ClusterAttn: KV Cache Compression under Intrinsic Attention Clustering

Minwei Zhang[*], **Haifeng Sun**[*], **Jingyu Wang**[†], **Shaolong Li**, **Wanyi Ning**,
Qi Qi[†], **Zirui Zhuang**, **Jianxin Liao**
State Key Laboratory of Networking and Switching Technology
Beijing University of Posts and Telecommunications
`{mmwzhang,hfsun,wangjingyu,qiqi8266}@bupt.edu.cn`

Abstract

Sparse attention can effectively alleviate the significant demands on memory when large language models (LLMs) process long contexts. Existing methods typically apply the same sparse pattern across different attention heads and inputs. However, this uniform approach fails to capture the inherent diversity of attention patterns within LLMs — the intrinsic attention clustering. To address this, we propose ClusterAttn, a training-free sparse attention method that provides an efficient prompt cache compression scheme under intrinsic attention clustering for efficient LLM inference.

Our findings show that attention heads consistently focus on specific clusters of the prompt during decoding, a pattern detectable from an observation window at the prompt's end. ClusterAttn adaptively fits these clusters utilizing a density-based attention clustering algorithm, thus compressing the KV cache of the prompt. Evaluations on different models across various benchmarks demonstrate ClusterAttn's superior compression rates and efficiency. By utilizing only 1024 tokens, it can reduce memory usage by 10%–65%, resulting in a latency reduction of 12%–23% and a throughput increase of 2.6–4.8 times, all with nearly no accuracy loss. Additionally, ClusterAttn can handle up to 128k context on a single A100-80GB GPU, outperforming existing methods.

1 Introduction

Large Language Models (LLMs) like GPT (Brown et al., 2020; OpenAI, 2023; Ouyang et al., 2022) and LLaMA (Touvron et al., 2023a,b; Meta, 2024) series have significantly advanced natural language processing and artificial general intelligence. These models are trained on large-scale datasets using extensive computational resources (Kaplan et al., 2020) and massive datasets (Anil et al., 2023), enabling them to generate human-like text and perform complex reasoning. However, efficient deployment is challenging due to the high memory demands of KV cache during inference. For instance, a GPT-3 model with 175 billion parameters requires about 1208 GB of GPU memory to store the KV cache when processing 64 sequences of 4096 tokens, which is 3.45 times the memory needed for model weights. To address this challenge, KV cache compression emerges to reduce memory usage, speed up generation, and lower costs, offering substantial commercial benefits.

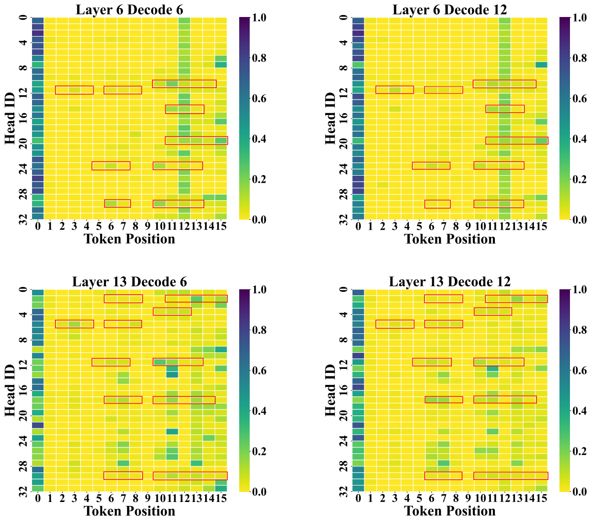

Figure 1: The intrinsic attention clustering phenomenon within the prompt exhibited by Mistral 7B across attention heads during the decode phase.

Previous works proposed sparse attention to improve efficiency but they often lack detailed evaluations for long contexts and primarily focus on optimizing KV cache during decoding. This neglects prompt cache compression, a significant memory bottleneck in real-world applications where inputs are typically larger than responses (OpenAI, 2023; Liu and Mazumder, 2021; Bairi et al., 2024). Ex-

---

[*] Equal Contribution.
[†] Corresponding Author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14451–14473
July 27 - August 1, 2025 ©2025 Association for Computational Linguistics

isting methods also overlook the inherent attention patterns in LLMs that naturally highlight important tokens, a key insight driving our approach — the intrinsic attention clustering.

Our analysis explored the intrinsic attention clustering phenomenon during LLM generation, as illustrated in Figure 1. We observed that, for the same prompt, although the attention distributions vary across attention heads, most heads consistently follow a similar clustering pattern. Specifically, tokens with higher attention scores tend to cluster together, and these clusters remain largely stable throughout the decoding process. This intriguing characteristic suggests that by utilizing these clusters within the prompt, we can extract the critical tokens required for subsequent decoding, enabling effective prompt cache compression while maintaining strong performance, as shown in Table 2.

In this paper, we propose ClusterAttn, a training-free sparse attention method that provides an effective prompt cache compression scheme for efficient LLM inference. As shown in Figure 2, ClusterAttn identifies intrinsic attention clusters in the prompt before decoding, enabling high-compression, high-accuracy prompt cache reduction. It first performs a full prefill, then the process involves three steps: (1) aggregating attention importance from context prefix using an observation window to form the clusters, (2) fitting the clusters with a density-based attention clustering algorithm to compress the context prefix, and (3) concatenating the clusters with the observation window as the final compressed KV cache for subsequent decoding.

Our contributions are summarized as follows:

- We introduce ClusterAttn, a simple yet effective KV cache compression scheme for the prompts. To our knowledge, it's the first method to explore KV compression via intrinsic attention clustering.

- Our experiments reveal the universality of intrinsic attention clustering and its consistency during generation, guiding the compression of the prompt cache.

- Inspired by intrinsic attention clustering, we propose an efficient density-based attention clustering algorithm that fits the clusters for compression. With it, we can profile the optimal fit for the clusters for specific models and datasets within hours.

We conducted extensive experiments on ClusterAttn ,using two large language models, Mistral-7B-Instruct-v0.2 (Jiang et al., 2023) and LWM-text-chat-1m (Liu et al., 2024a). Our evaluation of ClusterAttn demonstrated its ability to significantly compress the KV cache while maintaining accuracy comparable to full attention. With only 1024 tokens, ClusterAttn can reduce memory usage by 10%–65%, resulting in a latency reduction of 12%–23% and a throughput increase of 2.6–4.8 times, with nearly no accuracy loss. Additionally, it can handle up to 128k context on a single A100-80GB GPU, outperforming existing methods.

2 Related Work

Previous works have proposed several methods to reduce the computational and memory costs of attention mechanisms. Some studies use dynamic sparse attention masks to skip computations during prefill stage (Pagliardini et al., 2023; Qu et al., 2022; Roy et al., 2021; Wang et al., 2021; Lu et al., 2021; Kitaev et al., 2020) , while others discard KV cache based on the input sequence during decoding (Anagnostidis et al., 2023; Sheng et al., 2023; Zhang et al., 2023; Ge et al., 2024; Liu et al., 2023). However, dynamic prefill often necessitates specific hardware for effective real-time acceleration (Qu et al., 2022; Wang et al., 2021; Lu et al., 2021; Ham et al., 2021, 2020) , and dynamic KV cache pruning during decoding may require extensive retraining (Anagnostidis et al., 2023) or extra cumulative attention score calculations (Sheng et al., 2023; Zhang et al., 2023; Liu et al., 2023; Ge et al., 2024).

Another research avenue focuses on static sparse attention, where predefined masks are applied consistently across all inputs. This fixed computation flow makes static sparse attention more efficient and better suited for GPUs. In models like BERT (Devlin et al., 2019), various masks have been used (Zaheer et al., 2020; Beltagy et al., 2020; Child et al., 2019; Zhou et al., 2024; Xiao et al., 2024; Han et al., 2024). For generative large language models, fixed-span sliding window masks with global attention on initial tokens are commonly used (Xiao et al., 2024; Han et al., 2024). This local attention pattern allows KV cache outside current span to be discarded, reducing memory usage in long sequences. However, applying static masks uniformly across different attention heads and input lengths may not consider model and data

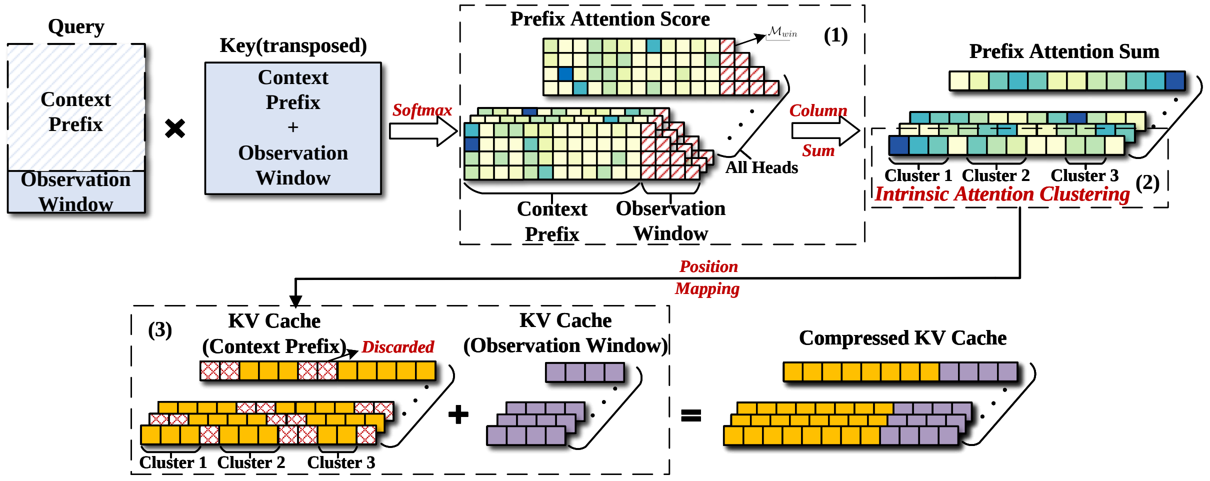

Figure 2: Overview of the ClusterAttn on each layer. After a full prefill, the process involves three steps: (1) aggregating attention importance from context prefix using an observation window to form the clusters, (2) fitting the clusters with a density-based attention clustering algorithm to compress the context prefix, and (3) concatenating the clusters and the observation window to form the KV cache required for subsequent decoding.

characteristics, potentially limiting effective context length and leading to suboptimal performance in long sequences.

Our method falls within dynamic sparse attention, benefiting from the training-free advantages, it can address the performance limitations encountered by previous methods.

3 Observation

In this section, we introduce the intrinsic clustering patterns that emerge within the attention distribution during decoding. We discuss how these patterns can be leveraged for prompt cache compression. Our findings are based on an analysis of various contexts and the behavior of attention mechanisms in LLMs.

3.1 Preliminaries

To structure our experimental analysis, we introduce the following terminologies:

Prompt($prompt$): User-provided input composed of context prefix and observation window.

Context Prefix (pre): It's part of the prompt, which provides contextual information for current conversation or task.

Observation Window (win): It's the final part of the prompt, typically containing the user's query in the task. This window is crucial for analyzing the attention patterns within context prefix.

The definitions above are related as follows ($\mathcal{L}$ denotes the length):

$$\mathcal{L}_{prompt} = \mathcal{L}_{pre} + \mathcal{L}_{win} \tag{1}$$

Observation Window Mask ($\mathcal{M}_{win}$): It's the mask that obscures the attention weights for subsequent softmax function in the observation window.

Feature Aggregation: Within the observation window, we perform a column sum of the attention scores for the context prefix, facilitating the observation of intrinsic attention clusters.

For each sequence, the feature aggregation process can be expressed through Equation 2 to 5:

$$\mathcal{W}_{win} = Q[:, -\mathcal{L}_{win} :]K^T \tag{2}$$

$\mathcal{W}_{win}$ denotes the attention weights between the prompt and observation window across all heads.

$$\mathcal{W}^m_{win} = \text{mask}(\mathcal{W}_{win}[:, :, -\mathcal{L}_{win} :], \mathcal{M}_{win}) \tag{3}$$

$\mathcal{W}^m_{win}$ denotes the attention weights masked by $\mathcal{M}_{win}$ because we only focus on the attention weights of the context prefix.

$$\mathcal{A}^m_{win} = \text{softmax}(\mathcal{W}^m_{win}) \tag{4}$$

$\mathcal{A}^m_{win}$ denotes the attention scores after softmax normalization of $\mathcal{W}^m_{win}$.

$$\mathcal{P} = \sum_{i=-\mathcal{L}_{win}}^{i=\mathcal{L}_{prompt}-1} \mathcal{A}^m_{win}[:, i, : \mathcal{L}_{pre}] \tag{5}$$

$\mathcal{P}$ denotes the aggregated column sum for the context prefix within the observation window, which reflects the intrinsic attention clustering patterns.

Hit Rate: The hit rate quantifies the effectiveness of feature aggregation by measuring the probability that intrinsic attention clusters remain important during subsequent generation. It can be expressed through Equation 6 to 9:

$$\mathcal{I}_{win} = \begin{cases} 1, & \text{if } \mathcal{P} > \theta \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$\mathcal{I}_{cur} = \begin{cases} 1, & \text{if } \mathcal{A}_{cur} > \mu \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

$\mathcal{A}_{cur}$ denotes the attention scores of the context prefix for the current generated token. The thresholding operation filters $\mathcal{P}$ and $\mathcal{A}_{cur}$, retaining positions exceeding θ and μ, indicating significant attention activations. We used $\theta = 5e-4$ and $\mu = 5e-5$ (note that these are relatively large values due to the softmax function over long sequences).

$$\text{overlap} = \mathcal{I}_{win} \wedge \mathcal{I}_{cur} \tag{8}$$

The overlap between these significant activations quantifies the consistency of the clusters.

$$\mathcal{H} = \frac{\sum \text{overlap}}{\sum \mathcal{I}_{cur}} \tag{9}$$

The hit rate $\mathcal{H}$ is then calculated as the ratio of total overlap to total current significant activations, measuring the attention clusters' effectiveness in emphasizing important contextual features.

All our observations were conducted on Mistral-7B-Instruct-v0.2 (Jiang et al., 2023).

3.2 Clustering Universality Across Contexts

We conducted observations on sequences with prompt lengths around 3k from long-document question-answering datasets, including QMSum (Zhong et al., 2021) and OpenReview (An et al., 2024). Before decoding, we used the observation window to aggregate attention features and filtered out positions smaller than θ. We found that for different prompts, intrinsic attention clustering patterns universally appeared in context prefix across attention heads, as shown in Figure 3.

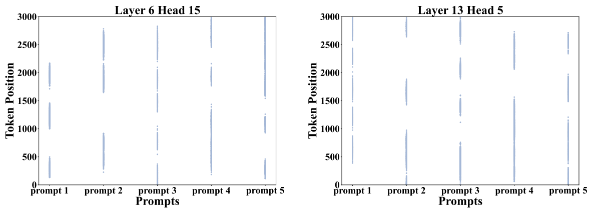

Figure 3: Intrinsic attention clusters in attention heads exhibit universality across different prompts before decoding, where tokens with higher attention scores tend to cluster together.

3.3 Clustering Consistency during Generation

We then explored whether intrinsic attention clusters retain their significance during subsequent generation. Using samples from Ultrachat (Ding et al., 2023) dataset, we filtered sequences with response lengths over 512 and prompt lengths over 1000. We calculated the average hit rate of the intrinsic attention clusters during the generation of 512 tokens. As shown in Figure 4, the clusters maintain consistency throughout the process, evidenced by the high hit rate.

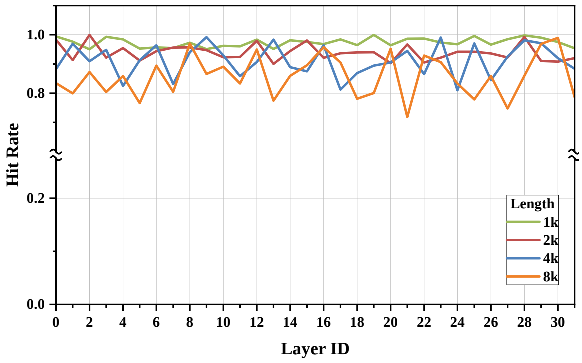

Figure 4: The layer-wise average hit rate of intrinsic attention clusters along token generation with input length varies from 1k to 8k.

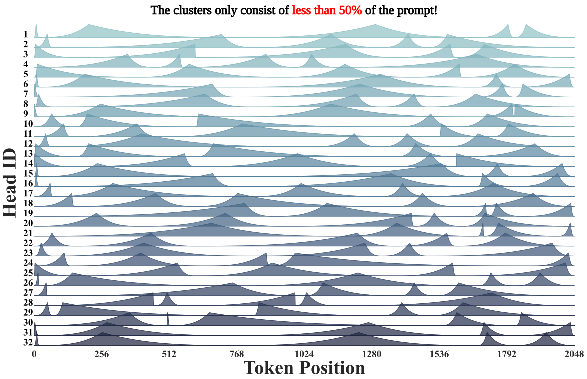

Figure 5: The attention score sum of the context prefix with length 2048 across all heads in a layer during generation of 512 tokens. The ridges indicate that the clusters consistently hold higher attention scores during generation.

To further validate this, we summed the attention scores of the context prefix across all heads during generation. Figure 5 shows that, although the clusters vary across different heads, they exhibit a high degree of consistency within each head. Since these clusters account for a significant proportion of the attention scores, we heuristically assumed that the context prefix can be greatly compressed based on the intrinsic attention clustering patterns and designed an algorithm that leverages these patterns to dynamically compress the prompt cache.

4 Method

In this section, we introduce ClusterAttn, a simple yet effective method for compressing prompt cache leveraging intrinsic attention clustering after a full prefill, as shown in Figure 2. Compared to full attention, it significantly reduces the KV cache by retaining only the most relevant information, decreasing memory usage and latency while maintaining accuracy and improving throughput.

4.1 Aggregating Attention Features from the Context Prefix to Form Clusters

In Section 3.1, based on the observed phenomena, we proposed using the query of every prompt as the observation window to aggregate attention features from the context prefix and form intrinsic attention clusters. This approach ensures context awareness, laying a foundation for subsequent cluster fitting. In this section, for each sample, we use corresponding $\mathcal{P}$ obtained in Equation 5 as the object for cluster fitting.

4.2 Density-Based Attention Clustering Algorithm for Compressing Context Prefix

After feature aggregation, we need to compress the context prefix based on the clustering patterns. Since the intrinsic attention clusters are formed based on attention scores, and unlike traditional clustering of discrete data, these clusters are composed of continuous tokens, we designed a density-based attention clustering algorithm inspired by DBSCAN (Ester et al., 1996) to fit the intrinsic attention clusters, as shown in Algorithm 1.

Algorithm 1 Density-Based Attention Clustering.

Input: K, V, $\mathcal{P}$, num_block, θ, $\mathcal{L}_{pre}$, $\mathcal{L}_{win}$, $\mathcal{L}_{cp}$
Output: Compressed Context Prefix KV Cache

```
1:  for Attention Head $h$ in LLM do
2:      $blksize \leftarrow \frac{\mathcal{L}_{pre}}{num\_block}$                      ▷ Cluster Range
3:      $\mathcal{P}'_h, \mathcal{I} \leftarrow$ maxpool$(\mathcal{P}_h, blksize, blksize)$
4:      $\mathcal{I}_\theta \leftarrow \{i \mid \mathcal{P}'_h[i] \geq \theta\}$                 ▷ $\theta$ Filtering
5:      $r \leftarrow \frac{\mathcal{L}_{cp} - \mathcal{L}_{win}}{2\|\mathcal{I}_\theta\|}$                        ▷ Neighborhood Size
6:      $\mathcal{I}_G \leftarrow$ gather$(\mathcal{I}, \mathcal{I}_\theta) +$ range$(-r, r)$
7:      $\mathcal{I}_U \leftarrow$ unique(clamp$(\mathcal{I}_G, 0, \mathcal{L}_{cp} - \mathcal{L}_{win})$)
8:      $\mathcal{I}_K \leftarrow$ topk$(\mathcal{P}_h^{\overline{\mathcal{I}_U}}, \mathcal{L}_{cp} - \mathcal{L}_{win} - \|\mathcal{I}_U\|)$
9:      $\mathcal{I}_F \leftarrow \mathcal{I}_U \cup \mathcal{I}_K$                  ▷ Clusters' Indices
10:     $\hat{K}_h, \hat{V}_h \leftarrow K_h^{\mathcal{I}_F}, V_h^{\mathcal{I}_F}$              ▷ Compressed KV
11: end for
12: return $\{\hat{K}, \hat{V}\}$
```

Given $\mathcal{L}_{pre}$ as the context prefix size, $\mathcal{L}_{cp}$ as the compression size, and $\mathcal{L}_{win}$ as the observation window size, we define $blksize$ and r in Line 2 and 5 respectively, corresponding to the minimal points(minPts) and neighborhood size used in DBSCAN, where num_block and θ are the hyperparameters to manage the granularity of the clustering. Notably, different from the minimal points required for each cluster in DBSCAN, we use $blksize$ to control the size and range for attention clusters.

According to the algorithm, we first perform max pooling on $\mathcal{P}$ based on $blksize$ to obtain the block-wise maximum indices $\mathcal{I}$ within each cluster range. Then, we filter out indices with scores less than θ to obtain $\mathcal{I}_\theta$. The clustering process is defined as Line 6, where the gather operation obtains the cluster centers and r controls the clustering neighborhood. $\mathcal{P}_h^{\overline{\mathcal{I}_U}}$ denotes the remaining attention after excluding the already selected attention clusters. Notably, the unique and topk operations in Line 7 and 8 are intended to eliminate redundant indices while collecting critical elements between cluster gaps. Finally, we obtain the compressed context prefix KV cache $\{\hat{K}, \hat{V}\}$. We use different θ for threshold filtering depending on the compression size, as shown in Table 1. Thus, the optimal fit for intrinsic attention clusters is primarily determined by num_block, which dictates the granularity of the size and the range of the clusters.

Compression Size	1024	2048	4096
θ(theta)	2e-3	1e-3	8e-4

Table 1: Values of θ under different compression sizes.

To select the most appropriate num_block, we need to profile num_block for each dataset at each compression size. To ensure the consistency and stability of the attention distribution before and after compression, we use the cosine similarity between the attention distribution at the corresponding positions before and after compression during decoding as the metric. Then, we evaluate the optimal fit by measuring the average cosine similarity across all heads for all samples. This is because, although num_block is same for each dataset, the formation of attention clusters for each sample and head is adaptive when performing the algorithm.

The final optimization objective is as follows:

$$\underset{num\_block}{\arg\max}\left(\text{avg}\left[\sum_{\text{sample}}\sum_{\text{head}}\sum_{i=1}^{D}\text{cossim}(\mathcal{A}^{i}_{full},\mathcal{A}^{i}_{\mathcal{I}_F})\right]\right) \quad (10)$$

Here, $\mathcal{A}^{i}_{full}$ and $\mathcal{A}^{i}_{\mathcal{I}_F}$ denote the attention distribution at the corresponding positions before and after compression when generating the ith token. D denotes the total number of generated tokens. For each dataset, we select 20% samples and profile the optimal num_block. It should be noted that there is a trade-off with respect to num_block in our experiments and a detailed analysis is lied in Appendix F. We choose the num_block resulting in the highest average similarity as the optimal clustering fit. By applying the corresponding num_block for each dataset, ClusterAttn is able to make tailored cluster selection under fine granularity, allowing for context-aware KV cache compression.

4.3 Concatenating Clusters with Observation Window as Compressed Prompt Cache

After compressing the context prefix with the cluster patterns, the last step is to combine the clusters with the observation window to obtain the final compressed KV cache for the prompt, like in Figure 2. This is because the clusters represent a compression of the context prefix and do not include user's query within the observation window, which is crucial for the model's responses. Additionally, previous studies (Xiao et al., 2024; Zhang et al., 2023; Sun et al., 2024) have demonstrated that recent tokens are vital for the quality of subsequent generations, as they ensure the model's stability and fluency.

5 Experiments

In this section, we evaluated the performance of ClusterAttn across multiple models and various tasks, comparing it with other state-of-the-art baselines to highlight the effectiveness of our proposed method. Below are some experimental settings:

Baselines We compared ClusterAttn with the latest sparse attention methods: StreamingLLM (Xiao et al., 2024) and H2O (Zhang et al., 2023), which are all training-free. We followed the configurations from their respective papers. For StreamingLLM, the initial four tokens remain unmasked as attention sinks. For H2O, we ensured the same number of key tokens (heavy hitter tokens) and recent tokens. We adopted the compression sizes of 1024, 2048, and 4096 from their papers and applied them to ClusterAttn. Besides, we used FlashAttn (Dao et al., 2022) as the baseline for full attention since ClusterAttn was implemented on it.

Models and Benchmarks We conducted our experiments primarily on two models: LWM-text-chat-1m (Liu et al., 2024a) and Mistral-7B-Instruct-v0.2 (Jiang et al., 2023). To effectively evaluate the models' handling of different sequence lengths and retrieval capability, we conducted the "Needle in a Haystack" (Gkamradt, 2023) experiment on LWM for ClusterAttn. To comprehensively assess the models' long-context understanding abilities, we performed experiments using LongBench (Bai et al., 2024), which includes 16 sub-tasks.

ClusterAttn's Settings Before conducting the experiments, we first selected 20% samples from each dataset required for the experiments. We then performed profiling at compression sizes of 1024, 2048, and 4096 to obtain the optimal num_block for fitting intrinsic attention clusters. During experiments, we directly used the optimal num_block for inference to demonstrate the best performance of our method. Unless otherwise specified, the same num_block was used for each model across each task and length.

Further experiments and ablation studies are shown in Appendix. All our experiments were conducted on a single A100 80GB GPU.

5.1 Evaluation on Intrinsic Attention Clusters Fitting

Before testing the performance of ClusterAttn, we first conducted experiments on the Qasper (Dasigi et al., 2021) dataset to evaluate the fitting performance of our density-based attention clustering algorithm on the attention clusters. We compared it with traditional K-means (MacQueen, 1967) and DBSCAN (Ester et al., 1996). To ensure a fair comparison, for prompts with length 3k, we set the number of clusters for K-means to 8, and set miniPts and neighborhood size for DBSCAN to 128 and 64, respectively. For our algorithm, we set θ, num_block and compression size to 2e-3, 8 and 1024, respectively.

Figure 6 shows the clustering performance of different algorithms on a specific head (left) and the average fitting rate across all heads in the LLM (right). The high fitting rate demonstrates the effectiveness of our clustering algorithm, avoiding the inefficiency of K-means in defining correlation

Model	Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning		
		NtrvQA (F1 Score)	Qasper (F1 Score)	MF-en (F1 Score)	HotpotQA (F1 Score)	2WikiMQA (F1 Score)	Musique (F1 Score)	GovReport (Rouge-L)	QMSum (Rouge-L)	MultiNews (Rouge-L)	TREC (Accuracy)	TriviaQA (F1 Score)	SAMSum (Rouge-L)
LWMChat	Full Attention	**18.20**	**25.56**	40.94	24.57	19.39	10.49	**27.97**	24.9	**24.81**	71.0	60.9	39.73
	StreamingLLM: 4096	11.83	22.45	18.57	16.32	9.15	7	18.54	21.74	23.93	67.5	58.86	37.69
	H2O: 4096	13.18	24.81	20.02	16.86	9.74	7.2	25.74	23.25	23.82	**71.0**	61.05	**40.32**
	ClusterAttn: 1024	18.04	23.72	40.23	24.6	**19.85**	10.76	19.8	24.45	23.54	70.0	**61.44**	39.64
	ClusterAttn: 2048	17.94	25.01	**41.41**	24.5	19.39	**11.33**	21.62	24.21	24.38	70.0	61.14	39.93
	ClusterAttn: 4096	17.93	25.46	40.78	**24.91**	19.52	11.28	25.36	**25.44**	24.61	70.5	61.12	39.65
Mistral	Full Attention	**26.82**	33.06	49.28	**42.77**	27.33	19.27	**32.85**	24.23	27.06	71.0	86.23	42.98
	StreamingLLM: 4096	20.47	27.64	45.55	37.87	20.74	12.47	25.77	20.86	23.93	66.5	79.85	39.63
	H2O: 4096	22.59	29.07	47.17	36.52	20.61	16.26	30.01	23.8	26.74	70.5	86.18	42.98
	ClusterAttn: 1024	25.56	29.5	49.25	40.93	25.74	**19.45**	25.92	23.83	26.13	69.5	**86.53**	42.10
	ClusterAttn: 2048	25.88	32.48	48.6	41.72	27.33	18.69	28.84	**24.5**	26.61	70.0	86.29	42.44
	ClusterAttn: 4096	26.41	**33.37**	**49.81**	42.35	**27.96**	18.79	30.78	24.23	**27.1**	**71.0**	86.26	**43.04**

Table 2: Evaluation of different KV cache compression methods across different LLMs on LongBench.

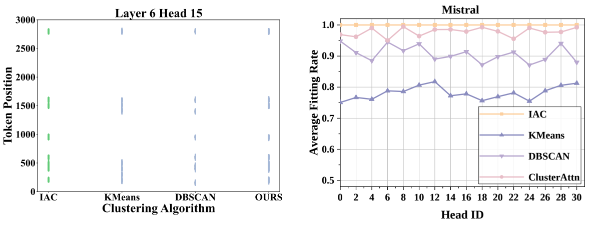

Figure 6: Different algorithms for fitting the intrinsic attention clusters on a specific head (left) and across all heads (right). (IAC denotes the intrinsic attention clusters.)

through mean in long-context scenarios, while being able to capture elements between cluster gaps with greater granularity than DBSCAN.

5.2 Evaluation on Long Context Understanding

We conducted our experiments on LongBench (Bai et al., 2024) for long context understanding ability. For each model, we tested the performance of ClusterAttn at different compression sizes. As shown in Table 2, with only 1024 tokens, the performance drop across 12 different datasets using ClusterAttn is negligible compared to full attention, with some tasks even outperforming it. For LWM model, the average input length is 13,422; for Mistral, it is 13,160. Thus, ClusterAttn achieves an average compression rate of 92% with 1024 tokens, and 68% with 4096 tokens, with negligible accuracy loss. We also compared ClusterAttn with H2O (Zhang et al., 2023) and StreamingLLM (Xiao et al., 2024) on the LongBench datasets to further demonstrate ClusterAttn's performance. To ensure a fair accuracy comparison, we set the compression size of H2O and StreamingLLM to 4096 tokens. As shown in Table 2, ClusterAttn significantly outperforms them. Even with KV cache compressed to 1024 tokens, ClusterAttn on Mistral outperforms H2O with 4096 tokens on 8 out of 12 tasks, and it comprehensively outperforms StreamingLLM.

5.3 Evaluation on Long Context Retrieval

To evaluate the retrieval capability and effective context length of ClusterAttn, we applied it to the LWM model for the "Needle in a Haystack" (Gkamradt, 2023) test. This test requires the model to accurately retrieve a specific sentence ("Needle") hidden in a random position within a long document ("Haystack"). To rigorously assess ClusterAttn's capability, we extended the document length to 128k tokens, which is the maximum sequence length a single A100-80GB GPU can handle when using ClusterAttn with compression size 1024. Results in Figure 7 show that while full attention leads to an out-of-memory error when the sequence length reaches 40k, ClusterAttn can handle a maximum sequence length of 128k with retrieval accuracy still exceeding 90%. This remarkable result highlights ClusterAttn's potential to accurately process details in extremely long input contexts with a 128x compression rate, as well as its immediate effectiveness in saving GPU memory.

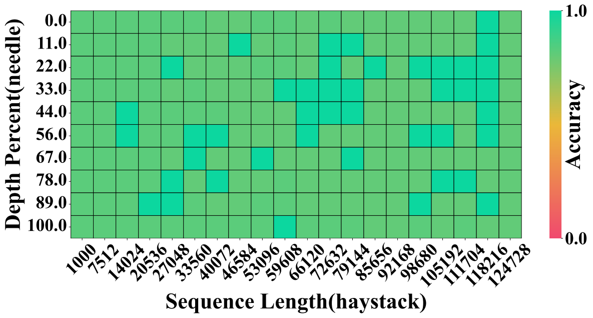

Figure 7: Needle-in-a-Haystack test performance of ClusterAttn on a single A100-80GB GPU. The x-axis denotes the length of the sequence (the "haystack"); the y-axis indicates the position that the "needle" (a short sentence) is located within the sequence, from 1k to 128k tokens. For example, 50% indicates that the needle is placed in the middle of the sequence.

5.4 Coherence Testing

To test the impact of ClusterAttn on the coherence of the original model, we constructed a comprehensive and concise test set by extracting 50 × 4 data items at each length level from the test sets of four long-context understanding datasets: Qasper (Dasigi et al., 2021), MultiNew (Fabbri et al., 2019), TREC (Hovy et al., 2001; Li and Roth, 2002), and LCC (Mohler et al., 2016). These datasets respectively represent the LLM's capabilities in question answering, summarization, few-shot learning, and code completion. According to LongBench (Bai et al., 2024) standards, these data items are organized as question-answer pairs. The questions and answers are human-written and included in the datasets. Perplexity is calculated only on the answer portions of the data to demonstrate the model's coherence in responding to user requests. Table 3 shows the average perplexity of ClusterAttn, H2O, StreamingLLM, and full attention across different inputs and compression sizes. The table reveals that ClusterAttn consistently exhibits lower perplexity, indicating that our method does not negatively impact the coherence of the generated inference results. On the contrary, by removing redundant information, it can even enhance the coherence of the generated outputs.

		Perplexity		
Model	**Method**	**1024**	**2048**	**4096**
Mistral	Full Attention	3.79	**3.79**	3.79
	H2O	3.94	3.90	3.85
	StreamingLLM	4.48	4.27	4.08
	ClusterAttn	**3.75**	3.82	**3.77**
LWM	Full Attention	4.52	4.52	4.52
	H2O	4.63	4.58	4.56
	StreamingLLM	4.79	4.72	4.63
	ClusterAttn	**4.49**	**4.52**	**4.51**

Table 3: Average perplexity for four types of tasks across different models and methods.

5.5 Efficiency

In the following experiments, we set the compression size of ClusterAttn and H2O to 1024 tokens to better prove the performance of ClusterAttn. We didn't compare with StreamingLLM here because it performed poorly in previous experiments thus has limited application in real-world scenarios. We used samples of different lengths in "Needle in a Haystack" (Gkamradt, 2023) as the inputs.

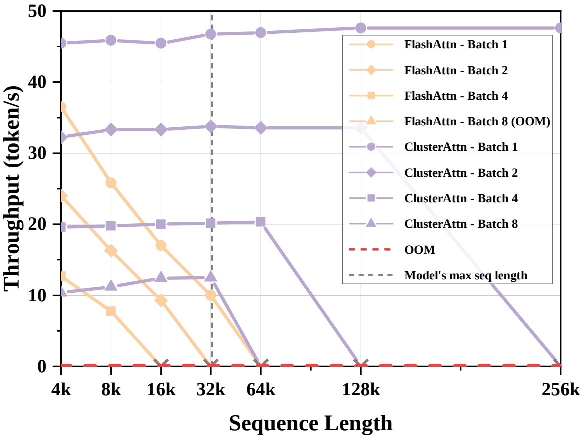

Figure 8: Throughput comparison of FlashAttn (full attention) and ClusterAttn on various batch sizes. Compared to FlashAttn, ClusterAttn improves throughput by 2.6 to 4.8 times.

Throughput Improvement We first tested the throughput of the LWM model using ClusterAttn under different batch sizes. The results shown in Figure 8 lead to two main conclusions. First, as the input length increases, the throughput of the ClusterAttn-optimized model remains constant because the KV cache size is nearly fixed during inference, while FlashAttn (Dao, 2024) (full attention) increases exponentially. Second, under same batch size, ClusterAttn-optimized model can decode significantly longer sequences. This demonstrates the effectiveness of ClusterAttn in minimizing memory consumption and enhancing efficiency.

		Memory (GB)					
Model	**Method**	**4k**	**8k**	**16k**	**32k**	**64k**	**128k**
Mistral	FlashAttn	**16.9**	18.8	22.6	30.2	-	-
	H2O	21.9	33.3	56.7	72.3	-	-
	ClusterAttn	17.1	**18.3**	**21.1**	**26.4**	-	-
LWM	FlashAttn	35.1	40.7	51.8	73.9	OOM	OOM
	H2O	45.7	58.7	75.8	OOM	OOM	OOM
	ClusterAttn	**31.7**	**32.9**	**35.6**	**41.5**	**51.9**	**74.2**

Table 4: Peak memory comparison across different methods and models. With 1024 tokens, ClusterAttn reduces memory usage by 10%–65%.

Peak Memory and Inference Latency We next tested the peak memory usage and inference latency of different methods as sequence length increases, with batch size of 1.

Regarding memory usage (Table 4), for Mistral, the maximum sequence length is set to 32k due to its maximum sequence length limit. H2O reaches its memory limit at 32k, consuming two to three

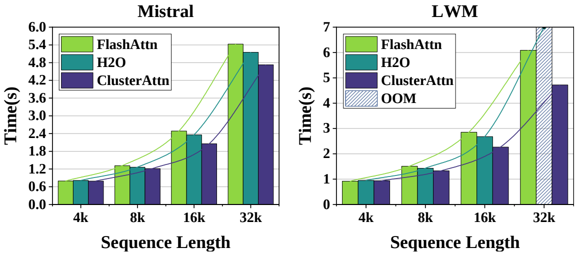

Figure 9: Inference latency comparison across different methods and models. With 1024 tokens, ClusterAttn reduces latency by 12%–23%.

times more memory than FlashAttn and ClusterAttn, because it requires complete prefill and additional overhead for generating dynamic masks. For LWM, with a maximum sequence length of 1 million, H2O encounters OOM issues beyond 16k, FlashAttn faces OOM beyond 32k, while ClusterAttn extends the sequence length to 128k, far exceeding the limits of the other two methods.

In terms of inference latency (Figure 9), as the sequence length increases, the latency reduction achieves by ClusterAttn gradually widens the gap with FlashAttn and H2O. When the sequence length reaches 32k, ClusterAttn provides at least a 20% reduction in latency. These outstanding results in both memory usage and latency highlight ClusterAttn's significant potential to reduce inference costs and improve inference quality.

6 Conclusion

We present a sparse attention method for prompt cache compression based on intrinsic attention clustering – ClusterAttn. It's simple, effective, and training-free, leveraging the attention clustering patterns during the decode process. With only 1024 tokens, it can reduce memory usage by 10%–65%, resulting in a latency reduction of 12%–23% and a throughput increase of 2.6–4.8 times, with nearly no accuracy loss. Furthermore, it can handle up to 128k context on a single A100-80GB GPU, outperforming existing methods. ClusterAttn effectively alleviates the computational and memory burdens of processing large inputs, offering valuable insights and tools for the community to better manage the challenges of large-scale language modeling.

Ethical Considerations

In compliance with ethical considerations, we emphasize that the entirety of our research revolves around open-source datasets, models, and tools. Notably, we exclusively focus on improving model's efficiency and mitigating memory constraints during inference and do not engage in any commercial usage or ethical implications.

Limitations

Despite ClusterAttn's advantages, its research scope is primarily limited to the generative aspects of models. This limitation means that if a model inherently struggles with handling long contexts or exhibits poor performance, ClusterAttn cannot extend the model's long-context capabilities. Additionally, ClusterAttn's design does not address the decoding process, which still relies on standard attention or FlashAttn. This limitation restricts its ability to dynamically update the KV cache during decoding, potentially leading to performance bottlenecks if the output sequence is too long.

Acknowledgments

This work was supported in part by the National Key R&D Program of China 2024YFE0200800, the National Natural Science Foundation of China under Grants (62101064, 62321001, 62471055, U23B2001, 62171057, 62201072, 62071067), the High-Quality Development Project of the MIIT(2440STCZB2584), the Ministry of Education and China Mobile Joint Fund (MCM20200202, MCM20180101), the Fundamental Research Funds for the Central Universities (2024PTB-004) and the 2024 Education and Teaching Reform Project Funding at Beijing University of Posts and Telecommunications (2024YB17).

References

Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. 2023. GQA: training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. 2022. Deepspeed- inference: Enabling efficient inference of transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis, Dallas, TX, USA, November 13-18, 2022*, pages 46:1–46:15. IEEE.

Chenxin An, Shansan Gong, Ming Zhong, Xingjian Zhao, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. 2024. L-eval: Instituting standardized evaluation for long context language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 14388–14411. Association for Computational Linguistics.

Sotiris Anagnostidis, Dario Pavllo, Luca Biggio, Lorenzo Noci, Aurélien Lucchi, and Thomas Hofmann. 2023. Dynamic context pruning for efficient and interpretable autoregressive transformers. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3119–3137. Association for Computational Linguistics.

Ramakrishna Bairi, Atharv Sonwane, Aditya Kanade, Vageesh D. C., Arun Iyer, Suresh Parthasarathy, Sriram K. Rajamani, Balasubramanyan Ashok, and Shashank Shet. 2024. Codeplan: Repository-level coding using llms and planning. *Proc. ACM Softw. Eng.*, 1(FSE):675–698.

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

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, David Benjamin Belanger, Lucy J. Colwell, and Adrian Weller. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Luciano Del Corro, Allie Del Giorno, Sahaj Agarwal, Bin Yu, Ahmed Awadallah, and Subhabrata Mukherjee. 2023. Skipdecode: Autoregressive skip decoding with batching and caching for efficient LLM inference. *CoRR*, abs/2307.02628.

Damai Dai, Chengqi Deng, Chenggang Zhao, R. X. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y. K. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. 2024. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 1280–1297. Association for Computational Linguistics.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of* *the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 4599–4610. Association for Computational Linguistics.

DeepSeek-AI, Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, Hao Zhang, Hanwei Xu, Hao Yang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jin Chen, Jingyang Yuan, Junjie Qiu, Junxiao Song, Kai Dong, Kaige Gao, Kang Guan, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruizhe Pan, Runxin Xu, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Size Zheng, Tao Wang, Tian Pei, Tian Yuan, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaosha Chen, Xiaotao Nie, and Xiaowen Sun. 2024. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *CoRR*, abs/2405.04434.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3.int8(): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3029–3051. Association for Computational Linguistics.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231. AAAI Press.

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. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 1074–1084. Association for Computational Linguistics.

William Fedus, Barret Zoph, and Noam Shazeer. 2022. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.*, 23:120:1–120:39.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. 2024. Model tells you what to discard: Adaptive KV cache compression for llms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Gkamradt. 2023. Llmtest_needleinahaystack: Doing simple retrieval from llm models. [Online; accessed 29-December-2023].

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *CoRR*, abs/1911.12237.

Albert Gu and Tri Dao. 2023. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752.

Tae Jun Ham, Sungjun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. 2020. A^3: Accelerating attention mechanisms in neural networks with approximation. In *IEEE International Symposium on High Performance Computer Architecture, HPCA 2020, San Diego, CA, USA, February 22-26, 2020*, pages 328–341. IEEE.

Tae Jun Ham, Yejin Lee, Seong Hoon Seo, Soosung Kim, Hyunji Choi, Sung Jun Jung, and Jae W. Lee. 2021. ELSA: hardware-software co-design for efficient, lightweight self-attention mechanism in neural networks. In *48th ACM/IEEE Annual International Symposium on Computer Architecture, ISCA 2021, Virtual Event / Valencia, Spain, June 14-18, 2021*, pages 692–705. IEEE.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. 2024. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3991–4008. Association for Computational Linguistics.

Eduard H. Hovy, Laurie Gerber, Ulf Hermjakob, Chin-Yew Lin, and Deepak Ravichandran. 2001. Toward semantics-based answer pinpointing. In *Proceedings of the First International Conference on Human Language Technology Research, HLT 2001, San Diego, California, USA, March 18-21, 2001*. Morgan Kaufmann.

Changho Hwang, Wei Cui, Yifan Xiong, Ziyue Yang, Ze Liu, Han Hu, Zilong Wang, Rafael Salas, Jithin Jose, Prabhat Ram, HoYuen Chau, Peng Cheng, Fan Yang, Mao Yang, and Yongqiang Xiong. 2023. Tutel: Adaptive mixture-of-experts at scale. In *Proceedings of the Sixth Conference on Machine Learning and Systems, MLSys 2023, Miami, FL, USA, June 4-8, 2023*. mlsys.org.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b. *CoRR*, abs/2310.06825.

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. 2023. Polysketchformer: Fast transformers via sketches for polynomial kernels. *CoRR*, abs/2310.01655.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. 2018. The narrativeqa reading comprehension challenge. *Trans. Assoc. Comput. Linguistics*, 6:317–328.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, pages 611–626. ACM.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753. PMLR.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. 2023a. How long can context length of open-source llms truly promise? In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *19th International Conference on Computational Linguistics, COLING 2002, Howard International House and Academia Sinica, Taipei, Taiwan, August 24 - September 1, 2002*.

Yuhong Li, Tianle Cai, Yi Zhang, Deming Chen, and Debadeepta Dey. 2023b. What makes convolutional models great on long sequence modeling? In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

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. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. 2024. AWQ: activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of the Seventh Annual Conference on Machine Learning and Systems, MLSys 2024, Santa Clara, CA, USA, May 13-16, 2024*. mlsys.org.

Bing Liu and Sahisnu Mazumder. 2021. Lifelong and continual learning dialogue systems: Learning during conversation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium* *on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 15058–15063. AAAI Press.

Hao Liu, Wilson Yan, Matei Zaharia, and Pieter Abbeel. 2024a. World model on million-length video and language with blockwise ringattention. *CoRR*, abs/2402.08268.

Jing Liu, Ruihao Gong, Xiuying Wei, Zhiwei Dong, Jianfei Cai, and Bohan Zhuang. 2024b. QLLM: accurate and efficient low-bitwidth quantization for large language models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Liqiang Lu, Yicheng Jin, Hangrui Bi, Zizhang Luo, Peng Li, Tao Wang, and Yun Liang. 2021. Sanger: A co-design framework for enabling sparse attention using reconfigurable architecture. In *MICRO '21: 54th Annual IEEE/ACM International Symposium on Microarchitecture, Virtual Event, Greece, October 18-22, 2021*, pages 977–991. ACM.

Xuezhe Ma, Xiang Kong, Sinong Wang, Chunting Zhou, Jonathan May, Hao Ma, and Luke Zettlemoyer. 2021. Luna: Linear unified nested attention. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 2441–2453.

J. MacQueen. 1967. Some methods for classification and analysis of multivariate observations.

AI Meta. 2024. Introducing meta llama 3: The most capable openly available llm to date. *Meta AI*.

Michael Mohler, Mary Brunson, Bryan Rink, and Marc T. Tomlinson. 2016. Introducing the LCC metaphor datasets. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation LREC 2016, Portorož, Slovenia, May 23-28, 2016*. European Language Resources Association (ELRA).

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Matteo Pagliardini, Daniele Paliotta, Martin Jaggi, and François Fleuret. 2023. Faster causal attention over large sequences through sparse flash attention. *CoRR*, abs/2306.01160.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan S. Wind, Stanislaw Wozniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. RWKV: reinventing rnns for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 14048–14077. Association for Computational Linguistics.

Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah A. Smith, and Lingpeng Kong. 2021. Random feature attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. 2023. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 28043–28078. PMLR.

Zheng Qu, Liu Liu, Fengbin Tu, Zhaodong Chen, Yufei Ding, and Yuan Xie. 2022. DOTA: detect and omit weak attentions for scalable transformer acceleration. In *ASPLOS '22: 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Lausanne, Switzerland, 28 February 2022 - 4 March 2022*, pages 14–26. ACM.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Tran, Yi Tay, and Donald Metzler. 2022. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

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

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single GPU. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 31094–31116. PMLR.

Hanshi Sun, Zhuoming Chen, Xinyu Yang, Yuandong Tian, and Beidi Chen. 2024. Triforce: Lossless acceleration of long sequence generation with hierarchical speculative decoding. *CoRR*, abs/2404.11912.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. 2023. Retentive network: A successor to transformer for large language models. *CoRR*, abs/2307.08621.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023a. Llama: Open and efficient foundation language models. *CoRR*, abs/2302.13971.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023b. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288.

Hanrui Wang, Zhekai Zhang, and Song Han. 2021. Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *IEEE International Symposium on High-Performance Computer Architecture, HPCA 2021, Seoul, South Korea, February 27 - March 3, 2021*, pages 97–110. IEEE.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Haoyi Wu and Kewei Tu. 2024. Layer-condensed KV cache for efficient inference of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 11175–11188. Association for Computational Linguistics.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark W. Barrett, Zhangyang Wang, and Beidi Chen. 2023. H2O: heavy-hitter oracle for efficient generative inference of large language models. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan Awadallah, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, and Dragomir R. Radev. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5905–5921. Association for Computational Linguistics.

Wangchunshu Zhou, Canwen Xu, Tao Ge, Julian J. McAuley, Ke Xu, and Furu Wei. 2020. BERT loses patience: Fast and robust inference with early exit. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. 2024. A survey on efficient inference for large language models. *CoRR*, abs/2404.14294.

A Additional Related Work

Large Language Models (LLMs) are constrained by considerable computational and memory requirements during inference, particularly in resource-constrained environments. To mitigate these challenges, alone with sparse attention, a variety of efficient inference techniques have been developed.

For instance, dynamic inference methods (Zhou et al., 2020; Corro et al., 2023; Schuster et al., 2022; Wu and Tu, 2024), represented by mixture-of-experts (MoE) (Fedus et al., 2022; Lepikhin et al., 2021; Dai et al., 2024; Hwang et al., 2023; DeepSeek-AI et al., 2024), adaptively select specific sub-structures of the model during the inference process based on the input data, significantly improving inference efficiency while keeping model capacity.

Techniques like Multi-Query Attention (Ainslie et al., 2023; Shazeer, 2019) and low-rank attentions (DeepSeek-AI et al., 2024; Wang et al., 2020; Ma et al., 2021; Lee et al., 2019) approximate the functionality of traditional attention mechanisms but with more efficient implementations.

Quantization (Lin et al., 2024; Dettmers et al., 2023, 2022; Liu et al., 2024b) involves converting the model's weights and activations into a low bit-width format, thereby reducing memory footprint and computational intensity.

Alternative mechanisms have also been proposed to replace traditional attention for long-sequence modeling (Gu and Dao, 2023; Peng et al., 2023; Sun et al., 2023; Li et al., 2023b; Kacham et al., 2023; Peng et al., 2021; Choromanski et al., 2021; Wang et al., 2020; Poli et al., 2023). However, these new mechanisms often require weights different from the original transformer, leading to significant retraining costs for large language models.

Previous work has also introduced acceleration frameworks for large language models (Aminabadi et al., 2022; Sheng et al., 2023; Kwon et al., 2023) and kernel-level optimizations (Dao et al., 2022; Dao, 2024). These kernel and system optimizations are orthogonal to our work and can be integrated to further enhance efficiency.

B Extended Comparison with SnapKV

SnapKV (Li et al., 2024) is also a method for KV cache compression that does not require training. It discovered that each attention head in the model consistently focuses on specific prompt features during generation, and these features can also be obtained through an observation window at the end of the prompt. Based on this observation, SnapKV automatically compresses the KV cache by selecting important KV positions for each attention head.

SnapKV is very similar to our findings and solution but differs in terms of observation window and compression algorithm. First, ClusterAttn's observation window uses the user's query at the end of the input prompt as the observation window, whereas SnapKV uses a fixed-length observation window for all prompts. This approach may affect SnapKV's robustness in capturing task-relevant information from the context prefix and its adaptability to different user queries. Second, SnapKV employs a relatively simple pooling method to capture important information from the context prefix, whereas ClusterAttn uses a more fine-grained DBAC(Density-Based Attention Clustering) algorithm. As a result, SnapKV is likely to perform worse after compression compared to ClusterAttn.

To comprehensively compare the performance and accuracy of ClusterAttn and SnapKV, we conducted experiments on the LongBench (Bai et al., 2024) dataset using the Mistral (Jiang et al., 2023) and LWM (Liu et al., 2024a) models. According to the LongBench experimental results in Table 5, ClusterAttn performed slightly better overall than SnapKV, but the difference was not significant. We attribute this to the fact that the average length of the LongBench dataset is still not at an ultra-long level, making it difficult to create a clear performance gap when both methods have strong information extraction capabilities.

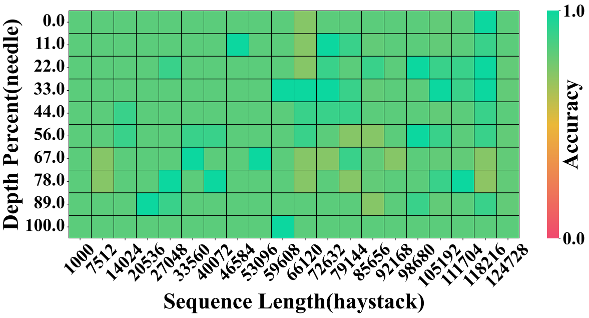

Figure 10: Needle-in-a-Haystack test performance of ClusterAttn on a single A100-80GB GPU. The x-axis denotes the length of the sequence (the "haystack"); the y-axis indicates the position that the "needle" (a short sentence) is located within the sequence, from 1k to 128k tokens. For example, 50% indicates that the needle is placed in the middle of the sequence.

Therefore, we conducted a Needle-in-a-haystack (Gkamradt, 2023) experiment on SnapKV using the LWM model (with a compression size of 1024). The experimental results in Figure 10 indicate that although SnapKV maintains good compression ability as sequence length increases, its retrieval accuracy shows a significant decline compared to ClusterAttn. This demonstrates the robustness of ClusterAttn's performance, which benefits from the use of a more fine-grained compression algorithm.

C Extended Results on LLaMA-3

We also extended ClusterAttn on LLaMA3-8B-Instruct, we compared full attention, SnapKV (Li et al., 2024), and ClusterAttn on the LongBench (Bai et al., 2024) experiment, with the maximum sequence length in the experiment set to match the model's maximum processing length (8192). The experimental results in Table 6 show that ClusterAttn performed as expected, with both accuracy and efficiency consistent with other models.

D Different Sparse Attention on Preserving Contextual Integrity

In large language models (LLMs), information retrieval and generation rely on features with high attention scores, supplemented by using inductive heads to replicate the rest of the context. Therefore, simply selecting top-ranked features may result in retaining only partial details, leading to a loss of information integrity. For instance, this type of compression might cause an LLM to retrieve only the country code of a phone number while fabricating the rest. Our experiments also demonstrate that selecting only the highest-weighted features is insufficient. Such sparse selection can disrupt the contextual integrity of features, reducing accuracy, as shown in Table 7. In contrast, ClusterAttn performs well on all tasks as each cluster represents a continuous segment rather than a sparse selection, preserving the contextual integrity. This approach maintains the stability of attention scores in the compressed KV cache during subsequent decoding, comparable to full attention.

E LongEval

To further evaluate the long context retrieval capability of ClusterAttn, we alse used a modified version of the LongEval-Lines benchmark (Li et al., 2023a), which includes randomly generated pairs and average scores. Compared to the "Needle in a Haystack" (Gkamradt, 2023) test, LongEval-Lines presents a greater challenge as it involves identifying key-value pairs within a noisy context with similar formatting, whereas the relevant information in the "Needle in a Haystack" test is more distinctly separated from other context. We implemented this task on the Mistral model with a compression size of 1024, applying ClusterAttn, H2O (Zhang et al., 2023), and StreamingLLM (Xiao et al., 2024) respectively. Our findings (Figure 11) indicate that as the sequence length increases, ClusterAttn consistently performs on par with full attention, while H2O and StreamingLLM show poorer results, with H2O even encountering an OOM error when the sequence length exceeds 20k. This demonstrates that preserving intrinsic attention clusters enables accurate extraction of critical information in long contexts, significantly improving retrieval accuracy.

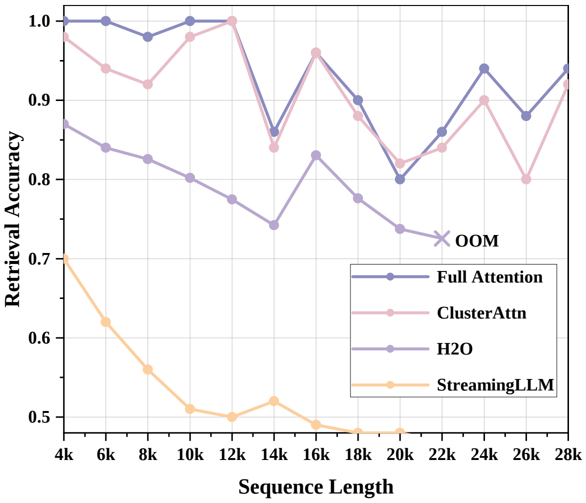

Figure 11: Retrieval accuracy tests on LongEval.

F Ablation Study

In Section 4.2, we mentioned our ClusterAttn algorithm. There, two hyperparameters play a crucial role in the cluster fitting process: θ and num_block. In the following sections, we will analyze the impact of different θ and num_block on the quality of the intrinsic attention cluster fitting on Mistral-7B-Instruct-v0.2 (Jiang et al., 2023).

F.1 The Impact of Theta Selection on Cluster Fitting

We evaluated the selection of θ for various tasks at different compression sizes by measuring the similarity between attention distributions at the corresponding positions before and after compression during decoding. Based on the attention scores in long contexts, we defined the range of θ between

Model	Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning		
		NtrvQA (F1 Score)	Qasper (F1 Score)	MF-en (F1 Score)	HotpotQA (F1 Score)	2WikiMQA (F1 Score)	Musique (F1 Score)	GovReport (Rouge-L)	QMSum (Rouge-L)	MultiNews (Rouge-L)	TREC (Accuracy)	TriviaQA (F1 Score)	SAMSum (Rouge-L)
LWMChat	Full Attention	**18.20**	**25.56**	40.94	24.57	19.39	10.49	**27.97**	24.9	**24.81**	71.0	60.9	39.73
	SnapKV: 1024	18.02	23.72	40.22	24.61	19.84	10.74	19.79	24.44	23.51	70.0	61.41	39.62
	SnapKV: 2048	17.92	24.99	41.38	24.48	19.38	11.33	21.6	24.22	24.36	70.0	61.11	39.9
	SnapKV: 4096	17.90	25.44	40.76	24.90	19.53	11.27	25.34	25.42	24.58	70.5	61.08	39.62
	ClusterAttn: 1024	18.04	23.72	40.23	24.6	**19.85**	10.76	19.8	24.45	23.54	70.0	**61.44**	39.64
	ClusterAttn: 2048	17.94	25.01	**41.41**	24.5	19.39	**11.33**	21.62	24.21	24.38	70.0	61.14	39.93
	ClusterAttn: 4096	17.93	25.46	40.78	**24.91**	19.52	11.28	25.36	**25.44**	24.61	70.5	61.12	39.65
Mistral	Full Attention	**26.82**	33.06	49.28	**42.77**	27.33	19.27	**32.85**	24.23	27.06	71.0	86.23	42.98
	SnapKV: 1024	25.54	29.51	49.23	40.91	25.71	19.42	25.88	23.81	26.1	69.5	86.49	42.08
	SnapKV: 2048	25.87	32.47	48.58	41.7	27.3	18.68	28.8	24.5	26.59	70.0	86.27	42.45
	SnapKV: 4096	26.41	33.36	49.8	42.33	27.94	18.77	30.76	24.2	27.08	71.0	86.23	43.01
	ClusterAttn: 1024	25.56	29.5	49.25	40.93	25.74	**19.45**	25.92	23.83	26.13	69.5	**86.53**	42.10
	ClusterAttn: 2048	25.88	32.48	48.6	41.72	27.33	18.69	28.84	**24.5**	26.61	70.0	86.29	42.44
	ClusterAttn: 4096	26.41	**33.37**	**49.81**	42.35	**27.96**	18.79	30.78	24.23	**27.1**	**71.0**	86.26	**43.04**

Table 5: Evaluation of ClusterAttn and SnapKV across different LLMs on LongBench.

Task	Attention Method	LLaMA-3-8B-Instruct		
		1024	2048	4096
Single QA	Full Attention	**37.33**	**37.33**	**37.33**
	SnapKV	36.38	36.65	36.87
	ClusterAttn	36.94	36.95	36.93
Multi QA	Full Attention	**36.04**	36.04	**36.04**
	SnapKV	35.92	35.98	35.96
	ClusterAttn	36.03	**36.05**	35.99
Sum	Full Attention	**26.83**	**26.83**	**26.83**
	SnapKV	22.58	23.78	25.34
	ClusterAttn	22.54	24.14	25.58
Few-shot	Full Attention	69.56	**69.56**	69.56
	SnapKV	69.34	69.41	69.33
	ClusterAttn	**69.68**	69.55	**69.62**

Table 6: Average performance of ClusterAttn and SnapKV on various tasks from LongBench. The results show that across three different compression lengths, ClusterAttn outperforms SnapKV in most cases and is comparable to full attention.

Task	Attention Method	Mistral-7B-Instruct-v0.2		
		1024	2048	4096
PCount	Full Attention	2.75	2.75	2.75
	H2O	2.89	**3.18**	**3.46**
	StreamingLLM	2.44	2.57	2.62
	TopK	2.73	2.85	2.83
	ClusterAttn	**2.98**	3.09	2.73
PRe	Full Attention	86.98	86.98	**86.98**
	H2O	85.45	87.12	86.38
	StreamingLLM	78.08	82.56	83.77
	TopK	81.18	82.47	81.56
	ClusterAttn	**88.56**	**87.43**	86.18
Lcc	Full Attention	55.51	55.51	55.51
	H2O	54.38	55.47	53.72
	StreamingLLM	50.23	52.24	53.01
	TopK	53.38	53.78	54.37
	ClusterAttn	**55.65**	**55.93**	**55.62**
RB-P	Full Attention	**52.88**	**52.88**	**52.88**
	H2O	50.74	52.24	51.1
	StreamingLLM	47.32	49.57	50.11
	TopK	51.29	51.12	52.45
	ClusterAttn	51.87	52.01	52.65

Table 7: Performance of different sparse attention on various tasks. PCount and PRe represent retrieval-related tasks, while Lcc and RB-P are related to code completion tasks. These four datasets are from LongBench (Bai et al., 2024). The results show that across three different compression lengths, ClusterAttn significantly outperforms the other three baselines in most cases and is comparable to full attention.

1e-4 and 8e-2. θ influences the neighborhood size during the clustering fitting process (such as ϵ-neighborhood in DBSCAN (Ester et al., 1996)), as shown in Figure 13, as θ increases, cosine similarity initially rises gradually to the top but rapidly decreases across compression sizes. This is because larger θ filters out weaker, yet crucial, clusters, leading to poorer fitting quality, while overly small θ will include irrelevant tokens. Notably, for each compression size, we selected the first θ where cosine similarity declines, balancing detailed cluster selection and information preservation.

F.2 The Impact of BlockNum Selection on Cluster Fitting

To evaluate the impact of num_block on fitting quality, we fixed the corresponding θ for each compression size and measured the similarity between attention distributions at the corresponding positions before and after compression during decoding across different tasks (Figure 13). We tested num_block values ranging from 1 to 32, ensuring $blksize$ were appropriate for the compression size.

Results show that cosine similarity peaks at a specific num_block value for each task. This is because that smaller $blksize$ may overfit to single tokens, limiting generalization, while larger $blksize$ may dilute high-score tokens with low-score ones, resulting in reduced differentiation among clusters. Larger $blksize$ also reduce fitting flexibility, constraining diversity within a fixed compression size. With the optimal num_block, ClusterAttn effectively captures crucial attention clusters, adapting to context variability and enabling context-aware

Qasper	**HotpotQA**	**GovReport**	**TREC**
8	8	16	10
NtrvQA	**2WikiMQA**	**QMSum**	**TriviaQA**
8	10	16	10
MF-en	**Musique**	**MultiNews**	**SAMSum**
8	24	18	10
PCount	**PRe**	**Lcc**	**RB-P**
10	12	10	16
LongEval			
12			

Table 8: The *num_block* to obtain the optimal cluster fitting for all experiments on Mistral-7B-Instruct-v0.2.

Qasper	**HotpotQA**	**GovReport**	**TREC**
8	10	18	8
NtrvQA	**2WikiMQA**	**QMSum**	**TriviaQA**
10	10	18	10
MF-en	**Musique**	**MultiNews**	**SAMSum**
8	20	24	8
PCount	**PRe**	**Lcc**	**RB-P**
10	12	10	18
Needle-in-a-Haystack			
10			

Table 9: The *num_block* to obtain the optimal cluster fitting for all experiments on LWM-text-chat-1m.

Qasper	**HotpotQA**	**GovReport**	**TREC**
0.49h,0.5h,0.56h	0.8h,0.79h,0.78h	4.12h,4.13h,4,38h	0.92h,0.92h,0.95h
NtrvQA	**2WikiMQA**	**QMSum**	**TriviaQA**
1.42h,1.44h,1.55h	0.48h,0.51h,0.52h	1.46h,1.55h,2.06h	0.76h,0.77h,0.84h
MF-en	**Musique**	**MultiNews**	**SAMSum**
0.64h,0.68h,0.72h	0.95h,1.03h,1.01h	4.2h,4.48h,4.64h	0.8h,0.81h,0.93h

Table 10: The profiling time for all datasets in LongBench on Mistral-7B-Instruct-v0.2. From left to right, the compression sizes are 1024, 2048, 4096.

Qasper	**HotpotQA**	**GovReport**	**TREC**
0.43h,0.43h,0.45h	0.77h,0.77h,0.79h	4.1h,4.1h,4,4h	0.34h,0.35h,0.38h
NtrvQA	**2WikiMQA**	**QMSum**	**TriviaQA**
1.97h,1.98h,2h	0.45h,0.46h,0.48h	0.97h,1h,1.05h	0.63h,0.65h,0.66h
MF-en	**Musique**	**MultiNews**	**SAMSum**
0.54h,0.54h,0.57h	0.92h,0.92h,0.94h	1.43h,1.51h,1.46h	0.66h,0.68h,0.68h

Table 11: The profiling time for all datasets in LongBench on LWM-text-chat-1m. From left to right, the compression sizes are 1024, 2048, 4096.

compression in the KV cache. Table 8 and Table 9 reveal the best *num_block* to fit the intrinsic attention clusters for all our experiments after profiling. Besides, we also show the profiling time required to obtain the optimal *num_block* for all tasks in LongBench as in Table 10 and Table 11. It can be seen that ClusterAttn only requires a small number of samples and just a few hours to get the optimal *num_block* for each dataset, demonstrating the efficiency and practicality of ClusterAttn.

F.3 ClusterAttn v.s. N-gram-based Attention Clustering

The intrinsic attention clustering is quite straightforward as a word has been tokenized to n-gram tokens. Thus, pivotal information naturally clusters among these n-gram tokens. To better show the effectiveness of our proposed Density-Based Attention Clustering Algorithm, we conducted the comparison between ClusterAttn and a simple n-gram-based Attention Clustering Algorithm here.

To implement the n-gram-based Attention Clustering Algorithm, we first use the observation window for feature aggregation like ClusterAttn. After aggregation, we select important positions based on the prefix attention sum in a descending order and perform n-gram-based attention clustering iteratively until we reach the pre-set compression size. Specifically, we first (1) cluster and retain each position and its surrounding tokens that belong to the same word or phrase. Then, (2) for the current position, retain the n-1 neighboring clusters obtained from step (1) that are adjacent to the already clustered word or phrase, until the total compression length reaches the set compression size.

We still chose to compare the n-gram-based attention clustering algorithm and our density-based attention clustering algorithm on LongBench (Bai et al., 2024) using the LWM (Liu et al., 2024a) model, with the compression size set to 1024. From the results in Table 12, the n-gram-based attention clustering algorithm achieved its best performance at 2-gram (although still showing some gap compared to ClusterAttn). Our analysis is as follows:

Overall, the n-gram-based attention clustering focuses more on clustering based on the physical proximity of words and phrases, rather than understanding the semantics of the context (an important evaluation criterion for the effectiveness of compression algorithms). This results in clusters that are discrete, lacking semantic continuity, which in turn undermines the integrity of contextual information to some extent, and cannot guarantee the stability of the attention distribution in subsequent generation stages.

At the same time, we observed that the overall performance first increased from 1-gram to 2-gram and then decreased. Our understanding of this is that, similar to the effect of *num_block* on the clustering range in ClusterAttn, the value of n in

Task	**Attention Method**	**LWM**
		1024
Single QA	1-gram	25.71
	2-gram	26.83
	3-gram	25.1
	4-gram	23.31
	ClusterAttn	**27.32**
Multi QA	1-gram	16.73
	2-gram	17.97
	3-gram	16.47
	4-gram	16.28
	ClusterAttn	**18.40**
Sum	1-gram	21.23
	2-gram	21.59
	3-gram	21.11
	4-gram	20.72
	ClusterAttn	**22.59**
Few-shot	1-gram	55.77
	2-gram	56.33
	3-gram	55.68
	4-gram	55.17
	ClusterAttn	**57.01**

Table 12: Average performance of ClusterAttn and the n-gram attention clustering algorithm on various tasks from LongBench.

n-gram also determines the size of the clustering range. Smaller values of n may result in clusters that are too dispersed, while larger values may include more redundant information, both of which lead to worse generation results after compression.

G Visualization of the BlockNum on Head-level Cluster Fitting

Additionally, to prove that the obtained optimal *num_block* can achieve a relatively suitable cluster fit for each head, we presented the average cosine similarity for different *num_block* at the same compression size across attention heads. Taking the Qasper (Dasigi et al., 2021) dataset at compression size of 1024 as an example, as is shown in Figure 12, the best fit occurs when *num_block* equals 8. It can be seen that the overall similarity is higher when *num_block* is 8 compared to the others, further highlighting our method's capability to refine and apply context-aware compression to the KV cache for each head.

H Visualization of Text Generation

We provide visualization of generation examples from narrativeqa (Kociský et al., 2018), qasper (Dasigi et al., 2021) and samsum (Gliwa et al., 2019) datasets on Mistral-7B-Instruct-v0.2, as shown in Table 13, Table 14 and Table 15. Results are compared between ground truth, ClusterAttn with compression size 1024, 2048, 4096 and the full attention.

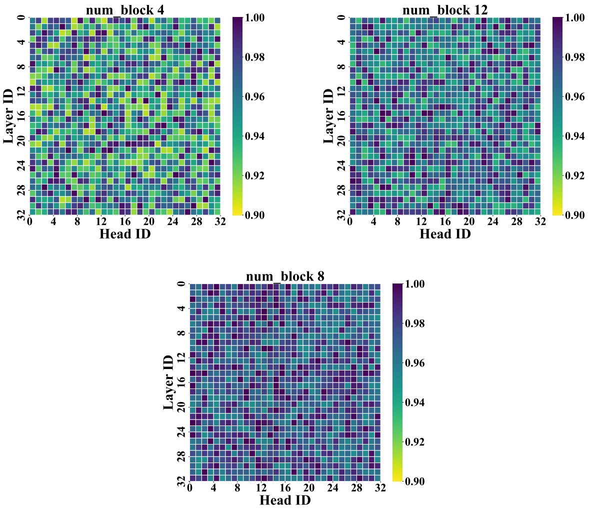

Figure 12: The average cosine similarity across all heads under different *num_block* with compression size 1024.

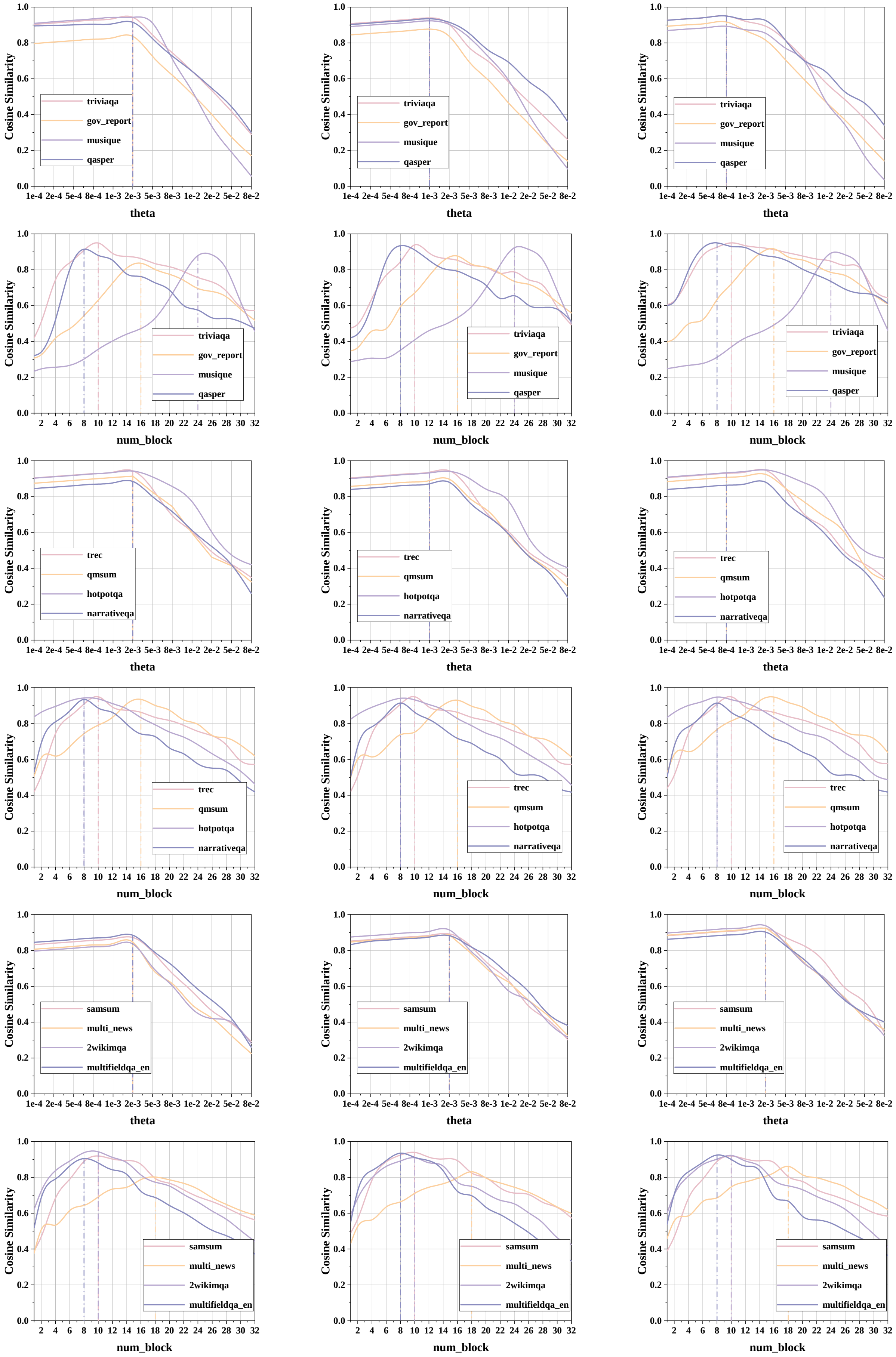

Figure 13: The average cosine similarity of θ and num_block under all compression sizes for LongBench.

NarrativeQA	
Context	You are given a story, which can be either a novel or a movie script, and a question. Answer the question asconcisely as you can , using a single phrase if possible. Do not provide any explanation. **Story:** Transcribed from the 1915 Martin Secker edition by David Price, email ccx074@pglaf.org [Picture: Book cover] THE COXON FUND BY HENRY JAMES [Picture: Decorative graphic] * * * * * LONDON: MARTIN SECKER NUMBER FIVE JOHN STREET ADELPHI * * * * * This edition first published 1915 The text follows that of the Definitive Edition * * * * * I “THEY’VE got him for life!” I said to myself that evening on my way back to the station; ...... Now, answer the question based on the story as concisely as you can, using a single phrase if possible. Do not provide any explanation. **Question:** What is Saltram’s living situation? **Answer:**
Ground Truth	He is a guest in the home of the Mulvilles.
ClusterAttn 1024	Saltram is an inmate at the Mulvilles’ house.
ClusterAttn 2048	Saltram is an inmate and doesn’t dress, but he keeps dinner on at the Mulvilles’.
ClusterAttn 4096	Saltram is an inmate at the Mulvilles’ and doesn’t dress for dinner.
Full Attention	Saltram is an inmate at the Mulvilles’ house.

Table 13: Visualization of generation examples from narrativeqa dataset on Mistral-7B-Instruct-v0.2. Results are compared between ground truth, ClusterAttn with compression size 1024, 2048, 4096 and the full attention.

Qasper	
Context	Answer the question based on the given passages. Only give me the answer and do not output any other words. **The following are given passages.** INTRODUCTION The idea of language identification is to classify a given audio signal into a particular class using a classification algorithm. Commonly language identification task was done using i-vector systems [1]. A very well known approach for language identification proposed by N. Dahek et al. [1] uses the GMM-UBM model to obtain utterance level features called i-vectors. ...... Answer the question based on the given passages. Only give me the answer and do not output any other words. **Question:** What is the GhostVLAD approach? **Answer:**
Ground Truth	extension of the NetVLAD, adds Ghost clusters along with the NetVLAD clusters
ClusterAttn 1024	The GhostVLAD approach is an extension of the NetVLAD approach for face recognition, which adds Ghost clusters to map any noisy or irrelevant content into ghost clusters and are not included during the feature aggregation stage.
ClusterAttn 2048	The GhostVLAD approach is an extension of the NetVLAD approach for face recognition, which adds Ghost clusters to map any noisy or irrelevant content and assigns less weights to them during feature aggregation.
ClusterAttn 4096	The GhostVLAD approach is an extension of the NetVLAD approach for face recognition, which adds Ghost clusters to map any noisy or irrelevant content into ghost clusters and are not included during the feature aggregation stage.
Full Attention	The GhostVLAD approach is an extension of the NetVLAD approach for face recognition, which adds Ghost clusters to map any noisy or irrelevant content into ghost clusters and are not included during the feature aggregation stage.

Table 14: Visualization of generation examples from qasper dataset on Mistral-7B-Instruct-v0.2. Results are compared between ground truth, ClusterAttn with compression size 1024, 2048, 4096 and the full attention.

SAMSum	
Context	Answer the question based on the given passage. Only give me the answer and do not output any other words. **The following are some examples.** Dialogue: Caron: are you getting the last c/s message? Rob: can you? Caron: yes ok Rob: I'm tied up Caron: that was good.. they wanted referrals for people top drove their food bank vans after they have done one of our courses Rob: I dont understand? Caron: they thought we were a training school and had a list of trained people looking for work Rob: oh?? ...... Dialogue: Meg: still at school? Ann: i have extra math classes Meg: so when can i meet you? Ann: i will be home at 7 Summary:
Ground Truth	Ann is still at school. She will be home at 7 so she can meet Meg then.
ClusterAttn 1024	Meg is asking Ann when she can meet her, Ann tells her she will be home at 7.
ClusterAttn 2048	Meg is asking Ann when she can meet her. Ann will be home at 7.
ClusterAttn 4096	Meg is asking Ann when she can meet her. Ann will be home at 7.
Full Attention	Meg is asking Ann when she can meet her. Ann will be home at 7.

Table 15: Visualization of generation examples from samsum dataset on Mistral-7B-Instruct-v0.2. Results are compared between ground truth, ClusterAttn with compression size 1024, 2048, 4096 and the full attention.