DiliLazyKV: Diligent–Lazy Head Effect on Robust KV Cache Compression

Anonymous ACL submission

Abstract

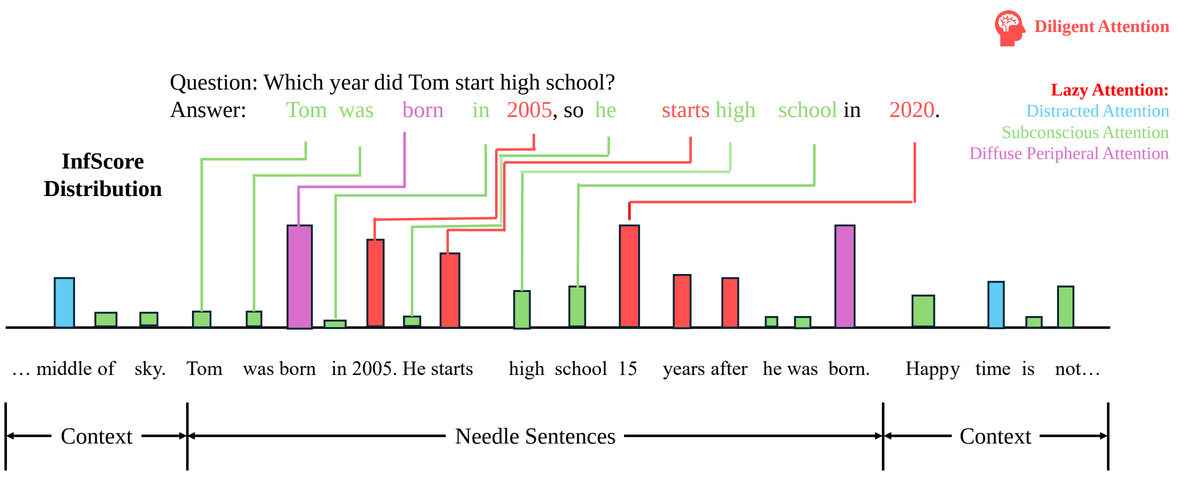
The increasing length of context windows in Large Language Models (LLMs) puts significant pressure on key-value (KV) cache storage, making efficient inference more challenging. Existing compression techniques, which operate at the token, layer, and head levels, often risk discarding valuable information and lack comprehensive adaptability. To overcome these limitations, this paper introduces DiliLazyKV, a novel two-stage approach that utilizes finer-grained functional adaptability based on the proposed Inference Score at the head-layer level. DiliLazyKV achieves greater compression while maintaining better performance across various tasks and longer contexts with β=1.351 in low resources (KV Size =64 & 128), providing a robust KV cache compression strategy. The code is available at `https://github.com/DiliLazyKV/DiliLazyKV`.

1 Introduction

Transformer-based large language models (LLMs) (Vaswani et al., 2017) have demonstrated remarkable performance in language understanding and generation, driving their adoption across a wide range of applications. With the emergence of advanced prompting techniques, the lengths of input context windows have gradually increased. For instance, OpenAI's o3 and o4-mini models (OpenAI, 2025) support up to 128K tokens, Claude 3.7 Sonnet (Anthropic, 2025) uses 200K tokens, and Gemini 2.5 Pro (DeepMind, 2025) can handle up to 1M tokens. However, the increase in context length has created a critical bottleneck in key-value (KV) cache. As KV cache memory usage scales proportionally with sequence length, longer inputs set stronger pressure on GPU memory (Fu, 2024). This constraint limits the maximum sequence length to process, degrades inference speed, and raises the risk of out-of-memory errors, particularly in resource-constrained environments.

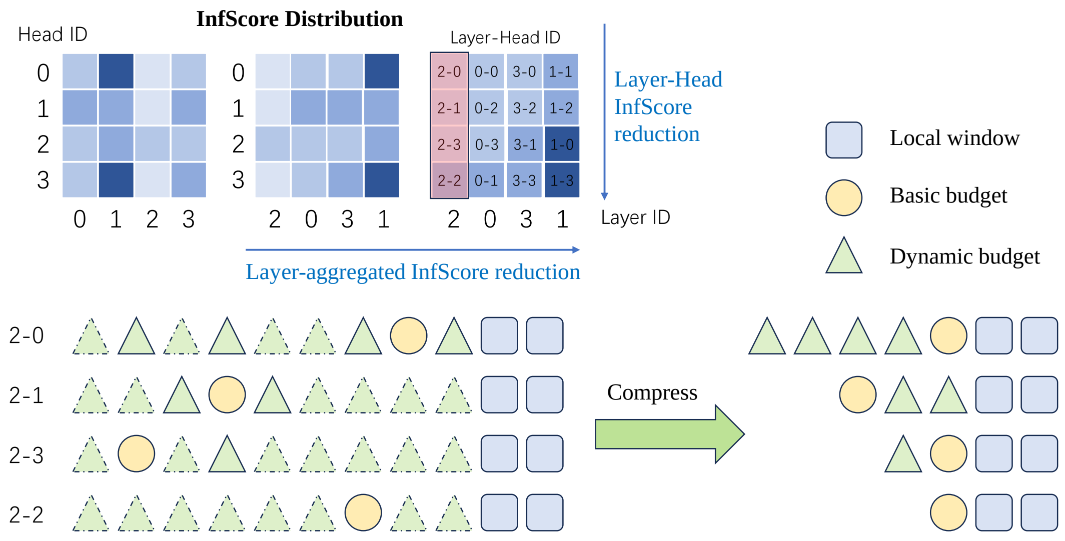
To alleviate the KV cache bottleneck, various compression techniques have been proposed at different granularities. Token-level methods (Li et al., 2024; Zhang et al., 2023; Ribar et al., 2024; Liu et al., 2024, 2023; Wang et al., 2025; Xiao et al., 2024a,c; Jiang et al., 2023) aim to reduce cache size by identifying and discarding the KV pairs of tokens that are deemed unimportant. Layer-level approaches (Zhong et al., 2025; Brandon et al., 2024; Cai et al., 2024; Nawrot et al., 2024) compress the KV cache dynamically across different layers, and head-level schemes (Fu et al., 2024; Feng et al., 2024; Tang et al., 2024; Xiao et al., 2024b) utilize the importance of different attention heads. Although the prior studies have made significant progress in reducing the memory footprint of the KV cache, they pose inherent limitations. They generally rely on a fixed set of importance evaluation criteria, which can result in the aggressive discarding of information that may still be valuable. Globally unimportant tokens, layers, or heads can still have a positive impact on inference in specific local contexts or reasoning tasks. Thus, a single, fixed perspective is insufficient to account for the diversity of inputs. While the overall effectiveness of these methods may be evident, the considerable overhead required to ascertain their validity makes them impractical for LLM deployment. To overcome these limitations, this paper aims to address the following key questions:

- How to define the validity of KV-cached history and enhance traditional importance scores?
- How to design a finer-grained, adaptive mechanism to identify the information defined, and how to leverage more comprehensive context (beyond just token, layer, or head level) to guide KV cache compression?
- How to devise a robust compression strategy that remains effective under real-world deployment constraints?



Diligent Attention
Lazy Attention:
Distracted Attention
Subconscious Attention
Diffuse Peripheral Attention
Question: Which year did Tom start high school?
Answer: Tom was born in 2005, so he starts high school in 2020.
InfScore Distribution
… middle of sky. Tom was born in 2005. He starts high school 15 years after he was born. Happy time is not…
Context
Needle Sentences
Context

(a) Diligent Head and Lazy Head Identification



(b) Collaborative Layer-Head KV Cache Budget Allocation

Figure 1: Our proposed DiliLazyKV method comprises two steps: (1) Diligent Head and Lazy Head Identification (Part a). We employ a Needle-in-a-Haystack (Kamradt, 2023) test to identify heads crucial for different capabilities. DiliScore and LazyScore jointly constitute the Inference Score (InfScore). (2) Collaborative Layer-Head KV Cache Budget Allocation (Part b). During the prefill stage, we allocate the KV cache budget for each head based on the inter-layer aggregation and intra-layer distribution of InfScore variance across heads.

This paper examines the Diligent-Lazy head effect on KV cache compression, referred to as *DiliLazyKV*. In response to above questions, we outline our contributions as follows.

- **Greater fine-grained adaptability to define the validity of information** As shown in Figure 1 (a), our approach identifies different functional heads. *Diligent heads* capture task-specific information, while *lazy heads* capture long-range dependencies via their distracted, subconscious, and diffuse peripheral attention.
- **Improved credibility with the layer-head level** In Figure 1(b), three Inference Score matrices show the process of layer-level distribution from the head level. The low score of early layer 1 highlights the limitations of rigid early, middle, and late segmentation, suggesting the need for a more nuanced layer-head influence.
- **Enhanced robustness for deployment** Experiments on benchmark using Llama-3-8B-Instruct model (Grattafiori et al., 2024) consistently demonstrate that our DiliLazyKV method generally delivers more robust performance than prior works and even outperforms FullKV on

long-context datasets.

2 Related Work

2.1 Attention Mechanism

Multi-Head Attention (MHA) is a foundational component of the Transformer architecture (Vaswani et al., 2017). Multi-query attention (MQA)(Shazeer, 2019) allows multiple attention heads to share a single set of key-value pairs. This mechanism has been embraced by models such as PaLM (Chowdhery et al., 2023), StarCoder (Li et al., 2023b), and Gemini (Google, 2023). As a middle ground, Grouped-Query Attention (GQA)(Ainslie et al., 2023) aims to balance efficiency and expressiveness. It is integrated into models such as LLAMA2-70B (Touvron et al., 2023), the entire LLAMA3 series (Grattafiori et al., 2024), and ChatGLM (GLM, 2024). FlashAttention (Dao, 2024) optimizes the process of attention matrix calculation, improving computational efficiency.

In comparison, DiliLazyKV introduces a mechanism for adapting KV cache utilization, leading to enhanced reasoning capabilities in LLMs.

2.2 KV Cache Compression

KV cache compression methods can be categorized into three granularities: token-level, layer-level, and head-level approaches. The following is an overview of representative methods at each level.

Token Level SnapKV (Li et al., 2024), H_2O (Zhang et al., 2023), SparQ Attention (Ribar et al., 2024), IntactKV (Liu et al., 2024), and Scissorhands (Liu et al., 2023) focus primarily on identifying and discarding unimportant tokens. Similarly, Self-Attention Guided Eviction (SAGE-KV) (Wang et al., 2025) conducts a one-time top-k selection at the token and head levels. Other methods aim to manage long-range context to efficiently access non-recent information, including Infllm (Xiao et al., 2024a), StreamingLLM (Xiao et al., 2024c), and LLMLingua (Jiang et al., 2023).

Layer Level ZigZagKV (Zhong et al., 2025) evaluates the uncertainty of each layer. Cross-Layer Attention (CLA) (Brandon et al., 2024) improves information coordination between different layers by sharing key-value headers between adjacent layers. Additionally, PyramidKV (Cai et al., 2024) and Dynamic Multi-Compression (DMC) (Nawrot et al., 2024) employ different cache allocation strategies for upper and lower layers.

Head Level Leveraging attention head importance has become a key strategy for KV cache optimization. Based on retrieval heads (Wu et al., 2024), HeadKV-R2 (Fu et al., 2024) evaluates the importance of each attention head by integrating text retrieval and reasoning capabilities. In contrast, Ada-KV (Feng et al., 2024) is constrained by a fixed layer budget. RazorAttention (Tang et al., 2024) maintains a complete cache for key retrieval heads and discards remote tags in non-retrieval heads. DuoAttention (Xiao et al., 2024b) categorizes attention heads into retrieval heads and streaming heads. While retrieval heads examine all relevant information in the sequence, streaming heads focus primarily on the most recent tokens, allowing for the use of smaller and even compressed KV caches.

While these approaches reduce KV cache usage and boost inference performance, their single-perspective design fails to leverage complementary insights from other levels. DiliLazyKV addresses this through a two-stage process. First, it computes head-level diligent and lazy scores and combines them into a single inference score that captures each head's global and local attention. Second, inspired by trends in the layer-level distribution, DiliLazyKV aggregates these inference scores across the heads in each layer to guide KV cache allocation. The approach ensures that both global importance and local task-relevant dependencies are maintained.

3 Diligent-Lazy Head Effect

This section introduces the diligent-lazy head effect on KV cache compression. We define diligent and lazy head (3.1) and propose the Inference Attention Score (InfScore) (3.2) indicator to integrate their characteristics. Then We explore Diligengt-Lazy head effect to compress KV cache (3.3).

3.1 Definition of Diligent and Lazy Heads

In attention mechanism, both the diligent head and lazy head are indispensable. With Needle-in-a-Haystack (Kamradt, 2023) test, diligent heads guarantee excellent reasoning ability for target tokens, lazy heads ensure the capacity to capture long-range contextual relationships.

Diligent Head An attention head is regarded diligent when it focuses on a high-scoring token in the needle that belongs to the correct answer. Unlike simply counting in HeadKV-R2 (Fu et al.,

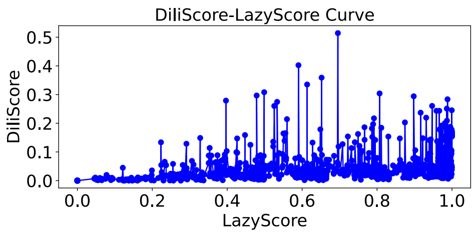


Figure 2: LazyScore-DiliScore Curve exhibits four key characteristics: (1) Data points are widely distributed, indicating that different attention heads have different combinations of diligent and lazy patterns. (2) Few data points with low DiliScore at low LazyScore pay less attention to global context information and task-specific information. (3) Data points that combine a high LazyScore with a broad DiliScore indicate a stronger focus on global contextual information at the expense of task-specific details, resulting in a conclusion that global vision is the basis for task focus, but not a sufficient condition. (4) Few data points with high DiliScore at high LazyScore balance task relevance and global context understanding.

2024), we use the attention score value to prevent it from hitting only once and then never paying attention to the needle again. *Diligent Attention Score (DiliScore)* is defined as the number of needle tokens that are successfully ranked among the model's top-k predictions, formally expressed as DiliScore $= \left|\mathcal{T} \cap \mathcal{N}\right|$ where $\mathcal{T} \subseteq \{1, 2, \ldots, C\}$ represents the top-k token set reasoned by the attention head, $\mathcal{N} \subseteq \{1, 2, \ldots, C\}$ represents the ground-truth needle token set, C is the context length, and $|\mathcal{N}| = N$ is the total number of positive samples.

Lazy Head Lazy attention head tends to alternate between attending to needle tokens and overlooking them. They seem to wander indiscriminately, but in reality, they perform systematic and comprehensive exploration of the extensive haystack context with no bias. The global attention allows them to capture subtle yet potentially important patterns, enhancing robustness in semantic understanding and long-context inference. *Lazy Attention Score (LazyScore)* quantifies the behavior performed when the model's attention fails to effectively focus on the target token in needle. The indicator can be divided into three attentions. Distracted Attention refers to the model's mistaken focus on non-needle tokens within the top-k predictions. Subconscious Attention describes the case where the model fails to include tokens that actually belong to $\mathcal{N}$ in the top-k but still assigns them non-negligible attention weights. Diffuse Peripheral Attention refers to the model's low-intensity attention distribution over non-needle areas outside the top-k range. We define the distraction term as Dist $= |\mathcal{T}| - |\mathcal{T} \cap \mathcal{N}|$, the subconscious term as Sco $= |\mathcal{N} \setminus \mathcal{T}|$, and the diffuse term as Dffu $= C - |\mathcal{N}| - (|\mathcal{T}| - |\mathcal{T} \cap \mathcal{N}|)$. As a result, LazyScore is formulated as the weighted sum of the three components.

Diligent head tends to focus on task-relevant information and is assigned a high DiliScore. Lazy heads, on the other hand, focus on global information and are assigned a high LazyScore. By combining these two types of heads, DiliLazyKV enables a more granular allocation of the KV cache budget, maintaining core reasoning capabilities while minimizing the impact on overall performance. Figure 2 shows the distribution of DiliScore and LazyScore experiments on Meta-Llama-3-8B-Instruct model. It is no obvious linear or simple nonlinear trend indicating a strong positive or negative correlation.

Algorithm 1 DiliLazyKV Algorithm

```
 1: Input:
      Total budget $b_c$
      Allocation ratio $\beta$
      Number of layers in a model $L$
      Number of heads in one layer $H$
      InfScore of $j_{\text{th}}$ head$(i,j)$ in the $i_{\text{th}}$ layer $H_{j,\text{inf}}$
 2: Output: Capacity of head$(i,j)$ C$i,j$
 3: Fixed budget $b_{\text{fixed}} = b_c\left(1 - \frac{1}{\beta}\right)$
 4: Global dynamic budget pool $B_{\text{total}} = \frac{b_c}{\beta} L\, H$
 5: for $i = 0$ to $L - 1$ do
 6:     Initialize layer-level InfScore $L_{i,\alpha} \leftarrow 0$
 7:     for $j = 0$ to $H - 1$ do
 8:         $L_{i,\alpha} \leftarrow L_{i,\alpha} + H_{j,\text{inf}}$
 9:     end for
10: end for
11: Initialize all layer-levle InfScore $L_s \leftarrow 0$
12: for $i = 0$ to $L - 1$ do
13:     $L_s \leftarrow L_s + L_{i,\alpha}$
14:     Compute relative importance $S_{j,h} = \frac{H_{j,\text{inf}}}{L_{i,\alpha}}$
15: end for
16: Compute relative importance of the $i_{\text{th}}$ layer
      $L_{i,h}$ in the model $L_{i,h} = \frac{L_{i,\alpha}}{L_s}$
17: Compute head $(i,j)$ dynamic allocation
      $b_{i,j}^{\text{dyn}} = B_{\text{total}} \cdot (0.01 + L_{i,h}) \cdot S_{j,h}$
18: Compute total KV cache budget of head$(i,j)$
      $b_{i,j} = b_{\text{fixed}} + b_{i,j}^{\text{dyn}}$
19: return C$_{i,j} = \lfloor \max(0, b_{i,j}) + 0.5 \rfloor$
```

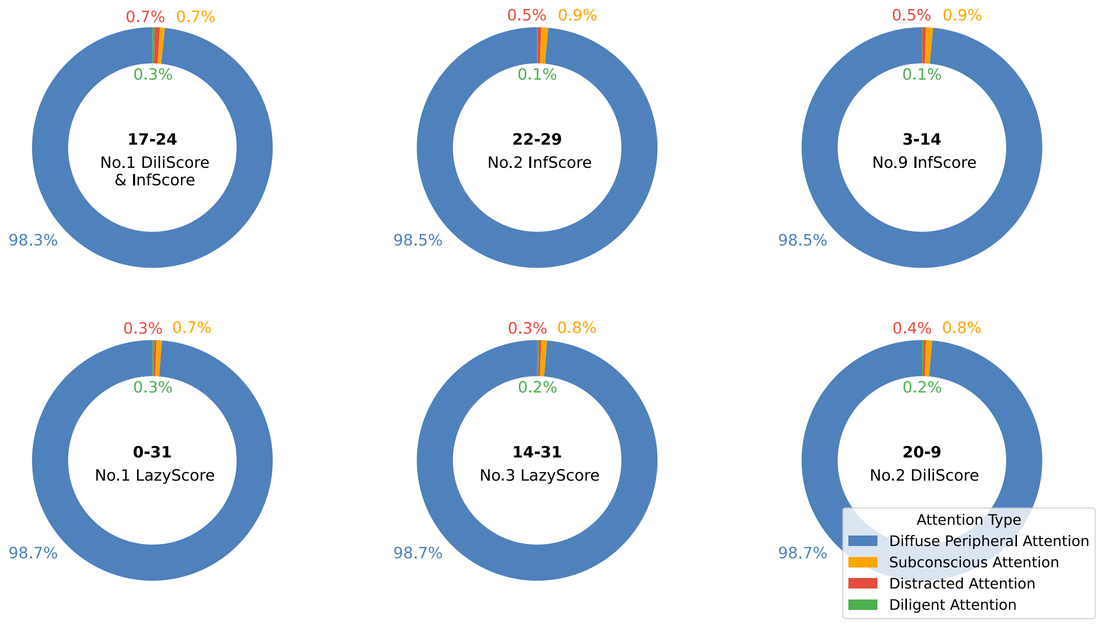


Figure 3: Dili-Lazy attention distribution of representative heads shows two characteristics: (1) Each head plays a dual role, exhibiting both precise (Diligent) and broad (Lazy) patterns simultaneously. Diffuse Peripheral Attention dominates, typically exceeding 95%, indicating a wide-scope, low-intensity scan of non-critical tokens. Subconscious Attention and Distracted Attention each account for less than 1%, reflecting semi-focused or off-target activations. Diligent Attention comprises only about 0.1–0.3%, marking a high-intensity focus on answer-relevant tokens. (2) Different heads work together on downstream tasks. Head 17–24 (top-left) ranks first in both DiliScore and InfScore and has the largest Diligent Attention slice, demonstrating the greatest precision in locating key information. Head 0–31 (bottom-left) leads in LazyScore, highlighting its role in global scanning of long-range context. The other four heads (22–29, 3–14, 14–31, and 20–9) fall between these two examples. Each balances Diligent and Lazy patterns in different proportions to fulfill diverse global versus local information-capturing roles.

3.2 Inference Attention Score (InfScore)

In order to refer to the diligent head's concentration on specific task features (i.e., DiliScore) and lazy head's perception on long-distance contextual dependencies (i.e., LazyScore), we propose *Inference Attention Score (InfScore)* as the harmonic mean of DiliScore and LazyScore in Equation (1). We expect the metric to take a high value only when both the DiliScore and LazyScore are high, and take a low value when one or both are low. For the sake of simplicity in mathematical expression, we simplify DiliScore to D and LazyScore to L.

$$\text{InfScore} = \frac{2 \cdot \mathcal{D} \cdot \mathcal{L}}{\mathcal{D} + \mathcal{L}} \tag{1}$$

Figure 3 highlights six representative attention heads across diligent–lazy attention modes. Figures 4 and 5 offer complementary perspectives on how layer-level insights emerge from head-level behavior, providing a solid foundation for KV cache compression. In particular, Figure 6 clearly illustrates the distinct roles of Layer 0 and 1.

3.3 KV Cache Allocation

Algorithm 1 outlines the main approach for DiliLazyKV method. Almost every attention head exhibits meaningful behavior in terms of either LazyScore or DiliScore, contributing to model's inference quality in at least one dimension. Therefore, a minimum allocation for every head is warranted. The KV cache space assigned to each head is allocated by a combination of a fixed base and a dynamic budget referring to InfScore. If the current sequence length does not exceed the capacity limit, all KV states will be fully retained. Once compression is required, we refer to previous work (Li et al., 2024; Cai et al., 2024; Feng et al., 2024) to obtain contextual content, and use the attention score to evaluate the importance of historical KV states. In order to ensure generation coherence, the KV state within a fixed-size window of the query sequence is always fully retained. Key-value items with higher attention score that are more relevant to the current query will be preserved, maintaining as much critical contextual information as possible.

As a result, DiliLazyKV manages KV cache ef-

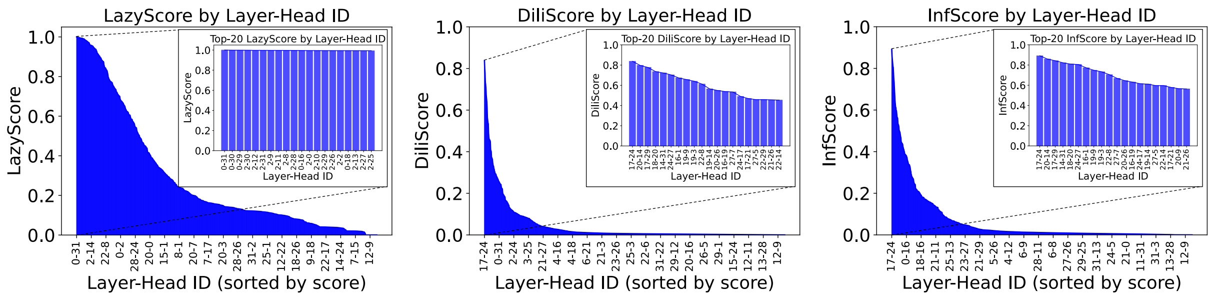


Figure 4: Dili–Lazy attention visualization curves are accompanied by zoomed-in bar charts of the top 20 layer–head IDs, sorted in descending order. These combined views reveal three key characteristics: (1) A skewed distribution is observed across all three metrics: LazyScore, DiliScore, and InfScore. The majority of Layer-Head IDs exhibit relatively low scores, while a small subset possesses significantly higher values. (2) The distribution of the three scores varies considerably among different Layer-Head IDs. Not all attention heads play the same role in the model's diverse capabilities. (3) LazyScore exhibits a more apparent skew compared to the other two scores. Compared to the specialized skill of a small set of heads that consistently and accurately pinpoint the "needle," the majority of attention heads often fail to include it among the top-k predictions.

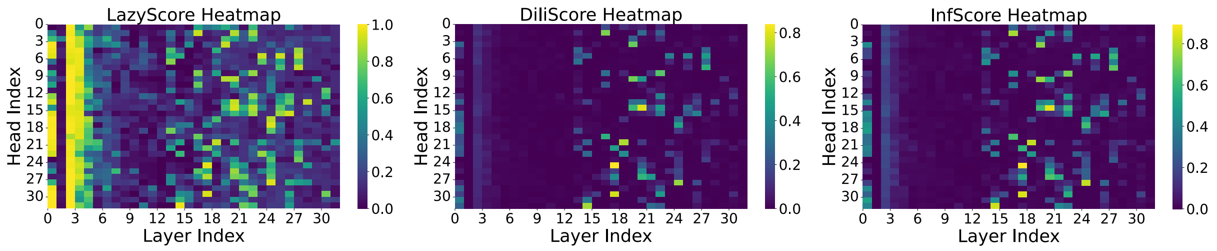
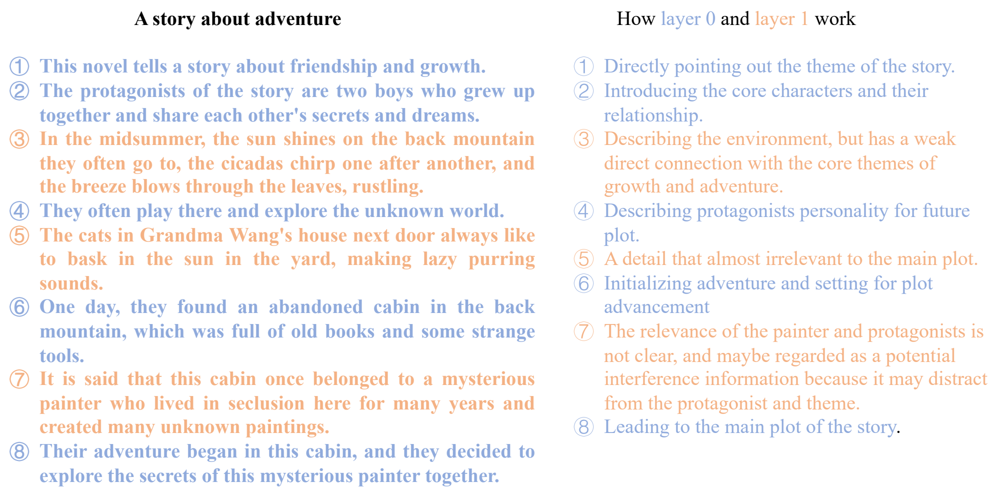


Figure 5: Dili-Lazy attention heatmap visualization on layer-head level shows three characters: (1) The attention heads in the early layers ensure that global information is not missed (high LazyScore). However, their focus is not precise enough (low DiliScore). (2) Layer 1 shows low InfScore. It does not indicate functional uselessness. Instead, a high score in layer 2 reflects the critical role of layer 1. Therefore, the metric should not be evaluated in isolation, but rather in the view of inter-layer dependencies. (3) DiliScore and LazyScore in mid- and late-stage layers (such as Layer 14, 16, 17, 20) show a suitable trade-off for detailed tasks. They can accurately locate and utilize effective contextual information for reasoning, association, and decision-making.



A story about adventure

① This novel tells a story about friendship and growth.
② The protagonists of the story are two boys who grew up together and share each other's secrets and dreams.
③ In the midsummer, the sun shines on the back mountain they often go to, the cicadas chirp one after another, and the breeze blows through the leaves, rustling.
④ They often play there and explore the unknown world.
⑤ The cats in Grandma Wang's house next door always like to bask in the sun in the yard, making lazy purring sounds.
⑥ One day, they found an abandoned cabin in the back mountain, which was full of old books and some strange tools.
⑦ It is said that this cabin once belonged to a mysterious painter who lived in seclusion here for many years and created many unknown paintings.
⑧ Their adventure began in this cabin, and they decided to explore the secrets of this mysterious painter together.

How layer 0 and layer 1 work

① Directly pointing out the theme of the story.
② Introducing the core characters and their relationship.
③ Describing the environment, but has a weak direct connection with the core themes of growth and adventure.
④ Describing protagonists personality for future plot.
⑤ A detail that almost irrelevant to the main plot.
⑥ Initializing adventure and setting for plot advancement
⑦ The relevance of the painter and protagonists is not clear, and maybe regarded as a potential interference information because it may distract from the protagonist and theme.
⑧ Leading to the main plot of the story.

Figure 6: How layer 0 and 1 work for reasoning shows three characteristics. (1) Layer 0 captures the most salient global information positively correlated with downstream tasks (e.g., keywords or key phrases), quickly highlighting core contextual elements and contributing to a higher InfScore. (2) Layer 1 focuses on more subtle long-distance content that might be weakly correlated or even interfere with the global theme, resulting in a lower InfScore. (3) Layer 2, as a slightly deeper layer, begins to differentiate between the helpful information from Layer 0 and the ambiguous disturbance from Layer 1. It learns to filter and integrate truly useful contextual cues for reasoning, achieving a better Diligent–Lazy pattern balance and higher InfScore.

Table 1: DiliLazyKV's more robust performance with fair β**=1.351** on baseline reported on (Fu et al., 2024). The β value for previous work (Fu et al., 2024) is not a fixed constant value **(e.g., 2, 1.2, 1.5,1.01)** but varies with the KV cache size, requiring adjustments based on specific resource constraints and indicating a lack of robustness. In contrast, the DiliLazyKV method balances performance more stably, giving an advantage in practical deployment.

Method	Single-Doc QA			Multi-Doc QA			Avg.	Long Dependency QA				Avg.	β
	NartQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique		DocQA	Info. Retrieval	Timeline	Computation		
Llama-3-8B-Instruct, KV Size = Full													
FullKV	25.56	32.07	39.71	43.57	35.28	21.18	32.90	8.73	11.21	0.67	7.43	7.01	-
Llama-3-8B-Instruct, KV Size = 64													
SnapKV	20.51	12.80	31.69	37.02	25.91	17.02	24.16	8.84	9.43	**0.66**	6.18	6.28	-
PyramidKV	21.17	13.66	29.34	34.86	23.46	15.88	23.06	8.27	9.31	0.63	6.86	6.27	-
Ada-SKV	22.26	17.30	33.37	39.82	27.86	17.85	26.41	9.08	9.86	0.55	6.82	6.58	-
HeadKV-R	22.67	23.54	37.51	37.45	29.76	19.01	28.32	8.80	10.51	0.58	6.68	6.64	2
HeadKV-R2	23.21	25.33	**38.71**	40.64	**31.33**	19.35	29.76	**9.46**	10.66	0.61	6.92	6.91	1.2
DiliLazyKV	**26.22**	**26.30**	38.05	**43.89**	31.06	**20.75**	**31.05**	9.23	**10.67**	0.63	**7.42**	**6.99**	1.351
Llama-3-8B-Instruct, KV Size = 128													
SnapKV	22.11	15.79	31.01	41.12	29.20	19.35	26.43	8.36	9.46	**0.79**	6.56	6.29	-
PyramidKV	22.01	17.05	31.52	39.27	28.99	18.34	26.20	8.89	9.63	0.61	6.72	6.46	-
Ada-SKV	22.99	19.95	34.22	42.97	30.82	20.15	28.52	9.07	10.30	0.54	6.59	6.63	-
HeadKV-R	23.49	25.39	38.15	42.45	32.84	19.95	30.38	8.87	10.35	0.78	7.52	6.88	1.5
HeadKV-R2	21.80	29.19	**41.89**	43.73	**35.01**	20.40	32.00	**9.60**	11.13	0.67	7.22	7.16	1.01
DiliLazyKV	**25.47**	**29.95**	38.02	**44.67**	34.28	**20.66**	**32.18**	9.20	**11.32**	0.62	**7.83**	**7.24**	1.351

fectively and ensures model inference performance, supporting longer and longer sequence generation.

4 Experiments and Analysis

This section will conduct a series of experiments with 8K-context-window Llama-3-8B-Instruct (Grattafiori et al., 2024) to verify the DiliLazy head effect for KV cache compression. The baseline results comes from prior work (Fu et al., 2024), including token-level SnapKV (Li et al., 2024), layer-level PyramidKV (Cai et al., 2024), and head-level approaches, namely HeadKV-R(Fu et al., 2024) and HeadKV-R2 (Fu et al., 2024). The results show that our method can significantly improve the performance of long-context reasoning while maintaining inference performance (4.1). In addition, this section provides a comprehensive comparison of the ablation study (4.2).

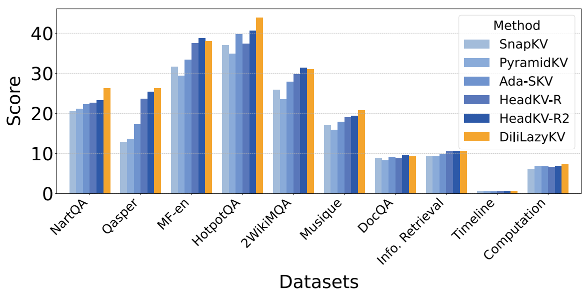
4.1 Main Results

Table 1 presents the evaluation results of the DiliLazyKV method in comparison with baseline methods on the LongBench (Bai et al., 2024) and LooGLE (Li et al., 2023a) benchmark. The baseline results are referenced from (Fu et al., 2024).

Figure 7 shows bar-form comparison. We observe that discovering the distribution of layer-level from head-level and combining the two levels is a very effective method for KV Cache compression. It is worth noting that the DiliLazyKV method shows robust characteristics with stable β, providing reliable credibility for deployment.

4.2 Ablation Study

Table 2 shows ablation results for the diligent-lazy head effect. Figure 8 shows the influence of dif-



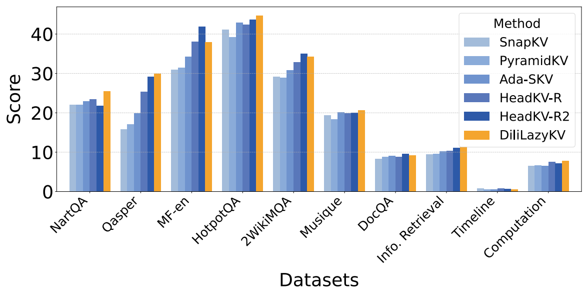
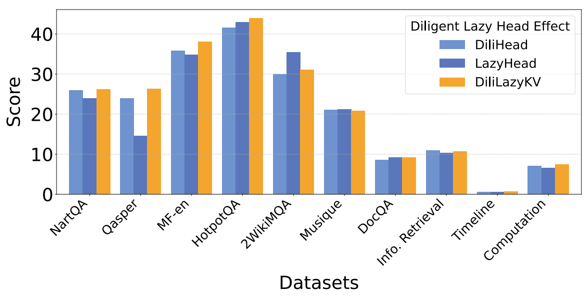


Figure 7: DiliLazyKV's superior performance across various datasets reveals two key characteristics: (1) The relative performance of the different methods varies depending on the dataset. (2) No single baseline method consistently outperforms all others, highlighting the complexity of KV cache compression and the need for adaptive or task-aware approaches.

Table 2: Ablation study for DiliLazy head effect under β=**1.351**. On QA tasks, DiliLazyKV demonstrates the highest average performance with a score of **31.05**, outperforming DiliHead's average of 29.71 and LazyHead's average of 28.81. The numerical difference of over 1.3 points compared to DiliHead and over 2.2 points compared to LazyHead strongly suggests that the coordinated approach implemented in DiliLazyKV is significantly more effective than relying solely on either diligent or lazy attention heads. Without effective coordination of lazy heads, diligent heads may expend excessive effort on seemingly relevant tokens, resulting in insufficient computation and not optimal overall performance.

Method	**Single-Doc QA**			**Multi-Doc QA**			**Avg.**	**Long Dependency QA**				**Avg.**
	NartQA	**Qasper**	**MF-en**	**HotpotQA**	**2WikiMQA**	**Musique**		**DocQA**	**Info. Retrieval**	**Timeline**	**Computation**	
	Llama-3-8B-Instruct, KV Size = 64											
DiliHead	25.94	23.93	35.79	41.57	29.93	**21.07**	29.71	8.61	**10.91**	0.60	7.11	6.81
LazyHead	23.97	14.55	34.74	42.96	**35.44**	21.19	28.81	9.18	10.33	0.54	6.57	6.67
DiliLazyKV	**26.22**	**26.30**	**38.05**	**43.89**	31.06	20.75	**31.05**	**9.23**	10.67	**0.63**	**7.42**	**6.99**
	Llama-3-8B-Instruct, KV Size = 128											
DiliHead	25.92	28.71	36.82	44.46	32.28	20.85	31.52	9.04	11.16	**0.62**	7.40	7.06
LazyHead	**26.14**	26.92	37.46	43.74	**36.56**	20.40	31.87	**9.58**	11.06	0.45	7.58	7.17
DiliLazyKV	25.47	**29.95**	**38.02**	**44.67**	34.28	**20.66**	**32.18**	9.20	**11.32**	**0.62**	**7.83**	**7.24**

ferent components. Both diligent heads and lazy heads contribute distinct benefits, and their combination in DiliLazyKV yields the best average performance. The component does not always synergize positively across all tasks. When the KV size is configured to 64, LazyHead performs best on the 2WikiMQA dataset (from the LongBench benchmark (Bai et al., 2024)), whereas Diligent-Head achieves its excellence on the Info. Retrieval dataset (from the LooGLE benchmark (Li et al., 2023a)). When the KV Size increases to 128, lazy heads contribute more than diligent heads for the whole performance. The combining effects between components are complex and may be closely related to other factors such as model architecture, hyperparameter settings, attention head count, and downstream task characteristics, all of which can modulate the balance between local precision and global context coverage.



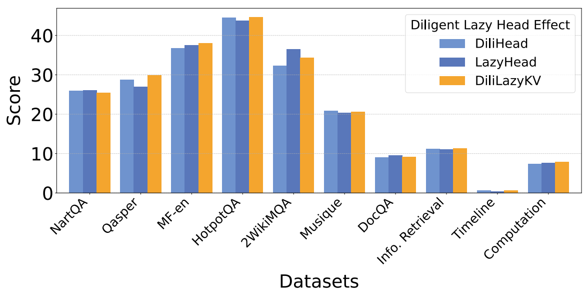


Figure 8: Ablation results on benchmark show two characteristics: (1) DiliLazyKV outperforms the individual Diligent Head and Lazy Head methods on most datasets, effectively combining their strengths. (2) The performance of Diligent Head and Lazy Head individually varies, depending on the specific dataset.

5 Conclusion and Future Work

In the paper, we propose DiliLazyKV, a layer-head level KV cache compression method. We first identify diligent and lazy attention heads, along with their corresponding DiliScore and LazyScore. Furthermore, we introduce the InfScore metric to quantify how both diligent and lazy attention heads contribute to inference. By coordinating at the layer-head level, we allocate varying KV cache budgets to attention heads according to their InfScore. Across multiple benchmarks, DiliLazyKV not only balances KV cache compression and performance but also exhibits enhanced robustness, making it highly beneficial for real-world deployment.

We hope that the paper can bring more inspiration to attention heads to promote the interpretability of LLM and the reduction of hallucinations. KV cache is not only a challenge in the algorithmic domain but also a significant bottleneck concerning memory problems in real-world deployment scenarios. We also hope that the Diligent-Lazy head effect can bring useful insights into software-hardware co-optimization for KV cache research.

6 Limitations

While DiliLazyKV combines layer-level and head-level information, effectively leveraging layer-level insights to guide KV cache allocation may require more in-depth research, as simple layer-wise statistics might not fully exploit the complex dependencies between different layers. The collaborative relationships between layers and heads still need extensive investigation to discover more optimized KV Cache compression methods.

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 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901.

Anthropic. 2025. Claude 3.7 Sonnet and Claude Code.

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 Annual Meeting of the Association for Computational Linguistics*, pages 3119–3137.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan-Kelley. 2024. Reducing Transformer Key-Value Cache Size with Cross-Layer Attention. *Advances in Neural Information Processing Systems*, 37:86927–86957.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, and 1 others. 2024. Pyramidkv: Dynamic KV Cache Compression based on Pyramidal Information Funneling. *Preprint*, arXiv:2406.02069.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, and 48 others. 2023. PaLM: Scaling Language Modeling with Pathways. *Journal of Machine Learning Research*, 24:1–113.

Tri Dao. 2024. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. In *Proceedings of International Conference on Learning Representations*, pages 1–14.

Google DeepMind. 2025. Gemini 2.5 Pro.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2024. Ada-KV: Optimizing KV Cache Eviction by Adaptive Budget Allocation for Efficient LLM Inference. *Preprint*, arXiv:2407.11550.

Yao Fu. 2024. Challenges in Deploying Long-Context Transformers: A Theoretical Peak Performance Analysis. *Preprint*, arXiv:2405.08944.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2024. Not All Heads Matter: A Head-Level KV Cache Compression Method with Integrated Retrieval and Reasoning. *Preprint*, arXiv:2410.19258.

Team GLM. 2024. ChatGLM: A Family of Large Language Models from GLM-130B to GLM-4 All Tools. *Preprint*, arXiv:2406.12793.

Gemini Team Google. 2023. Gemini: A Family of Highly Capable Multimodal Models. *Preprint*, arXiv:2312.11805.

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

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMLingua: Compressing Prompts for Accelerated Inference of Large Language Models. In *Proceedings of Conference on Empirical Methods in Natural Language Processing*, page 13358–13376.

Greg Kamradt. 2023. Needle In A Haystack - LLM Test.

Jiaqi Li, Mengmeng Wang, Zilong Zheng, and Muhan Zhang. 2023a. LooGLE: Can Long-Context Language Models Understand Long Contexts? *Preprint*, arXiv:2311.04939.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, and 48 others. 2023b. StarCoder: may the source be with you! *Transactions on Machine Learning Research*, 2023:1–55.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024. SnapKV: LLM Knows What You Are Looking for Before Generation. *Advances in Neural Information Processing Systems*, 37:22947–22970.

Ruikang Liu, Haoli Bai, Haokun Lin, Yuening Li, Han Gao, Zhengzhuo Xu, Lu Hou, Jun Yao, and Chun Yuan. 2024. IntactKV: Improving Large Language Model Quantization by Keeping Pivot Tokens Intact.

Findings of the Association for Computational Linguistics, pages 7716–7741.

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. *Advances in Neural Information Processing Systems*, 36:52342–52364.

Piotr Nawrot, Adrian Łańcucki, Marcin Chochowski, David Tarjan, and Edoardo M Ponti. 2024. Dynamic Memory Compression: Retrofitting LLMs for Accelerated Inference. *Proceedings of Machine Learning Research*, 235:37396–37412.

OpenAI. 2025. Introducing OpenAI o3 and o4-mini.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. 2024. SparQ Attention: Bandwidth-Efficient LLM Inference. In *Proceedings of International Conference on Machine Learning*, pages 42558–42583.

Noam Shazeer. 2019. Fast Transformer Decoding: One Write-Head Is All You Need. *Preprint*, arXiv:1911.02150.

Hanlin Tang, Yang Lin, Jing Lin, Qingsen Han, Shikuan Hong, Yiwu Yao, and Gongyi Wang. 2024. RazorAttention: Efficient KV Cache Compression Through Retrieval Heads. *Preprint*, arXiv:2407.15891.

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, and 49 others. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. *Preprint*, arXiv:2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Proceedings of Neural Information Processing Systems*, pages 1–11.

Guangtao Wang, Shubhangi Upasani, Chen Wu, Darshan Gandhi, Jonathan Li, Changran Hu, Bo Li, and Urmish Thakker. 2025. LLMs Know What to Drop: Self-Attention Guided KV Cache Eviction for Efficient Long-Context Inference. *Preprint*, arXiv:2503.08879.

Wenhao Wu, Yizhong Wang, Guangxuan Xiao, Hao Peng, and Yao Fu. 2024. Retrieval Head Mechanistically Explains Long-context Factuality. *Preprint*, arXiv:2404.15574.

Chaojun Xiao, Pengle Zhang, Xu Han, Guangxuan Xiao, Yankai Lin, Zhengyan Zhang, Zhiyuan Liu, and Maosong Sun. 2024a. InfLLM: Training-Free Long-Context Extrapolation for LLMs with an Efficient Context Memory. *Advances in Neural Information Processing Systems*, 37:119638–119661.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2024b. DuoAttention: Efficient Long-Context LLM Inference with Retrieval and Streaming Heads. *Preprint*, arXiv:2410.10819.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024c. Efficient Streaming Language Models with Attention Sinks. In *Proceedings of International Conference on Learning Representations*, pages 1–21.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H_2O: Heavy-Hitter Oracle for Efficient Generative Inference of Large Language Models. *Advances in Neural Information Processing Systems*, 36:34661–34710.

Meizhi Zhong, Xikai Liu, Chen Zhang, Yikun Lei, Yan Gao, Yao Hu, Kehai Chen, and Min Zhang. 2025. ZigZagKV: Dynamic KV Cache Compression for Long-context Modeling based on Layer Uncertainty. In *Proceedings of International Conference on Computational Linguistics*, pages 8897–8907.

A Needle Example

1. **Question:** "What are good ways to spend time in campus?"
   Needle: "The good ways to spend time in campus include relax and do nothing."
   Answer: "Relax and do nothing."

2. **Question:** "What habits are beneficial for health during Ph.D. career?"
   Needle: "The beneficial habits during Ph.D. career contain exercise and healthy diet."
   Answer: "Exercise and healthy diet."

3. **Question:** "Which year did Tom start high school?"
   Needle: "Tom was born in 2005. Tom started high school fifteen years after he was born."
   Answer: "Tom started high school in 2020."

B Dataset Details

The work uses the same datasets as previous work (Fu et al., 2024). Table 3 shows the details of the six question-answering datasets from LongBench (Bai et al., 2024) and the four question-answering datasets from LooGLE (Li et al., 2023a). They are all English datasets and use F1 score as evaluation metric.

Table 3: Details of Datasets (Label, Task and Average Length).

Label	Task	Avg Len
NrtvQA	NarrativeQA	18,409
Qasper	Qasper	3,619
MF-en	MultiFieldQA-EN	4,559
HotpotQA	HotpotQA	9,151
2WikiMultiHopQA	2WikiMultiHopQA	4,887
Musique	Musique	11,214
Doc.QA	Comprehension & reasoning	15,498
Info.Retrieval	Multiple information retrieval	14,808
Timeline	Timeline reorder	15,425
Computation	Computation	17,001

C TOP 10 Heads in LazyScore, DiliScore, and InfScore Metric Respectively

Table 4, Table 5 and Table 6 list representative scores from diligent and lazy head.

Table 4: Top 10 Heads by LazyScore.

Layer–Head ID	LazyScore
0–31	1.0
0–30	0.9983710690153388
0–29	0.9983553861104881
2–30	0.9979897022090343
2–12	0.9974390687495067
2–31	0.99742711187279
2–9	0.9970726671242299
2–11	0.9968930120652112
2–8	0.9968449894025974
2–28	0.9964799377628338

Table 5: Top 10 Heads by DiliScore.

Layer–Head ID	DiliScore
17–24	0.8377500583283318
20–14	0.7966464731600884
17–29	0.7769381458858717
18–20	0.7353739569464377
14–31	0.7222048680531286
24–27	0.7033271446398407
16–1	0.6728415614392841
19–9	0.6568146203877396
19–3	0.6402092661234061
22–8	0.6120521144671319

Table 6: Top 10 Heads by InfScore.

Layer–Head ID	InfScore
17–24	0.8914607342709882
20–14	0.8590445781510154
17–29	0.8426021128243087
14–31	0.8222214106683441
18–20	0.8093112523053374
24–27	0.8051997416416302
16–1	0.7725543106433685
19–9	0.7478591136516957
19–3	0.7333235532693692
22–8	0.7047732219595361