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

Anonymous ACL submission

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 Langue Models (LLMs) without sacrificing retrieval performance.

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 inefficiencies in self-attention.

LLMs leverage the Transformer architecture, which models long-range 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.

Despite these advantages, 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 attention at every step, increasing latency and making real-time processing impractical.

The inability to process long contexts directly impacts businesses and researchers. Legal and financial institutions rely on AI to analyze contracts and reports, but truncated inputs cause critical details to be lost. AI assistants in customer service forget past interactions, failing 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 destroys 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 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 dynami-

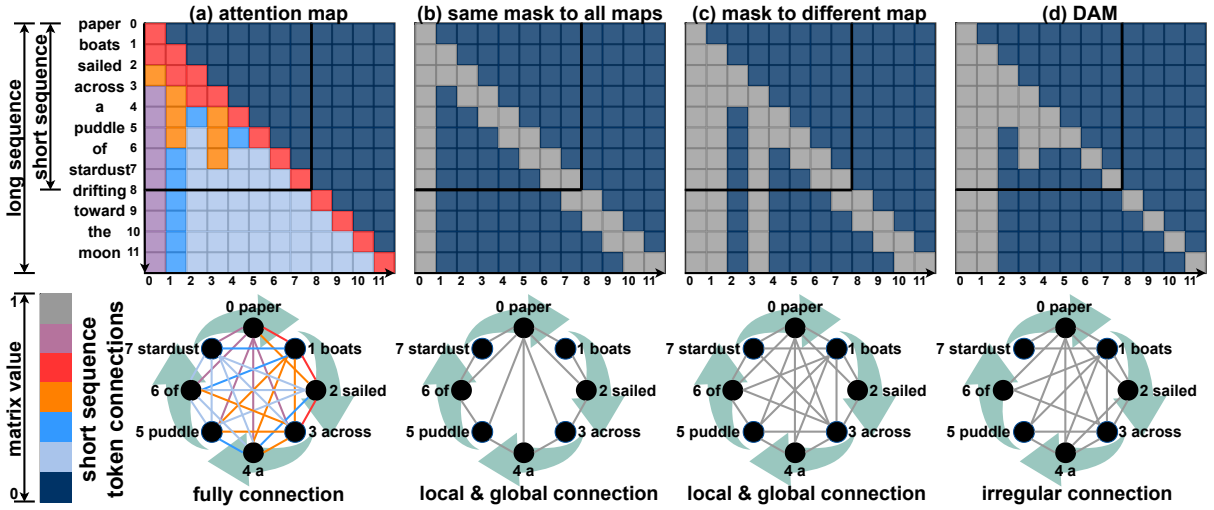

Figure 1: Attention patterns of the short input sequence "paper boats sailed across a puddle of stardust" is a subset of the longer input sequence "paper boats sailed across a puddle of stardust drifting toward the moon. (a) Each query token attends to the full list of others, longer attention patterns are extensions of short patterns. (b) Static attention map captures classical global and sliding-window patterns for general usage to all attention maps. (c) Different masks assign to corresponding maps with predefined patterns. (d) Heterogeneous map remains feature patterns.

cally. 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 novel dynamic sparse attention mechanism and framework that assigns distinct, adaptive 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 significantly 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 ap-

plies 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., 2024b; 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., 2024b; 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.

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.

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

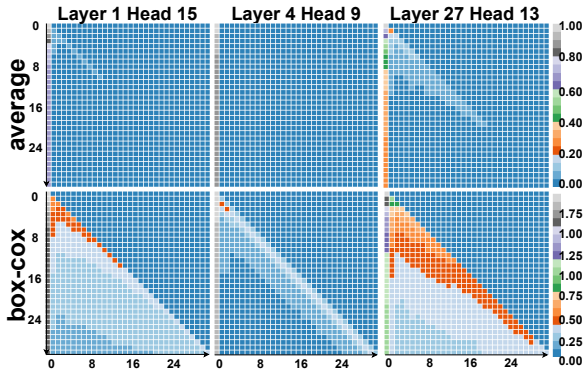

Figure 2: Visualization of dynamic attention patterns across layers and heads. The figure compares the effect of averaging (top row) and applying the Box-Cox transformation (bottom row) to attention values from a LLaMA 3.2 3B Instruct model on the Multi-News dataset. The Box-Cox transformation enhances 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 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 effectively amplifies both small and intermediate attention values without causing the largest values to become disproportionately large. This transformation results in a more uniform distribution of attention values, thereby facilitating the observation and selection of salient attention patterns. This improved visualization motivates the design of *DAM*, which aims to learn and exploit these dynamic patterns.

4.2 Two-Stage Dynamic Attention Masks

DAM is a novel framework designed to enhance 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. 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 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 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

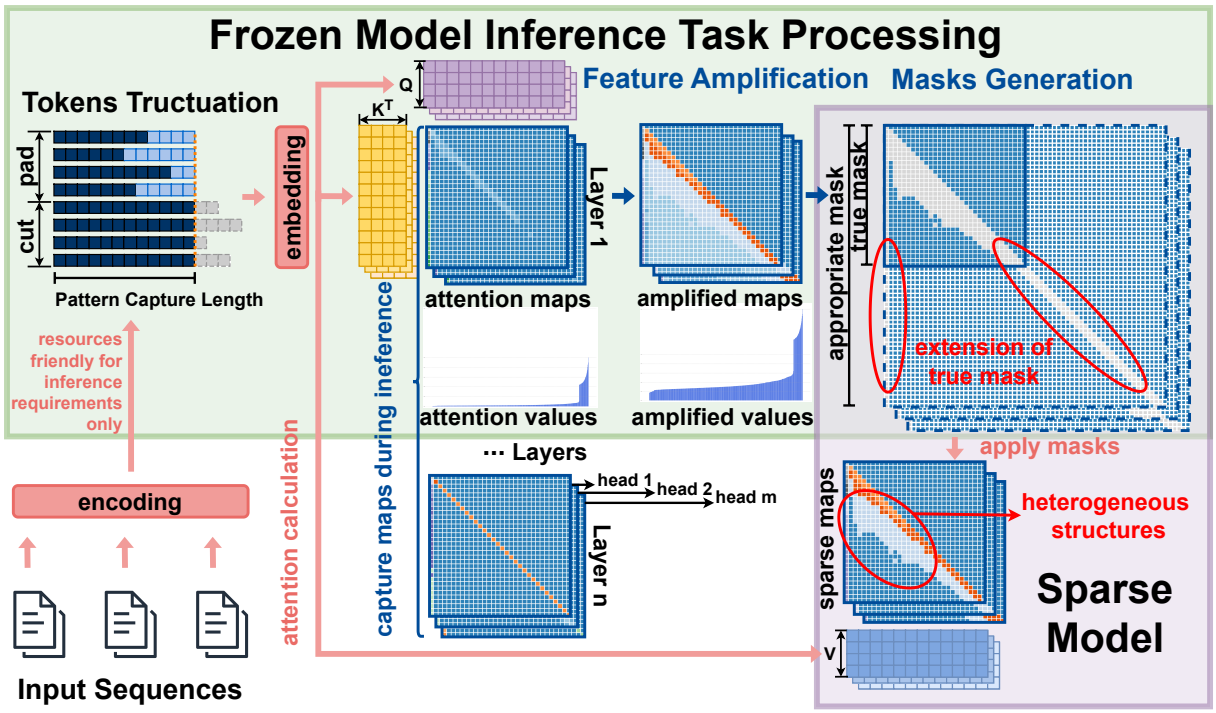

Figure 3: Two-stage *DAM* overview. The first stage extracts full attention patterns from sequences of varying lengths, applies a Box-Cox transformation, and generates masks that capture essential dependencies. The second stage applies these masks to a sparse model, enabling efficient inference while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

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). 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 token pairs within the dataset. Let $A_{\ell,h,i,j}$ denote the accumulated attention value at layer ℓ, head h, from token i to token j, summed across multiple batches. A binary mask $m_{i,j}^{(b)} \in \{0, 1\}$ indicates whether the attention weight for token pair (i, j) was computed in batch b. The count matrix $C_{\ell,h,i,j}$ records the number of times each token 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 this work, we set $\lambda = 0.5$.

To ensure the transformed values $B_{\ell,h,i,j}$ re-

main 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:

$$\gamma_k = \frac{\sum_{i,j} M^{(i,j)}_{\ell,h} \cdot P^{(i,j)}_k}{\sum_{i,j} P^{(i,j)}_k}.$$

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.

The extended mask, $\tilde{M}_{\ell,h}$, is constructed by summing all matched patterns. Because patterns can overlap, summing them effectively combines multiple structural patterns:

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

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

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

4.3 Applying the 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:

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

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

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

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

$$A'_{\ell,h} = \frac{Q_{\ell,h}K^T_{\ell,h}}{\sqrt{d_k}} \odot \tilde{M}_{\ell,h}.$$

The model sets masked positions $\tilde{M}^{(i,j)}_{\ell,h} = 0$ 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

Experiments 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 MoA (Fu et al., 2024a), 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

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 dynamically, leading to reduced retrieval accuracy.

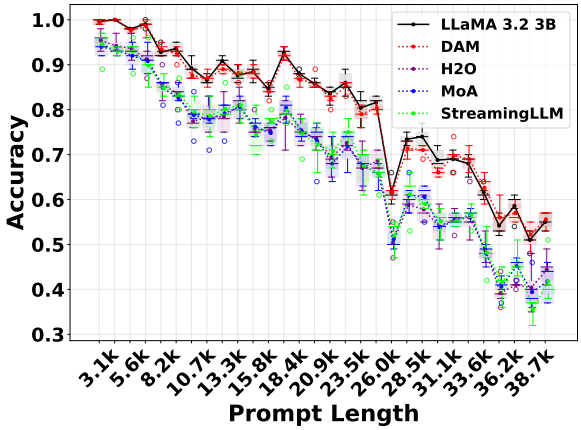

Figure 4: LongEval line accuracy task using the base model LLaMA 3.2 3B. The results depict input sequences ranging from 200 to 3100 lines, corresponding to prompt lengths from 3.1k to 38.7k tokens. DAM exhibits almost no performance degradation.

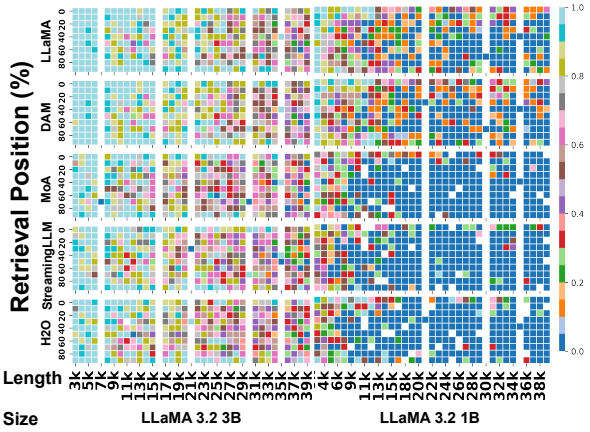

Figure 5: LongEval retrieval accuracy for LLaMA 3.2 3B and 1B models. DAM aligns with dense model performance, while others begin losing retrieval accuracy at early short input lengths.

The retrieval accuracy task evaluates the ability to locate predefined tokens across different sequence lengths. Figure 6 shows that DAM maintains alignment with the full attention model for both 1B and 3B configurations. For the 3B model, other methods show performance degradation at shorter sequences. DAM preserves accuracy across different retrieval positions and lengths, while MoA and StreamingLLM show early declines. At 10K tokens, both models begin to lose retrieval effectiveness, failing to track long-range dependencies.

Long-Context Tasks. The LV-Eval benchmark evaluates retrieval performance in long-context question-answering tasks. This experiment exam-

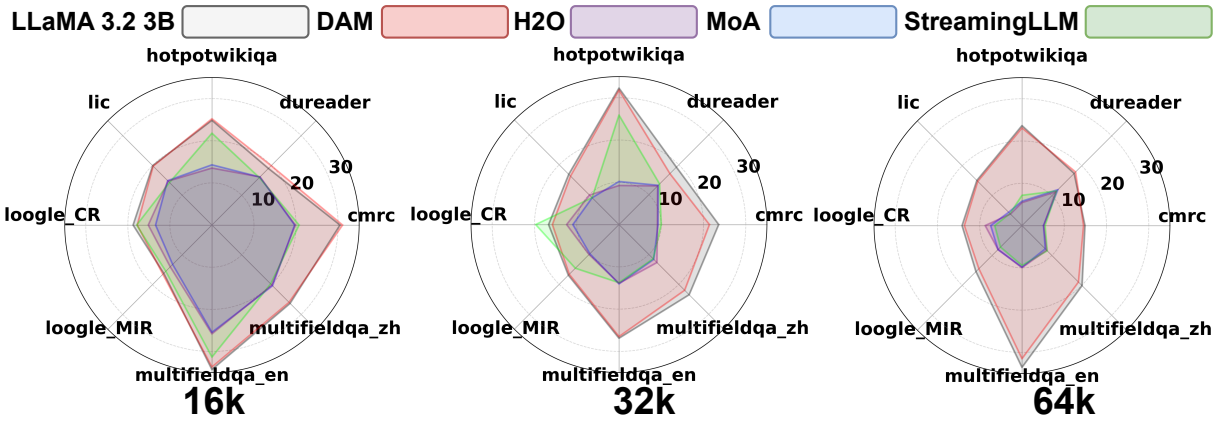

Figure 6: LV-Eval retrieval score across long-context QA tasks. DAM closely matches full attention, achieving 18.61 score at 64K tokens. MoA, StreamingLLM, and H2O lose performance as sequence length increases, with DAM outperforming alternative sparse attention methods.

ines sequence lengths from 16K to 256K tokens, with results shown 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. 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

The efficiency evaluation compares GPU memory usage across different models and sequence lengths, as shown in Table 1. LLaMA-3.2 represents the base model with full attention, while H2O, MoA, and DAM apply sparse attention mechanisms. For the 1B model, LLaMA-3.2 runs out of memory (OOM) at 16K tokens, consuming 28.4GB at 8K tokens. DAM maintains the lowest memory footprint, using 4.0GB at 16K tokens, compared to MoA (5.8GB) and H2O (9.3GB). For the 3B model, LLaMA-3.2 encounters OOM beyond 8K tokens, while DAM processes up to 16K tokens within 18.6GB. MoA and H2O exceed this memory usage before 16K tokens. DAM achieves efficient memory scaling while maintaining retrieval accuracy, demonstrating a practical balance between long-context capability and computational efficiency.

Base Size	Framework	Memory (GB)				
		1k	2k	4k	8k	16k
1B	H2O	3.9	4.3	5.4	6.7	9.3
	MoA	2.8	3.0	3.4	4.2	5.8
	DAM (ours)	**2.5**	**2.6**	**2.8**	**3.2**	**4.0**
	LLaMA-3.2	5.2	8.6	15.2	28.4	OOM
3B	H2O	15.7	20.7	27.3	30.7	OOM
	MoA	14.8	15.6	17.2	20.6	OOM
	DAM (ours)	**12.9**	**13.3**	**14.1**	**15.6**	**18.6**
	LLaMA-3.2	16.9	26.6	32.1	OOM	OOM

Table 1: GPU memory usage comparison for different methods across sequence lengths. DAM maintains the lowest memory footprint, enabling longer sequence processing compared to MoA, H2O, and LLaMA-3.2.

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.

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, Guohao Dai, Huazhong Yang, and Yu Wang. 2024a. Moa: Mixture of sparse attention for automatic large language model compression. *Preprint*, arXiv:2406.14909.

Tianyu Fu, Haofeng Huang, Xuefei Ning, Genghan Zhang, Boju Chen, Tianqi Wu, Hongyi Wang, Zixiao Huang, Shiyao Li, Shengen Yan, et al. 2024b. 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.

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*.