Shrivastava. 2021. A dynamic head importance computation mechanism for neural machine translation. *Preprint*, arXiv:2108.01377.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. 2019. Axial attention in multidimensional transformers. *arXiv preprint arXiv:1912.12180*.

Cunchen Hu, Heyang Huang, Junhao Hu, Jiang Xu, Xusheng Chen, Tao Xie, Chenxi Wang, Sa Wang, Yungang Bao, Ninghui Sun, et al. 2024. Memserve: Context caching for disaggregated llm serving with elastic memory pool. *arXiv preprint arXiv:2406.17565*.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*.

Kalpesh Krishna, Erin Bransom, Bailey Kuehl, Mohit Iyyer, Pradeep Dasigi, Arman Cohan, and Kyle Lo. 2023. Longeval: Guidelines for human evaluation of faithfulness in long-form summarization. *arXiv preprint arXiv:2301.13298*.

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

Yichao Liu, Zongru Shao, and Nico Hoffmann. 2021. Global attention mechanism: Retain information to enhance channel-spatial interactions. *Preprint*, arXiv:2112.05561.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2024. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36.

Ramesh Pingili. 2025. Ai-driven intelligent document processing for banking and finance.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Yi Tay, Aston Zhang, Luu Anh Tuan, Jinfeng Rao, Shuai Zhang, Shuohang Wang, Jie Fu, and Siu Cheung Hui. 2019. Lightweight and efficient neural natural language processing with quaternion networks. *arXiv preprint arXiv:1906.04393*.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Dongsheng Wang, Casper Hansen, Lucas Chaves Lima, Christian Hansen, Maria Maistro, Jakob Grue Simonsen, and Christina Lioma. 2020. Multi-head self-attention with role-guided masks. *Preprint*, arXiv:2012.12366.

Da Xiao, Qingye Meng, Shengping Li, and Xingyuan Yuan. 2024a. Improving transformers with dynamically composable multi-head attention. *Preprint*, arXiv:2405.08553.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024b. Efficient streaming language models with attention sinks. *Preprint*, arXiv:2309.17453.

Yi Xiong, Hao Wu, Changxu Shao, Ziqing Wang, Rui Zhang, Yuhong Guo, Junping Zhao, Ke Zhang, and Zhenxuan Pan. 2024. Layerkv: Optimizing large language model serving with layer-wise kv cache management. *arXiv preprint arXiv:2410.00428*.

Lu Ye, Ze Tao, Yong Huang, and Yang Li. 2024. Chunkattention: Efficient self-attention with prefix-aware kv cache and two-phase partition. *arXiv preprint arXiv:2402.15220*.

Tao Yuan, Xuefei Ning, Dong Zhou, Zhijie Yang, Shiyao Li, Minghui Zhuang, Zheyue Tan, Zhuyu Yao, Dahua Lin, Boxun Li, Guohao Dai, Shengen Yan, and Yu Wang. 2024. Lv-eval: A balanced long-context benchmark with 5 length levels up to 256k. *Preprint*, arXiv:2402.05136.

Chulhee Yun, Yin-Wen Chang, Srinadh Bhojanapalli, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. 2020. O (n) connections are expressive enough: Universal approximability of sparse transformers. *Advances in Neural Information Processing Systems*, 33:13783–13794.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Yanqi Zhang, Yuwei Hu, Runyuan Zhao, John Lui, and Haibo Chen. 2024a. Unifying kv cache compression for large language models with leankv. *arXiv preprint arXiv:2412.03131*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. 2023. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710. Curran Associates, Inc.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. 2024b. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36.

Junqi Zhao, Zhijin Fang, Shu Li, Shaohui Yang, and Shichao He. 2024. Buzz: Beehive-structured sparse kv cache with segmented heavy hitters for efficient llm inference. *arXiv preprint arXiv:2410.23079*.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. 2024a. Sglang: Efficient execution of structured language model programs. *arXiv preprint arXiv:2312.07104*.

Zhen Zheng, Xin Ji, Taosong Fang, Fanghao Zhou, Chuanjie Liu, and Gang Peng. 2024b. Batchllm: Optimizing large batched llm inference with global prefix sharing and throughput-oriented token batching. *arXiv preprint arXiv:2412.03594*.

A Attention Pattern Observation

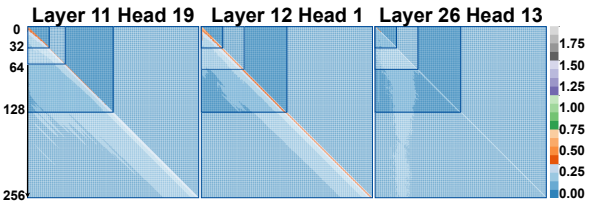

Figure 7: Attention patterns from the LLaMA 3.2 3B model across 32, 64, 128, 256 token lengths. In the left maps, sliding window patterns persist at 64 tokens but disappear by 256. The middle maps show transient tilt patterns from 64 to 128 that do not form regular shapes. Longer token length attention maps extend shorter ones at the same layer and head.

Figure 7 shows how attention patterns evolve with increasing sequence length in Layer 11 Head 19, Layer 12 Head 1, and Layer 26 Head 13 of the LLaMA 3.2 3B model.

Layer 11 Head 19 (left) exhibits a clear sliding window structure at short lengths (32–64 tokens), which gradually fades by 256 tokens. Layer 12 Head 1 (middle) displays irregular diagonal tilts at intermediate lengths (64–128), though these patterns are unstable. In contrast, Layer 26 Head 13 (right) forms consistent, vertically aligned sparse structures as input length increases, suggesting stable token selection in deeper layers.

These patterns motivate the design of *DAM*'s extended attention masks. We extract reliable structural motifs—such as diagonal and vertical bands—from short sequences and use them to extrapolate attention behavior at longer lengths. This avoids recomputing full attention maps while preserving meaningful structure.

DAM computes true attention up to a user-defined *PCL*, captures key patterns from the resulting $L \times L$ mask, and replicates them to extend the mask for longer inputs. The final mask is applied before the softmax to enforce structured sparsity during inference, reducing computation and memory while maintaining attention fidelity.

B Feature Amplification Transformation Methods

Let $A_{\ell,h,i,j}$ denote the accumulated attention weight from layer ℓ, head h, between token positions i and j, and let $C_{\ell,h,i,j}$ be the corresponding valid token count. We compute the average attention as:

$$\bar{A}_{\ell,h,i,j} = \frac{A_{\ell,h,i,j}}{C_{\ell,h,i,j} + \epsilon}$$

where $\epsilon = 10^{-10}$ ensures numerical stability. Let $X = \max(\bar{A}, \epsilon)$ denote the stabilized input. The transformed value is denoted by $\tilde{A}_{\ell,h,i,j}$, and unless otherwise stated, we subtract the global minimum such that $\tilde{A} := \tilde{A} - \min(\tilde{A})$.

The nine transformation methods are defined as follows:

1. Raw Sum: $\tilde{A}_{\ell,h,i,j} = A_{\ell,h,i,j}$

2. Average: $\tilde{A}_{\ell,h,i,j} = \bar{A}_{\ell,h,i,j}$

3. Log: $\tilde{A}_{\ell,h,i,j} = \log(X)$

4. Box-Cox:

$$\tilde{A}_{\ell,h,i,j} = \begin{cases} \frac{X^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(X), & \lambda = 0 \end{cases}$$

5. Yeo-Johnson:

$$\tilde{A}_{\ell,h,i,j} = \begin{cases} \frac{(X+1)^\lambda - 1}{\lambda}, & X \geq 0,\ \lambda \neq 0 \\ \log(X+1), & X \geq 0,\ \lambda = 0 \\ -\frac{(-X+1)^{2-\lambda} - 1}{2-\lambda}, & X < 0,\ \lambda \neq 2 \\ -\log(-X+1), & X < 0,\ \lambda = 2 \end{cases}$$

6. Z-Score: $\tilde{A}_{\ell,h,i,j} = \frac{X - \mu}{\sigma + \epsilon}$
where $\mu = \text{mean}(X)$, $\sigma = \text{std}(X)$

7. Min-Max: $\tilde{A}_{\ell,h,i,j} = \frac{X - \min(X)}{\max(X) - \min(X) + \epsilon}$

8. Square Root: $\tilde{A}_{\ell,h,i,j} = \sqrt{X}$

9. Arcsinh:

$$\tilde{A}_{\ell,h,i,j} = \sinh^{-1}(X) = \log\left(X + \sqrt{X^2 + 1}\right)$$

Figure 8 compares the resulting attention maps across six representative heads. The first two rows—**raw-sum** and **average**—serve as baselines but fail to reveal an informative structure. Raw-sum maps are dominated by large values in early tokens, while average maps mildly reduce saturation but still obscure subtle patterns, particularly in deeper layers (e.g., L25H9, L27H13).

In contrast, **box-cox** and **square-root** transformations enhance interpretability by exposing structural features such as diagonals, stripes, and off-diagonal regions. These patterns are most evident in L3H11 and L15H3, which remain hidden in the baseline maps.

The remaining transformations, including **yeo-johnson**, **z-score**, **min-max**, **arcsinh**, and **log**, either overcompress the range or introduce artifacts, resulting in flattened or noisy maps that hinder downstream use.

Table 2 quantifies the numerical differences between square-root and Box-Cox transformations for Layer 25 Head 9. Although both produce visually informative outputs, Box-Cox maps have bounded and compact ranges (e.g., max $\sim$2.0, mean $\sim$0.27), while square-root maps exhibit large variance and extreme values (e.g., max $\sim$150), mak-

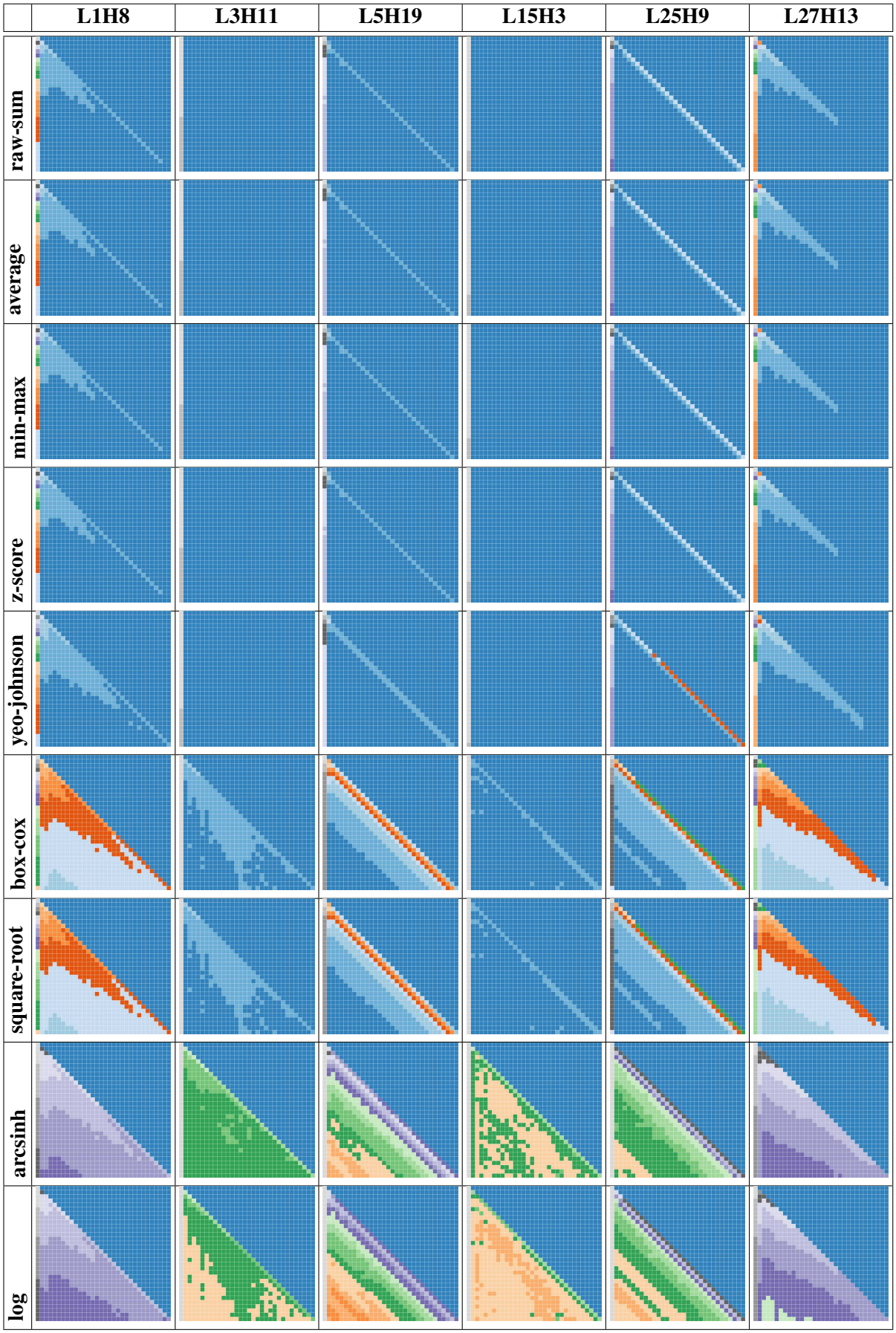

Figure 8: Feature amplification examples across six attention maps from the LLaMA 3.2 3B model under 9 transformation methods. Column headers indicate the layer and head indices; row headers correspond to the transformation methods. Box-Cox and Square Root transformations yield more uniform attention value distributions.

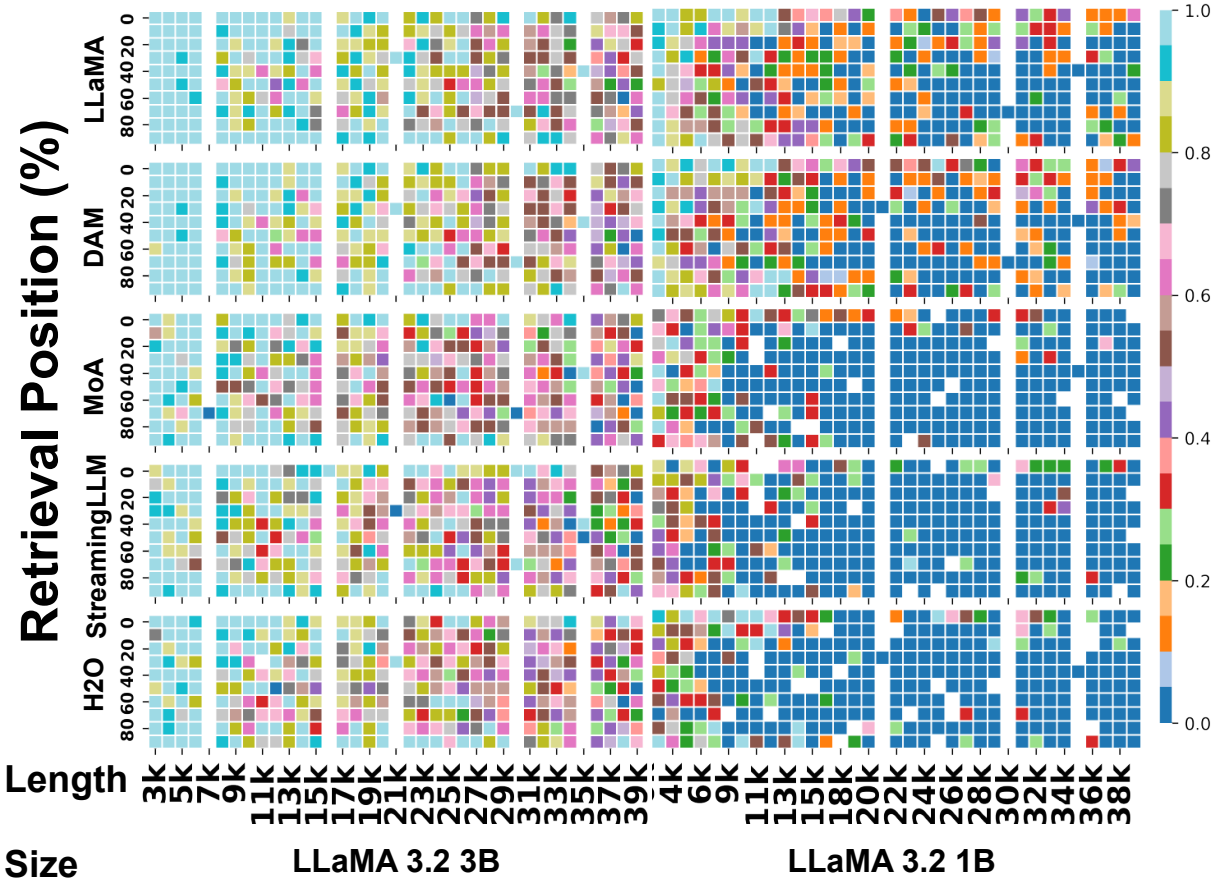

Figure 9: LongEval retrieval accuracy for LLaMA 3.2 3B and 1B models across input lengths to 40K tokens. DAM maintains alignment with the dense LLaMA baseline across retrieval positions and sequence lengths. In contrast, MoA, StreamingLLM, and H2O exhibit early and progressive degradation.

ing thresholding less stable. These results support the use of Box-Cox as the default transformation for attention pattern visualization.

Metric	Square-root	Box-Cox
Max Value	149.95	2.00
Min (non-zero)	4.93	0.07
Mean (non-zero)	13.57	0.27
Std (non-zero)	21.91	0.35
# Non-zero Values	500	500
99th Percentile	∼100	∼1.5

Table 2: Comparison of square-root and Box-Cox transformed attention values for Layer 25 Head 9. Box-Cox yields compact and stable value ranges that are easier to filter or threshold.

C Long-Context Retrieval

We evaluate long-context retrieval using the LongEval benchmark, which measures a model's ability to recover predefined tokens inserted at various positions within input sequences. Figure 9 presents results up to 40K tokens for LLaMA 3.2 3B and 1B models.

For the 3B models, DAM closely tracks the retrieval accuracy of the dense LLaMA baseline across all lengths. While accuracy gradually declines beyond 30K tokens, DAM preserves similar positional trends. In contrast, MoA, StreamingLLM, and H2O begin diverging much earlier, with noticeable color shifts appearing as early as 3K–7K tokens.

For the 1B models, the differences are more pronounced. LLaMA and DAM maintain high accuracy up to 33K tokens, while MoA and StreamingLLM show early degradation starting around 6K. H2O degrades almost immediately across all target positions.

Overall, DAM preserves fine-grained retrieval performance and retains long-range, position-sensitive dependencies without requiring full attention computation, outperforming alternative efficient methods that degrade under longer contexts.