EchoKV: Efficient KV Cache Compression via Similarity-Based Reconstruction

Shiyu Ji* **Yixuan Wang*** **Yijun Liu** **Qingfu Zhu** **Wanxiang Che†**
Research Center for Social Computing and Interactive Robotics,
Harbin Institute of Technology, China
{syji,car}@ir.hit.edu.cn
https://github.com/noforit/EchoKV

Abstract

The increasing memory demand of the Key-Value (KV) cache poses a significant bottleneck for Large Language Models (LLMs) in long-context applications. Existing low-rank KV compression methods reduce this footprint by modifying model projections, limiting the flexibility to switch back to standard full-cache inference when sufficient memory is available. In this paper, we propose EchoKV, a flexible KV cache compression framework that supports on-demand transitions from full KV caching to compressed caching. Unlike traditional compression-decompression paradigms, EchoKV utilizes a lightweight network to reconstruct the discarded KV components from a partial subset, exploiting intrinsic inter-layer and intra-layer similarities among attention heads. We further introduce a lightweight two-stage fine-tuning strategy, requiring only a few minutes on a single A100 GPU for Llama-3.1-8B. Experimental results on LongBench and RULER demonstrate that EchoKV offers a competitive quality–cache trade-off across multiple compression ratios and backbone models while preserving the throughput of full-cache inference in short-context scenarios.

1 Introduction

While the KV cache significantly accelerates Large Language Model inference by reducing redundant computations, it introduces a heavy memory burden that scales linearly with context length. As LLMs advance to support massive context windows for applications like repository-scale code generation [1] and deep reasoning [2] via ultra-long Chain-of-Thought [3], the memory overhead of the KV cache has become a prohibitive bottleneck, catalyzing a surge of research [4] into compression methodologies.

Recent research on KV cache compression has primarily centered on low-rank sharing. Similar to the Multi-Head Latent Attention (MLA) [5], Palu [6] achieves training-free low-rank sharing by performing Singular Value Decomposition (SVD) within attention head groups, effectively reducing the KV cache footprint. Furthermore, CommonKV [7] leverages the intrinsic similarity of KV caches across layers to extend low-rank sharing to a multi-head, multi-layer scope, achieving superior performance under low memory budgets.

While these methods effectively compress the KV cache footprint during inference, they all involve modifying model parameters. This parameter modification makes it difficult to switch between standard full-cache inference and compressed inference at runtime, thereby compromising inference performance when memory is abundant. Although some online compression methods [8, 9] offer

*Equal contribution.
†Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

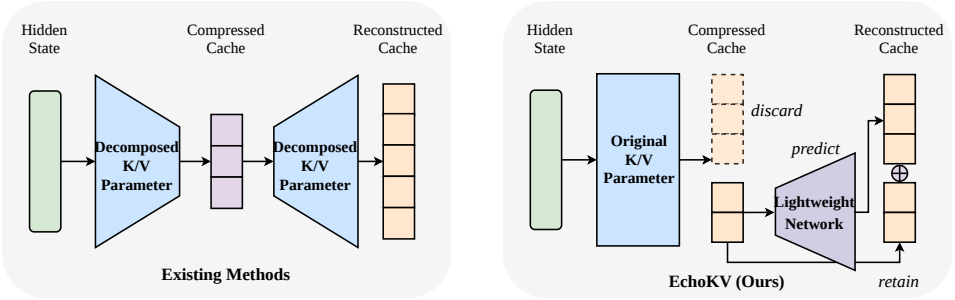


Figure 1: Illustration of the differences between existing low-rank sharing approaches and EchoKV. Unlike the compression-decompression paradigm, EchoKV employs a lightweight network to reconstruct the discarded KV components from retained ones.

selective compression, they struggle to maintain performance under tight memory budgets and incur additional latency overheads due to online compression. Consequently, there remains a lack of a flexible compression approach capable of reconciling **high performance on short contexts** with **low resource costs on long contexts**.

In this paper, we propose EchoKV, a more flexible KV cache compression scheme that enables on-demand switching from standard inference to compressed inference while maintaining performance. As shown in Figure 1, unlike other low-rank methods that perform uniform compression and reconstruction on the entire KV cache, EchoKV employs a lightweight network to predict the remaining KV pairs from a partial subset, thereby enabling a seamless transition from standard attention. Specifically, inspired by the observation of inter-layer [10, 7] and intra-layer similarities [11, 12] in KV representations, we propose to bypass the explicit compression process. Instead, we directly train a lightweight network to predict the remaining KV cache based on partial KV information. To ensure both the integrity and accuracy of the input information, we utilize the full KV cache from specific layers alongside a partial subset from the current layer as input features to predict the remaining KV components of the current layer. Furthermore, we employ a memory-friendly two-stage fine-tuning strategy that enables rapid and low-cost acquisition of the lightweight reconstruction network, requiring only a few minutes on a single A100 GPU for Llama-3.1-8B.

To validate the effectiveness of the proposed method, we evaluate models of different scales, ranging from 7B to 24B parameters, on two long-context benchmarks: LongBench [13], which comprises real-world data, and RULER [14], which consists of synthetic data. Experimental results demonstrate that EchoKV achieves competitive performance across different compression ratios, with particular advantages at lower cache budgets. Moreover, the capability to flexibly switch from FullKV to EchoKV maintains high throughput for short inputs while keeping long-context KV cache within memory limits, which provides significant practical advantages relative to alternative techniques. Through combination with low-rank techniques, EchoKV yields additional performance gains, specifically tailored to the distinct properties of keys and values.

The main contributions of this paper are summarized as follows:

- We propose EchoKV, a KV cache compression method that reconstructs the remaining cache using specific KV information. It enables flexible switching from FullKV to EchoKV.
- We introduce an efficient training method for the lightweight reconstruction network, allowing training to be completed in a time comparable to offline SVD.
- Experiments demonstrate that our proposed method achieves competitive performance across multiple benchmarks, compression ratios, and model scales while maintaining high throughput in short-context scenarios.

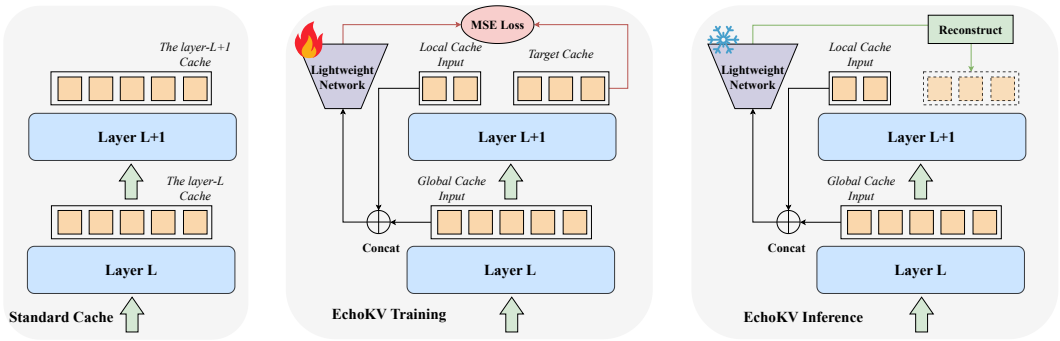


Figure 2: Schematic illustration of the training and inference workflows for EchoKV compared to the standard KV cache. The figure presents a schematic illustration for a single token, where distinct cache blocks correspond to different segments of the flattened KV representation.

2 Related work

2.1 KV cache compression

Quantization. KV cache quantization methods are akin to existing activation quantization techniques, yet they exhibit unique characteristics [15]. Based on observations of KV cache distributions, KIVI [16] adopts a combination of per-channel quantization for Keys and per-token quantization for Values, achieving 2-bit lossless performance. KVQuant [17] defines quantization levels using a non-uniform distribution learned from calibration data, further enhancing information density. AQUA-KV [18] uses lightweight predictors to exploit KV dependencies and stores quantized prediction residuals. Its predictors use the previous reconstructed layer, with Value prediction additionally using the current reconstructed Keys. EchoKV instead reconstructs discarded dimensions from a full group-anchor cache and retained target-layer dimensions, without storing residuals for the discarded dimensions. It is worth noting that quantization methods are orthogonal to EchoKV. By further quantizing the retained KV cache to low-bit precision, we can achieve even higher compression ratios.

Dimensionality reduction. Beyond compressing data bit-width, recent work [4] has also focused on directly compressing the dimensions of the KV cache. Architecturally, Multi-Query Attention (MQA) [19] and Grouped-Query Attention (GQA) [20] serve as explicit dimensional compression mechanisms for the KV cache. Furthermore, Multi-Head Latent Attention (MLA) [5] replaces explicit grouping with implicit grouping, utilizing low-rank projections to achieve significantly lower KV cache memory usage. Recent efforts further convert existing pre-trained attention modules into MLA-style architectures, reducing KV cache memory but still requiring substantial continued training to recover model performance [21, 22].

In addition to architectural improvements that require pre-training from scratch, recent work has also explored lightweight methods to compress KV cache dimensions. Palu [6] employs inter-group SVD decomposition and matrix fusion techniques, compressing the cache footprint while maintaining inference efficiency. CommonKV [7] extends this approach by applying SVD for inter-layer sharing, yielding better results even with limited memory budgets.

2.2 KV cache eviction

Based on the observation [23] that a small subset of important tokens consistently receives subsequent attention, KV cache eviction methods [24] significantly reduce cache occupancy in long-context scenarios. SnapKV [25] evaluates token importance by using the last block of the input as an observation window. LAQ [26] achieves importance assessment aligned with the generation phase by rapidly generating a pseudo-response as the observation window. Judge-Q [27] further trains a set of soft tokens to serve as an observation window, achieving more precise importance assessment. Since eviction methods operate at the granularity of individual tokens, they are orthogonal to EchoKV and can be combined with our method.

3 Methodology

3.1 Overview of EchoKV

As illustrated in Figure 2, the core idea of EchoKV is to retain a partial KV representation and use a lightweight network to reconstruct the discarded components on demand. By bypassing the explicit compression phase, our approach allows for directly evicting a portion of the KV cache when memory bottlenecks are encountered, and subsequently reconstructing it during inference using the retained KV information. Serving as the core component of the reconstruction process, the lightweight network is detailed in this section, covering both its model architecture (§3.2) and training specifications (§3.3).

3.2 Network architecture

Motivated by the effectiveness of SVD-based projections in prior KV compression methods, we adopt **a simple linear layer $\mathbf{W}_i$** as the default prediction network. This choice prioritizes inference efficiency by keeping the architecture lightweight. Leveraging the inter-layer and intra-layer similarity of the KV cache, the network input is composed of two parts: the full flattened KV representation of a specific layer serving as the global input, and a retained partial representation of the current layer acting as the local input. Aligning with prior works [6, 7], we utilize keys prior to Rotary Positional Embedding (RoPE) [28] as inputs to enhance model adaptation. Keys and Values use separate predictors with the same structure; $\mathbf{C}$ denotes either cache. For each token, we flatten the head and feature dimensions of the KV cache into a single feature axis and perform reconstruction on this flattened representation. Predictor dimensions are detailed in Appendix B.1, Table 6.

Global cache inputs. To leverage inter-layer similarity, we partition the KV cache layers into distinct groups. Within each group, the first layer retains its full cache to serve as global information, facilitating the prediction of the remaining layers in that group. Formally, the global input for the k-th group is defined as:

$$\mathbf{C}^{(k)}_{\text{global}} = \mathbf{C}_{k \cdot S}, \tag{1}$$

where S represents the group size (i.e., the number of layers in each group), and $\mathbf{C}_{k \cdot S}$ is the full KV cache of the first layer in that group.

Local cache inputs. Furthermore, to mitigate potential information loss across layers, we incorporate a retained subset of flattened KV dimensions from the current layer as the local input, thereby enhancing prediction accuracy. For the i-th layer within the k-th group, the local cache input can be formulated as:

$$\mathbf{C}^{(k,i)}_{\text{local}} = \mathbf{C}_{k \cdot S+i}[:, :\text{m}], \tag{2}$$

where $\mathbf{C}_{k \cdot S+i}$ is the full cache of the layer, and m is the number of retained local feature dimensions used to preserve intra-layer details. Considering the efficiency of contiguous memory access, we directly retain the first m flattened dimensions as the local input based on the target budget, without employing complex selection heuristics. Appendix D.2 and Appendix D.3 further examine the effects of the group size S, the retained local dimension m, and alternative local-dimension selection strategies.

Synthesizing the above, we concatenate the global and local features of each token and employ the lightweight model $\mathbf{W}_{k \cdot S+i}$ to predict the evicted cache. This process can be formulated as:

$$\hat{\mathbf{C}}^{(k,i)}_{\text{drop}} = \mathbf{W}_{k \cdot S+i} \left[\mathbf{C}^{(k)}_{\text{global}} \,;\, \mathbf{C}^{(k,i)}_{\text{local}} \right]. \tag{3}$$

The ground truth for the prediction corresponds to the discarded flattened dimensions $\mathbf{C}^{(k,i)}_{\text{drop}}$ that are evicted during inference:

$$\mathbf{C}^{(k,i)}_{\text{drop}} = \mathbf{C}_{k \cdot S+i}[:, \text{m}:]. \tag{4}$$

When $m = 0$, the group's first layer supplies the entire predictor input. We choose the smallest group whose first-layer cache fits the target budget, then allocate the remaining budget to local dimensions. Appendix D.2 gives the budget equations and motivation; Appendix B.2 lists the configurations.

Inference and switching. Prefill attention uses full K/V while storing the compact cache; decoding reconstructs dropped dimensions before attention. An existing FullKV cache can be compacted mid-session without recomputing the prefix. Appendix B.4 describes the procedure and switching protocol, with results in Appendix C.5.

3.3 Training details

To fully train the lightweight network and ensure the usability of the reconstructed cache, we employ a two-stage training strategy to progressively enhance the performance of EchoKV.

Reconstruction loss. In the initialization stage, we directly employ the reconstruction MSE loss computed on the corresponding Key-Value pairs for training. This can be formulated as:

$$\mathcal{L}_{\text{init}} = \left\| \hat{\mathbf{C}}_{\text{drop}} - \mathbf{C}_{\text{drop}} \right\|_2^2. \tag{5}$$

In this phase, we strive for a faithful reconstruction of the KV cache, which serves as a robust weight initialization for the subsequent training stage. However, we observe that relying solely on reconstruction yields suboptimal performance in long-context scenarios (as detailed in §5.1). We attribute this limitation to the fact that pure reconstruction loss does not directly optimize the downstream attention output. Consequently, we proceed to a second training stage to further refine the network under actual attention mechanisms.

Attention loss. Formally, the reconstructed Key and Value representations for the i-th layer within the k-th group are obtained by concatenating the retained local features and the predicted features, followed by reshaping them back to the original KV layout:

$$\tilde{\mathbf{X}}_{k,i} = \text{Reshape}\left(\left[\mathbf{X}_{\text{local}}^{(k,i)} \; ; \; \hat{\mathbf{X}}_{\text{drop}}^{(k,i)} \right] \right), \quad \mathbf{X} \in \{\mathbf{K}, \mathbf{V}\}, \tag{6}$$

where $[\cdot \; ; \; \cdot]$ denotes the concatenation operation, $\mathbf{X}_{\text{local}}^{(k,i)}$ denotes the retained local features, and $\hat{\mathbf{X}}_{\text{drop}}^{(k,i)}$ denotes the predicted components for $\mathbf{X} \in \{\mathbf{K}, \mathbf{V}\}$.

To ensure computational efficiency in long-context scenarios, we eschew the standard KL divergence loss, which typically requires materializing the full attention matrix to align $\mathbf{Q}\mathbf{K}^T$ and $\mathbf{Q}\tilde{\mathbf{K}}^T$ (detailed in Appendix D.1). Instead, we adopt an Output MSE (O-MSE) loss that is fully compatible with FlashAttention [29]. This approach facilitates efficient training while jointly optimizing both the reconstructed Keys ($\tilde{\mathbf{K}}$) and Values ($\tilde{\mathbf{V}}$), as formulated below:

$$\mathcal{L}_{\text{attention}} = \left\| \mathcal{A}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) - \mathcal{A}(\mathbf{Q}, \tilde{\mathbf{K}}, \tilde{\mathbf{V}}) \right\|_2^2, \tag{7}$$

where $\mathcal{A}(\cdot)$ denotes the FlashAttention operation.

It is worth noting that, despite the adoption of a two-stage training strategy, the training process remains highly efficient and can be completed within a very short timeframe. This is attributed to the lightweight nature of our network, which contains up to approximately 50M parameters for Llama-3.1-8B (detailed in Table 6). Specifically, we can complete the training of EchoKV for Llama-3.1-8B in only a few minutes on a single A100 GPU.

4 Experiments

4.1 Experimental setting

Datasets. To comprehensively evaluate the effectiveness of our compression method, we conduct experiments on both the real-world long-context benchmark LongBench [13] and the synthetic dataset RULER [14]. Specifically, we utilize the maximum context length of models for LongBench, while using 32K sequences for the main RULER evaluation. Additional 64K/128K tests and predictor-seed evaluations are reported in Appendices C.6 and C.7.

Regarding the training dataset, we explore a diverse range of options. As the results in §5.3 indicate, the proposed method is insensitive to the specific choice of training data. We ultimately employ LongAlpaca [30] for our experiments.

Table 1: Experimental results of different models and methods on LongBench. * Since ThinK only compresses Keys, we double the compression rate calculation to account for Value storage. We report results at 0.6 because the 0.5 ratio is infeasible. Compression Ratio = Size of Compressed Cache / Size of Full Cache.

	Methods	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
		Nrtv	Qasp	MF	Hotpot	2Wiki	Musiq	Gov	QMS	MN	TREC	TQA	SAM	PC	PR	LCC	RB	
Llama-3.1-8B-Instruct	FullKV	31.38	46.60	56.85	58.10	50.01	32.52	34.66	25.19	27.11	73.00	92.11	43.70	6.56	100.0	65.52	54.75	49.88
	Compression Ratio = 0.7																	
	MiniCache	17.87	22.26	29.51	26.72	11.47	9.71	24.61	21.85	23.29	65.00	75.05	31.46	5.83	60.50	41.34	37.19	31.48
	ThinK	**32.77**	45.44	**58.83**	**58.30**	**51.09**	**32.65**	**32.63**	24.96	**26.19**	**73.00**	91.72	42.10	**6.96**	99.50	**65.94**	55.64	**49.86**
	Palu	28.96	**46.73**	53.26	53.08	45.35	29.91	32.51	24.69	25.42	**73.00**	89.10	**43.77**	2.50	96.50	57.90	**56.78**	47.47
	CommonKV	29.87	41.16	53.86	54.90	45.42	29.96	27.91	24.21	25.68	66.00	90.03	43.10	6.67	**100.0**	61.92	53.41	47.13
	EchoKV	32.24	44.53	57.84	57.90	49.89	32.08	29.29	**25.06**	25.02	**73.00**	**92.52**	41.67	6.87	99.00	63.57	54.72	49.08
	Compression Ratio = 0.5																	
	ThinK*	28.51	22.92	37.42	52.90	47.84	28.01	19.67	21.96	19.01	38.00	88.11	40.72	7.50	**99.50**	59.97	49.92	41.37
	Palu	19.27	34.64	40.14	41.99	26.71	22.45	22.60	23.70	23.21	66.00	25.95	39.75	4.00	49.50	29.86	30.19	31.25
	CommonKV	25.24	40.64	54.45	54.56	45.58	29.82	25.23	23.85	24.92	59.00	89.98	**42.50**	**10.12**	**99.50**	62.46	52.50	46.27
	EchoKV	**33.06**	**44.80**	**57.51**	**57.91**	**50.25**	**31.79**	**29.22**	**24.10**	**25.21**	**73.00**	**91.92**	41.45	7.17	89.50	**64.61**	**54.91**	**48.53**
	Compression Ratio = 0.3																	
	Palu	1.90	2.42	4.36	1.00	1.70	0.66	3.52	9.28	5.95	1.25	3.19	6.31	0.00	0.00	18.20	18.79	4.91
	CommonKV	11.20	29.58	36.89	31.21	31.30	14.79	13.86	21.22	21.82	47.50	83.38	40.25	0.22	8.50	54.43	51.10	31.08
	EchoKV	**31.22**	**42.47**	**54.27**	**57.03**	**49.58**	**30.75**	**24.23**	**23.84**	**22.32**	**49.00**	**91.79**	**40.90**	**7.25**	**84.50**	**62.09**	**53.03**	**45.27**
Mistral-7B-Instruct-v0.3	FullKV	29.80	39.12	50.45	50.28	36.31	26.45	34.11	25.93	26.56	76.00	88.89	46.97	5.50	97.00	61.47	62.57	47.34
	Compression Ratio = 0.7																	
	MiniCache	14.49	21.16	26.26	18.38	18.16	6.33	19.12	21.06	23.22	61.00	82.68	32.60	2.67	24.17	44.28	40.75	28.52
	ThinK	**30.42**	**37.87**	50.13	50.35	34.55	25.92	**33.81**	**25.66**	26.40	**76.00**	88.43	45.81	**5.50**	95.00	61.90	61.95	46.86
	Palu	28.47	36.08	**52.56**	46.57	**36.12**	**26.37**	33.59	25.16	**26.54**	73.50	87.77	45.11	4.00	94.50	59.04	61.06	46.03
	CommonKV	26.02	37.13	48.21	47.61	29.87	24.43	30.05	24.75	25.76	74.00	88.12	44.54	4.00	92.50	61.16	62.10	45.02
	EchoKV	29.93	37.65	49.43	**51.50**	34.54	24.98	32.29	25.11	25.99	**76.00**	**89.47**	**47.44**	**5.50**	**95.50**	**62.26**	**62.18**	**46.86**
	Compression Ratio = 0.5																	
	ThinK*	29.27	27.37	39.90	45.95	31.83	20.41	22.24	23.21	19.80	65.50	87.27	43.43	**6.00**	80.00	59.03	59.32	41.28
	Palu	25.84	35.28	47.43	46.69	31.93	**27.21**	29.32	24.32	**25.37**	74.50	86.31	42.95	4.50	61.00	48.19	47.90	41.17
	CommonKV	27.27	34.71	46.70	44.74	27.02	24.21	26.36	23.52	24.42	51.00	88.56	43.82	4.50	88.00	60.14	59.94	42.18
	EchoKV	**30.26**	**36.92**	**49.71**	**51.33**	**34.26**	24.51	**31.54**	**25.28**	25.36	**76.00**	**89.32**	**46.93**	3.50	**93.50**	**61.72**	**61.66**	**46.36**
	Compression Ratio = 0.3																	
	Palu	11.04	11.91	23.13	16.00	14.48	8.77	10.60	20.50	16.87	58.50	58.98	26.55	**4.50**	4.08	23.34	25.56	20.93
	CommonKV	11.96	21.25	29.05	21.47	14.49	8.40	22.30	22.55	**24.43**	56.50	72.81	37.21	2.63	14.50	52.46	39.16	28.20
	EchoKV	**27.99**	**35.01**	**47.29**	**49.77**	**31.63**	**24.01**	**25.88**	**23.86**	23.63	**67.00**	**89.06**	**45.00**	4.00	**70.00**	**59.27**	**61.05**	**42.78**
Mistral-Small-24B-Instruct-2501	FullKV	33.72	47.19	54.95	67.24	64.93	49.04	33.22	24.54	25.47	75.50	93.71	48.86	21.00	100.0	63.67	70.66	54.61
	Compression Ratio = 0.7																	
	MiniCache	31.70	35.78	44.52	61.25	53.13	39.06	25.38	22.30	22.92	72.00	92.39	37.20	17.50	67.50	37.97	48.19	44.30
	ThinK	34.43	41.31	51.85	**67.44**	**64.09**	47.23	24.61	23.98	20.62	61.00	93.24	46.86	19.00	**100.0**	61.30	69.05	51.63
	Palu	34.71	**47.09**	55.35	66.61	63.78	44.51	**32.68**	**25.72**	25.53	74.50	92.12	**48.22**	14.50	**100.0**	54.65	65.63	52.85
	CommonKV	**36.16**	46.59	54.82	66.00	62.80	46.29	32.13	24.85	**25.55**	**76.00**	93.16	48.20	18.00	99.50	**68.02**	**71.11**	54.32
	EchoKV	33.74	46.73	**55.47**	67.18	64.07	**49.50**	31.71	25.28	24.92	75.50	**93.64**	**48.22**	**19.50**	**100.0**	64.38	70.98	**54.43**
	Compression Ratio = 0.5																	
	ThinK*	26.96	30.79	35.81	56.43	57.52	39.26	19.81	21.56	16.84	37.00	91.51	44.48	17.50	99.00	56.30	61.10	44.49
	Palu	28.82	44.48	47.35	61.33	53.18	37.76	23.24	**24.79**	22.87	73.00	91.12	43.68	9.50	97.00	22.62	38.07	44.93
	CommonKV	**35.98**	46.14	54.49	66.59	63.45	47.10	30.78	24.58	**25.52**	74.00	92.91	47.59	19.00	**100.0**	**66.90**	**70.02**	54.07
	EchoKV	32.78	**46.33**	**56.23**	**67.14**	**64.25**	**49.07**	**30.88**	24.69	24.38	**75.50**	**93.39**	**48.37**	**20.00**	**100.0**	64.14	69.87	**54.19**
	Compression Ratio = 0.3																	
	Palu	4.77	6.85	16.03	7.47	5.98	6.47	7.06	15.09	10.45	36.50	22.94	16.84	1.00	0.00	21.35	22.41	12.58
	CommonKV	27.41	41.87	48.97	55.45	53.02	28.62	26.16	22.74	**24.72**	72.50	90.17	45.51	1.23	95.00	**70.49**	68.78	48.29
	EchoKV	**30.77**	**44.80**	**54.43**	**66.78**	**62.74**	**49.01**	**27.57**	**24.66**	23.05	**74.50**	**93.23**	**45.76**	**18.50**	**99.50**	63.25	**69.95**	**53.03**

Baselines. We select *Palu* [6] and *CommonKV* [7], two SVD-based state-of-the-art (SOTA) methods, as strong baselines. Furthermore, we compare our approach with the low-rank method *ThinK* [31] and the post-compression merging technique *MiniCache* [8], and validate the potential for integrating these methods with our proposed *EchoKV*. We also include *TransMLA* [21], which converts existing attention modules into an MLA-style architecture, as an additional architecture-conversion baseline. For the backbone models, we employ Llama-3.1-8B-Instruct [32], Mistral-7B-Instruct-v0.3 [33], and Mistral-Small-24B-Instruct-2501, covering model scales from 7B to 24B.

Implementation details. To ensure the transferability of our method, we avoid specific hyper-parameter search or tuning for layer-wise budgets. Instead, we use the same group size S and retained local dimension m across layer groups (detailed in Appendix B.2). Considering the impact of attention sinks [34], we follow established baselines by retaining 4 sink tokens and a local window of 128 tokens at full precision. Reported target ratios describe the compressed token region; dimensional rounding and full-token storage are detailed in Appendix B.2. Appendix B.3 presents the hyperparameters for the lightweight network training stage.

We also evaluate an exploratory hybrid variant that combines low-rank compression for Keys with EchoKV reconstruction for Values, motivated by their asymmetric reconstruction difficulty. Detailed hybrid results and discussion are provided in Appendix C.3. We further defer several implementation

Table 2: Experimental results of different methods on RULER at a compression ratio of 0.5. * Indicates the actual compression ratio is 0.6, consistent with Table 1.

Method	Needle-in-a-Haystack (NIAH)								Synthetic			QA		Avg.
	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	
Llama-3.1-8B-Instruct														
FullKV	100.0	100.0	100.0	99.00	99.00	99.00	98.75	98.50	99.20	17.80	87.00	87.00	54.00	87.63
ThinK*	1.00	2.00	0.00	2.00	0.00	0.00	1.75	3.25	0.60	0.00	48.00	63.00	40.00	12.43
Palu	**100.0**	98.00	75.00	97.00	78.00	30.00	73.00	64.25	86.00	**15.30**	78.33	53.00	38.00	68.14
CommonKV	92.00	94.00	83.00	97.00	67.00	1.00	87.00	94.25	79.80	0.10	**85.67**	75.00	50.00	69.68
EchoKV	**100.0**	**100.0**	**100.0**	**99.00**	**99.00**	**94.00**	**89.50**	**99.50**	**87.20**	0.60	80.00	**85.00**	**52.00**	**83.52**
Mistral-7B-Instruct-v0.3														
FullKV	100.0	92.00	100.0	84.00	80.00	57.00	92.25	92.25	95.80	80.70	89.67	65.00	47.00	82.74
ThinK*	21.00	58.00	2.00	32.00	17.00	5.00	20.25	14.00	44.20	17.10	**89.67**	60.00	42.00	32.47
Palu	**100.0**	**97.00**	98.00	76.00	51.00	**26.00**	**88.50**	**80.75**	83.00	32.10	85.00	41.00	33.00	68.56
CommonKV	86.00	46.00	36.00	27.00	18.00	0.00	7.00	12.75	66.60	15.10	75.00	46.00	**44.00**	36.88
EchoKV	**100.0**	88.00	**99.00**	**82.00**	**59.00**	19.00	62.75	67.50	**85.20**	**46.60**	81.67	**65.00**	43.00	**69.13**
Mistral-Small-24B-Instruct-2501														
FullKV	98.00	100.0	100.0	95.00	98.00	94.00	99.75	99.00	100.0	95.90	92.00	83.00	63.00	93.67
ThinK*	0.00	1.00	1.00	1.00	0.00	0.00	2.75	2.50	0.40	12.80	25.33	59.00	46.00	11.68
Palu	98.00	**100.0**	68.00	97.00	84.00	76.00	**94.50**	96.25	71.20	34.20	80.00	55.00	58.00	77.86
CommonKV	**99.00**	99.00	98.00	**98.00**	92.00	90.00	**94.50**	**98.00**	89.60	71.10	**94.00**	81.00	**62.00**	89.71
EchoKV	**99.00**	**100.0**	**100.0**	95.00	**98.00**	**95.00**	85.00	96.25	**98.00**	**75.20**	92.33	**83.00**	60.00	**90.52**

ablations to Appendix D.2, Appendix D.3, and Appendix D.4, covering the choices of (S, m), local-dimension selection strategy, and predictor architecture.

4.2 Main results

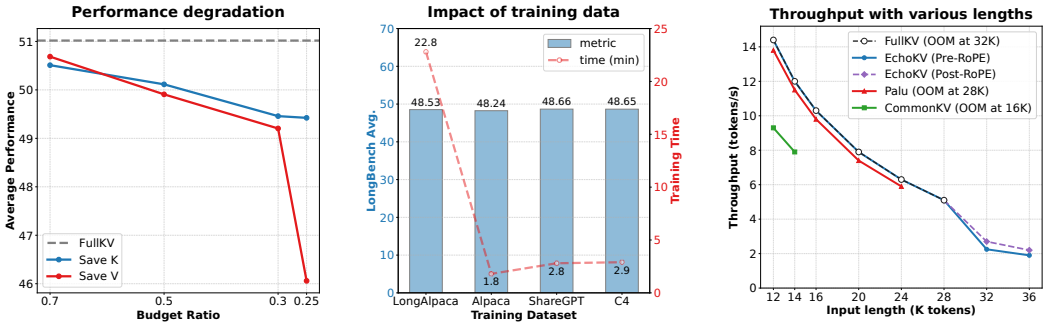
Results on LongBench. We evaluate the performance of various baseline methods across multiple models on LongBench [13], as shown in Table 1. By capturing latent similarities in KV representations, EchoKV achieves competitive performance, with advantages at lower cache budgets. At ratio 0.7, ThinK scores higher on Llama-3.1-8B (49.86 vs. 49.08) and ties EchoKV on Mistral-7B (46.86). While methods based on parameter SVD decomposition mitigate performance degradation, they fail to enable flexible switching during inference. Using both local and global KV information, EchoKV retains LongBench averages close to FullKV at ratio 0.5 and outperforms the evaluated compression baselines at 0.3. These results support a setting-dependent quality–cache trade-off alongside flexible inference switching. Notably, KV compression is typically orthogonal to eviction and quantization [6, 7], allowing these techniques to be combined to achieve even higher compression ratios. We also report an exploratory hybrid variant in Appendix C.3, where heterogeneous treatment of Keys and Values yields further gains in several moderate-budget settings.

Results on RULER. Beyond the real-world datasets of LongBench, we also perform a detailed assessment on the synthetic long-context benchmark, RULER [14]. Table 2 presents the results at a compression ratio of 0.5, while additional results, including the hybrid variant, are reported in Appendix C. It can be observed that compared to robust real-world datasets, the differences between methods are more significant on synthetic tasks. At ratio 0.5, EchoKV exceeds the evaluated compression baselines on average, with gaps to FullKV of 4.11, 13.61, and 3.15 points for Llama-3.1-8B, Mistral-7B, and Mistral-Small-24B, respectively. The appendix further shows that the hybrid variant can bring additional gains on some RULER settings, although this advantage is not uniform across models and compression regimes.

Table 3: LongBench average scores compared with TransMLA. "Converted" denotes direct architecture conversion, while "FT" denotes additional training on the same 1.6K LongAlpaca samples used by EchoKV.

Model	Ratio	Converted	FT	EchoKV
Llama-3.1-8B	0.5	27.65	36.89	**48.53**
Llama-3.1-8B	0.3	11.76	32.31	**45.27**
Mistral-7B	0.5	21.15	29.88	**46.36**
Mistral-7B	0.3	12.36	25.00	**42.78**

At 64K/128K, EchoKV scores 79.63/62.54 on Llama-3.1-8B, compared with FullKV's 84.13/75.46 (Appendix C.6). Across four predictor-training seeds on this model at ratios 0.7 and 0.5, standard deviations of benchmark averages are at most 0.14 on LongBench and 0.27 on RULER (Appendix C.7).



(a) Difficulty analysis of the EchoKV reconstruction task for keys and values. "Save K" denotes only values are reconstructed.

(b) Performance and training time of the lightweight network across different training datasets.

(c) Throughput of FullKV, adaptive EchoKV, and static compression baselines across input lengths.

Figure 3: Analysis experiments on EchoKV. All evaluations use Llama-3.1-8B-Instruct [32]; (a,b) report LongBench [13] results.

Comparison with TransMLA. Table 3 compares EchoKV with TransMLA [21] on LongBench. Direct architecture conversion causes severe degradation, and TransMLA requires substantial continued training to recover performance. Even after additional training on the same 1.6K LongAlpaca samples used by EchoKV, it still remains clearly behind EchoKV across both backbones and compression ratios. Detailed conversion settings and additional analysis are provided in Appendix C.2.

5 Analysis

5.1 Selection of objective functions

As discussed in §3.3, we employ a two-stage training strategy with distinct loss functions to optimize the lightweight network. In this section, we analyze different loss functions to validate the effectiveness of our proposed method.

As shown in Table 4, optimizing O-MSE from scratch is less effective than first fitting the dropped KV entries with KV-MSE. This suggests that the first stage mainly serves as a representation-alignment step: before the predictor learns a faithful cache reconstruction, the attention-space objective is too indirect to provide a strong optimization signal. Once such an initialization is available, however, O-MSE becomes a better second-stage objective because it directly optimizes the downstream attention outputs.

We also compare O-MSE with the commonly used KL divergence loss on QK attention distributions (detailed in Appendix D.1). While QK-KL and O-MSE reach similar final performance, O-MSE is considerably more practical for long-context training. Unlike QK-KL, it remains compatible with FlashAttention [29, 35] and avoids explicit materialization of the $L \times L$ attention matrix. In our setting, this reduces training time by nearly $3\times$ without sacrificing final quality, which makes the two-stage training procedure efficient in practice. Appendix B.3 reports the complete training schedules and corresponding RULER results.

Table 4: Training time and performance across different loss functions.

Loss	Stage	Total time	LB Avg.
KV-MSE	I	12.5 min	48.99
O-MSE	I	15.4 min	48.76
QK-KL	II	39.7 min	49.11
O-MSE	II	14.5 min	**49.26**

5.2 Asymmetric reconstruction difficulty of keys and values

Recent studies [16, 36] indicate that keys and values exhibit distinct numerical distributions, necessitating separate treatment. To compare the reconstruction difficulty of keys and values using EchoKV, we conduct experiments where we reconstruct them separately. As shown in Figure 3a, the "Save K" setting ensures key integrity by applying reconstruction exclusively to the values. We evaluate the performance of the two configurations on LongBench across different budgets. Experimental

Table 5: Results of the ablation study on different input features. Evaluations are conducted on LongBench using Llama-3.1-8B-Instruct with a compression ratio of 0.75. **Bolded** entries represent datasets where the two features exhibit differences.

Methods	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code	
	Nrtv	Qasp	MF	Hotpot	2Wiki	Musiq	Gov	QMS	MN	TREC	TQA	SAM	PC	PR	LCC	RB
FullKV	31.38	46.60	56.85	58.10	50.01	32.52	**34.66**	25.19	27.11	73.00	92.11	43.70	6.56	**100.00**	65.52	54.75
Local Input	31.88	46.45	57.32	58.02	49.49	32.68	**30.80**	24.98	26.16	73.00	92.50	42.45	6.88	**98.50**	64.72	55.53
Global Input	31.67	46.10	57.85	58.19	49.54	32.81	**32.27**	24.63	26.32	73.00	92.41	41.77	7.00	**88.00**	64.34	55.32
Combined Input	31.60	46.48	57.43	58.27	49.84	33.02	**32.34**	25.09	26.22	73.00	92.58	42.49	7.00	**94.50**	64.46	55.73

results indicate that compressing keys is more challenging than compressing values. As the budget decreases, predicting only keys causes significant performance degradation, whereas predicting only values performs well. This suggests that the inductive bias exploited by EchoKV, namely cross-layer and intra-layer similarity, is stronger for Values than for Keys. Combined with prior observations that Keys are often more amenable to low-rank compression [6], this asymmetry indicates that heterogeneous treatment of Keys and Values may be a promising direction for future KV compression methods.

5.3 Impact of training data

Despite its high efficiency, EchoKV fundamentally requires training on a given dataset. We experiment with different training datasets to verify robustness. We test four datasets: long instruction dataset LongAlpaca [30], short instruction dataset Alpaca [37], multi-turn instruction dataset ShareGPT [38], and a subset of pre-training dataset C4 [39].

As shown in Figure 3b, we train the lightweight network on four distinct datasets and perform a unified evaluation on LongBench. The resulting LongBench averages are highly comparable across these datasets, even on the noisier pre-training subset. This suggests that the lightweight predictor is primarily calibrating the model-specific similarity structure within the KV space, rather than relying strongly on the semantics of a particular training corpus. In addition, while performance remains comparable, training durations vary substantially due to sequence length. Even without sufficient long-context data, for Llama-3.1-8B we can train the lightweight network on short sequences in roughly 2 minutes on a single A100 GPU and still obtain strong long-context performance.

5.4 Throughput analysis

EchoKV supports on-demand compression as memory demand grows. Figure 3c compares adaptive EchoKV with FullKV and static compression baselines on a single NVIDIA A100-SXM4-80GB, with batch size 8, 16 output tokens, and a compression ratio of 0.5. EchoKV uses the native FullKV path through 28K and activates compression at 32K/36K, where FullKV runs out of memory. The Pre-RoPE and Post-RoPE variants reach 2.25/1.90 and 2.70/2.20 tokens/s at these lengths, respectively. The quality benchmarks use compressed caches from prefill; the adaptive curves instead reflect when compression is enabled.

Appendix C.5 evaluates length routing and memory-triggered conversion on 20 batches of 8 sequences, with prompts of 4K, 8K, 32K, and 36K. Both adaptive policies complete all 20 batches, compared with 10 for FullKV, and retain comparable short-prompt time per output token (TPOT). Memory-triggered conversion costs 193.52/165.52 ms at 32K/36K, followed by TPOT of 475.72/534.09 ms. Cache reconstruction adds decoding work, trading latency for the capacity to complete longer requests. Fusing reconstruction with cache assembly and writing directly into the attention buffer reduces intermediate materialization and redundant memory traffic.

5.5 Ablation of input features

As discussed in §3.2, the predictor input consists of the retained global cache from the first layer of the group and the partial local cache from the current layer. We perform an ablation study to validate the effectiveness of various input features, with the results presented in Table 5. To ensure a fair evaluation, we fix the compression ratio at 0.75 and reconstruct the remaining cache using only different input features. Experimental results indicate that global and local features achieve

similar effectiveness across most datasets, demonstrating substantial redundancy both across layers and within the flattened per-layer KV representation. However, in some summarization and synthetic tasks, the two types of features exhibit significant performance differences. Fortunately, by combining these two features, we achieve consistent performance improvements. We adopt this configuration in our main experiments, as it retains the lower-layer global cache to mitigate error accumulation within the group, while utilizing the local cache to ensure intra-layer precision.

6 Conclusion

In this paper, we introduced EchoKV, a novel and flexible KV cache compression framework designed to alleviate the memory bottleneck in long-context LLM inference. By employing a lightweight network to predict the remaining cache segments from a subset, we achieve competitive performance across multiple benchmarks, with particular advantages at lower cache budgets. We further report an exploratory hybrid variant in the appendix, indicating that differentiated compression mechanisms for Keys and Values are a promising direction beyond the main EchoKV design.

Acknowledgments and Disclosure of Funding

We gratefully acknowledge the support of the National Natural Science Foundation of China (NSFC) via grant 62236004 and 62476073.

References

[1] Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? *arXiv preprint arXiv:2310.06770*, 2023.

[2] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[3] Qiguang Chen, Libo Qin, Jinhao Liu, Dengyun Peng, Jiannan Guan, Peng Wang, Mengkang Hu, Yuhang Zhou, Te Gao, and Wanxiang Che. Towards reasoning era: A survey of long chain-of-thought for reasoning large language models. *arXiv preprint arXiv:2503.09567*, 2025.

[4] Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. Keep the cost down: A review on methods to optimize LLM's KV-cache consumption. *arXiv preprint arXiv:2407.18003*, 2024.

[5] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. DeepSeek-V2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[6] Chi-Chih Chang, Wei-Cheng Lin, Chien-Yu Lin, Chong-Yan Chen, Yu-Fang Hu, Pei-Shuo Wang, Ning-Chi Huang, Luis Ceze, Mohamed S Abdelfattah, and Kai-Chiang Wu. Palu: Compressing KV-cache with low-rank projection. *arXiv preprint arXiv:2407.21118*, 2024.

[7] Yixuan Wang, Haoyu Qiao, Lujun Li, Qingfu Zhu, and Wanxiang Che. CommonKV: Compressing KV cache with cross-layer parameter sharing. *arXiv preprint arXiv:2508.16134*, 2025.

[8] Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. MiniCache: KV cache compression in depth dimension for large language models. *Advances in Neural Information Processing Systems*, 37:139997–140031, 2024.

[9] Chi-Chih Chang, Chien-Yu Lin, Yash Akhauri, Wei-Cheng Lin, Kai-Chiang Wu, Luis Ceze, and Mohamed S Abdelfattah. XKV: Cross-layer SVD for KV-cache compression. *arXiv preprint arXiv:2503.18893*, 2025.

[10] Wonbeom Lee, Jungi Lee, Junghwan Seo, and Jaewoong Sim. InfiniGen: Efficient generative inference of large language models with dynamic KV cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, pages 155–172, 2024.

[11] Yixuan Wang, Huang He, Siqi Bao, Hua Wu, Haifeng Wang, Qingfu Zhu, and Wanxiang Che. ProxyAttn: Guided sparse attention via representative heads. *arXiv preprint arXiv:2509.24745*, 2025.

[12] Dan Peng, Zhihui Fu, Zewen Ye, Zhuoran Song, and Jun Wang. Accelerating prefilling for long-context LLMs via sparse pattern sharing. *arXiv preprint arXiv:2505.19578*, 2025.

[13] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, et al. 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)*, pages 3119–3137, 2024.

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

[15] Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. QAQ: Quality adaptive quantization for LLM KV cache. *arXiv preprint arXiv:2403.04643*, 2024.

[16] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. *arXiv preprint arXiv:2402.02750*, 2024.

[17] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

[18] Alina Shutova, Vladimir Malinovskii, Vage Egiazarian, Denis Kuznedelev, Denis Mazur, Nikita Surkov, Ivan Ermakov, and Dan Alistarh. Cache me if you must: Adaptive key-value quantization for large language models. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 55451–55473. PMLR, 2025. URL `https://proceedings.mlr.press/v267/shutova25a.html`.

[19] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[20] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. GQA: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[21] Fanxu Meng, Pingzhi Tang, Xiaojuan Tang, Zengwei Yao, Xing Sun, and Muhan Zhang. TransMLA: Multi-head latent attention is all you need. *arXiv preprint arXiv:2502.07864*, 2025.

[22] Guihong Li, Mehdi Rezagholizadeh, Mingyu Yang, Vikram Appia, and Emad Barsoum. X-EcoMLA: Upcycling pre-trained attention into MLA for efficient and extreme KV compression. *arXiv preprint arXiv:2503.11132*, 2025.

[23] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for LLM KV cache compression at test time. *Advances in Neural Information Processing Systems*, 36:52342–52364, 2023.

[24] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2O: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36:34661–34710, 2023.

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

[26] Yixuan Wang, Shiyu Ji, Yijun Liu, Yuzhuang Xu, Yang Xu, Qingfu Zhu, and Wanxiang Che. Lookahead Q-Cache: Achieving more consistent KV cache eviction via pseudo query. *arXiv preprint arXiv:2505.20334*, 2025.

[27] Yijun Liu, Yixuan Wang, Yuzhuang Xu, Shiyu Ji, Yang Xu, Qingfu Zhu, and Wanxiang Che. Judge Q: Trainable queries for optimized information retention in KV cache eviction. *arXiv preprint arXiv:2509.10798*, 2025.

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

[29] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

[30] Yukang Chen, Shaozuo Yu, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. LongAlpaca: Long-context instruction-following models. `https://github.com/dvlab-research/LongLoRA`, 2023.

[31] Yuhui Xu, Zhanming Jie, Hanze Dong, Lei Wang, Xudong Lu, Aojun Zhou, Amrita Saha, Caiming Xiong, and Doyen Sahoo. ThinK: Thinner key cache by query-driven pruning. *arXiv preprint arXiv:2407.21018*, 2024.

[32] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[33] Albert Qiaochu 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. Mistral 7b. *ArXiv*, abs/2310.06825, 2023. URL `https://api.semanticscholar.org/CorpusID:263830494`.

[34] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[35] Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024.

[36] Wanyun Cui and Mingwei Xu. Homogeneous keys, heterogeneous values: Exploiting local KV cache asymmetry for long-context LLMs. *arXiv preprint arXiv:2506.05410*, 2025.

[37] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford Alpaca: An instruction-following LLaMA model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

[38] ShareAI Lab. ShareGPT-chinese-english-90k: A bilingual chinese-english human-machine dialogue dataset, 2023. URL `https://huggingface.co/datasets/shareAI/ShareGPT-Chinese-English-90k`. License: Apache-2.0.

[39] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

Appendices

A Discussion and limitations 14

B Experimental details 14
- B.1 Hyperparameter configurations 14
- B.2 Compression-ratio configurations 14
- B.3 Training details 15
- B.4 Inference and cache conversion 15

C Additional results 17
- C.1 Results on RULER 17
- C.2 Comparison with TransMLA 17
- C.3 Hybrid results 18
- C.4 Visualization of Needle-in-a-Haystack 19
- C.5 Adaptive inference and switching 20
- C.6 Ultra-long context 21
- C.7 Seed robustness 21

D Additional analysis 21
- D.1 Analysis of QK-KL divergence loss 21
- D.2 Group size, local dimension, and prediction distance 22
- D.3 Local-dimension selection 23
- D.4 Prediction network architecture 23

E Use of AI tools 24

A Discussion and limitations

As discussed in Appendix D.3, EchoKV adopts a simple prefix-based rule for retaining local dimensions, primarily for efficiency and contiguous memory access. While this heuristic is already competitive, the results also suggest that more sophisticated selection strategies may still offer modest gains in some regimes. Designing adaptive local-dimension selection mechanisms that better balance performance and implementation cost remains an open direction for future work.

We also find an intriguing asymmetry between Keys and Values. Our analysis suggests that the EchoKV-style reconstruction mechanism is more effective for Values than for Keys, which contrasts with prior compression methods such as Palu [6] and XKV [9], where Keys are generally the easier component to compress. This discrepancy indicates that Keys and Values may favor fundamentally different compression mechanisms, and a deeper study of their distributional and structural differences could lead to more principled hybrid designs.

Reconstructing and assembling K/V adds decoding latency, while tighter cache budgets can reduce task quality. The evaluated conversion proceeds from FullKV to EchoKV; recovering discarded entries relies on reconstruction. Serving measurements use the current PyTorch implementation on an A100-80GB, so latency and memory limits depend on the kernels and hardware. The 64K/128K and multi-seed evaluations cover Llama-3.1-8B, with seed variation tested at ratios 0.7 and 0.5.

B Experimental details

B.1 Hyperparameter configurations

Let $D = H_{\mathrm{KV}} d_{\mathrm{head}}$ be the flattened dimension of Keys or Values. For each non-anchor layer ($S > 1$), separate affine K/V predictors map $[B, L, D + m]$ to $[B, L, D - m]$, where B is the batch size and L is the number of tokens to reconstruct. Each has a $(D - m) \times (D + m)$ weight matrix and a bias of length $D - m$. At $m = 0$, this becomes a $D \to D$ map from anchor to target. Table 6 lists the experimental configurations.

Table 6: EchoKV configurations for Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Mistral-Small-24B-Instruct-2501. Predictor dimensions are shared across these models; approximate total K/V predictor counts refer to Llama-3.1-8B-Instruct and depend on the number of non-anchor layers.

Target Ratio	**Group Size (S)**	**Local Dim (m)**	**Predictor Input Dim**	**Predictor Output Dim**	**Total Params**
0.7	2	384	1408	640	28.8M
0.5	2	0	1024	1024	33.6M
0.3	4	64	1088	960	50.2M

B.2 Compression-ratio configurations

To achieve the target compression ratios, we choose the group size S and retained local dimension m using the budget construction in Appendix D.2. In our experiments, Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Mistral-Small-24B-Instruct-2501 all utilize Grouped Query Attention (GQA) with 8 KV heads and a head dimension of 128. Consequently, the flattened dimension of either Keys or Values is identical across these models, i.e., $D = 8 \times 128 = 1024$. Based on this shared KV layout, the three models use the same predictor configurations in Table 6. For models with different KV dimensions, the same construction applies by adjusting the predictor input and output dimensions according to the target compression ratio. For the three main configurations, the continuous local budgets $m^* = 409.6$, 0, and 68.27 are rounded down to $m = 384$, 0, and 64, respectively, using multiples of 64.

Compression ratio = 0.7. We set the group size to $S = 2$. In the compressed layer, we retain a local feature dimension of $m = 384$, corresponding to retaining 384 flattened local dimensions. Consequently, the input to the lightweight linear network consists of the full global cache from the preceding layer (1024) concatenated with the local cache (384), resulting in an input dimension

of 1408. The network predicts the remaining discarded features ($1024 - 384 = 640$). Thus, the lightweight network $\mathbf{W}$ has a shape of $1408 \to 640$. The nominal cache ratio for the compressed token region is $(1024 + 384)/(1024 \times 2) = 0.6875$.

Compression ratio = 0.5. We set the group size to $S = 2$. In this setting, we retain no local flattened dimensions ($m = 0$) for the compressed layer, relying entirely on the inter-layer similarity from the global input. The linear layer maps the global input (1024) to the full cache of the current layer (1024). The nominal cache ratio for the compressed token region is $(1024 + 0)/(1024 \times 2) = 0.5$.

Compression ratio = 0.3. We increase the group size to $S = 4$. For the three compressed layers within the group, we retain a minimal local feature dimension of $m = 64$, corresponding to retaining only 64 flattened local dimensions. The lightweight network takes the global input (1024) and the small local embedding (64) to reconstruct the remaining information ($1024 - 64 = 960$). The nominal cache ratio for the compressed token region is $(1024 + 64 \times 3)/(1024 \times 4) = 0.296875$.

A summary of these configurations is provided in Table 6. These ratios describe complete layer groups over the compressed token region. Full sink/window tokens and the decode tail contribute additional cache storage; Appendix B.4 details the memory accounting. In the batch-1 switching test at ratio 0.5, the measured persistent cache is 1.010/2.000 GiB at 16K and 2.010/4.000 GiB at 32K (EchoKV/FullKV), corresponding to 50.50%/50.25%. The BF16 predictors add 0.0626 GiB, bringing cache-plus-predictor storage to 53.6%/51.8% of the FullKV cache; total peak GPU memory also includes the backbone and temporary tensors.

B.3 Training details

We implement EchoKV using the PyTorch framework. The lightweight reconstruction network is optimized using the **AdamW** optimizer. We employ a **Cosine** learning rate scheduler with no warmup steps to adjust the learning rate during training. To minimize memory overhead, the batch size is set to 1 for both stages. For the main experiments, we set the global random seed to 42 for PyTorch and other relevant libraries; Appendix C.7 additionally evaluates independently trained predictors with seeds 43–45.

The main training configuration uses two stages to progressively refine reconstruction quality:

- **Stage 1 (Initialization):** The network is trained for 600 steps using the direct Key-Value reconstruction loss (KV-MSE) to establish a solid initialization.
- **Stage 2 (Adaptation):** We switch to the attention-aware output loss (O-MSE) and continue training for another 1,000 steps to align the reconstructed cache with the attention mechanism.

Detailed hyperparameter configurations are provided in Table 7.

Training schedules and objective ablation. The main LongAlpaca configuration takes 22.8 minutes on one A100; the shorter Alpaca, ShareGPT, and C4 settings in Figure 3b train in a few minutes with comparable LongBench performance. The objective ablation in Table 4 instead uses 1,000 total steps: either a single objective throughout or 700 KV-MSE steps followed by 300 second-stage steps. Table 8 gives the full schedules and total workflow times, along with RULER averages over 13 tasks. Compared with KV-MSE alone, two-stage O-MSE improves LongBench/RULER by 0.27/1.70 points with two additional minutes of training.

B.4 Inference and cache conversion

Prefill and decoding. Prefill attention uses full K/V, while the cache is stored directly in EchoKV form. Each group retains a full anchor cache; other layers retain full K/V for 4 sink tokens and a 128-token local window, plus the first m flattened dimensions for intervening tokens. Decoding reconstructs this middle region from the anchor and local features, then assembles it with the retained full K/V for attention. Newly generated tokens are appended in full to the cache tail.

Conversion from FullKV. Inference starts on the native FullKV path. A trigger based on generation length or memory usage compacts the existing cache to a preselected (S, m) configuration, retaining

Table 7: Hyperparameter details of the EchoKV training process. In Stage 2, only the parameters that differ from those in Stage 1 are described.

Configuration	Value
Stage 1	
Optimizer	AdamW
Learning rate	$5e{-4}$
Scheduler	Cosine Decay
Warmup steps	0
Batch size	1
Training steps	600
Loss function	KV-MSE
Stage 2	
Optimizer	AdamW
Learning rate	$5e{-4}$
Scheduler	Cosine Decay
Warmup steps	0
Batch size	1
Training steps	1000
Loss function	O-MSE

Table 8: Complete objective-ablation results. Times cover the entire training workflow; schedules list the steps for each objective in order.

Training objective	Steps	Total time	LongBench	RULER
KV-MSE only	1,000	12.5 min	48.99	81.23
O-MSE only	1,000	15.4 min	48.76	78.74
KV-MSE then QK-KL	$700 + 300$	39.7 min	49.11	81.33
KV-MSE then O-MSE	$700 + 300$	14.5 min	**49.26**	**82.93**

the anchors, sink/window tokens, and local dimensions. Decoding continues with the corresponding trained predictors, reusing the cached prefix.

RoPE handling. The main Pre-RoPE configuration reconstructs unrotated Keys and applies RoPE at their original positions before attention. Conversion from native FullKV applies inverse RoPE before compaction. The Post-RoPE variant reconstructs rotated Keys, saving the inverse transformation at switching and repeated rotation during decoding.

Memory accounting. Persistent KV storage comprises anchors, full sink/window caches and the decode tail, and retained local dimensions. Predictor parameters are counted separately; reconstructed K/V form per-layer workspace. Total peak GPU memory also includes backbone parameters, activations, and temporary tensors for conversion and attention.

Mid-session switching protocol. We evaluate Llama-3.1-8B-Instruct at ratio 0.5 with greedy decoding. Switch@t generates t tokens with FullKV, converts the populated cache, and generates token $t + 1$ onward with EchoKV. Quality is evaluated at $t \in \{32, 64\}$ on GovReport, QMSum, and MultiNews using the LongBench prompts and generation limits. Timing uses Switch@32 with 16K/32K prompts, 256 output tokens, and batch size 1 on an A100-80GB. Latencies are averaged over five warmed-up runs. Switch latency includes inverse RoPE for Pre-RoPE; time per output token (TPOT) measures steady decoding after conversion.

Adaptive workload. Using the same model and ratio with Pre-RoPE, we evaluate 20 batches of 8 sequences (160 sequences): five batches each at 4K, 8K, 32K, and 36K, with 16 output tokens and the same shuffled order for every policy. The length policy routes prompts of at least 32K to EchoKV before prefill. The memory policy starts with FullKV and converts completed layers at a 62-GiB

Table 9: Experimental results of different methods on RULER at compression ratios of 0.5 and 0.3. * Indicates the actual compression ratio is 0.6.

Method	Needle-in-a-Haystack (NIAH)								Synthetic			QA		Avg.
	S1	S2	S3	MK1	MK2	MK3	MV	MQ	VT	CWE	FWE	QA1	QA2	
Llama-3.1-8B-Instruct														
Compression Ratio = 0.5														
FullKV	100.0	100.0	100.0	99.00	99.00	99.00	98.75	98.50	99.20	17.80	87.00	87.00	54.00	87.63
ThinK*	1.00	2.00	0.00	2.00	0.00	0.00	1.75	3.25	0.60	0.00	48.00	63.00	40.00	12.43
Palu	**100.0**	98.00	75.00	97.00	78.00	30.00	73.00	64.25	86.00	**15.30**	78.33	53.00	38.00	68.14
CommonKV	92.00	94.00	83.00	97.00	67.00	1.00	87.00	94.25	79.80	0.10	**85.67**	75.00	50.00	69.68
EchoKV	**100.0**	**100.0**	**100.0**	**99.00**	**99.00**	94.00	89.50	99.50	87.20	0.60	80.00	**85.00**	52.00	83.52
EchoKV-Hybrid	**100.0**	**100.0**	**100.0**	**99.00**	**99.00**	**99.00**	**97.00**	**100.0**	**97.40**	13.80	79.67	**85.00**	**54.00**	**86.45**
Compression Ratio = 0.3														
Palu	11.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.30	1.00	0.00	0.00	0.95
CommonKV	45.00	28.00	0.00	13.00	0.00	0.00	7.50	8.50	**38.40**	**9.90**	39.33	21.00	30.00	18.51
EchoKV	**83.00**	**81.00**	**72.00**	70.00	**79.00**	9.00	57.25	88.00	37.00	0.00	**75.67**	**82.00**	**50.00**	**60.30**
EchoKV-Hybrid	57.00	79.00	46.00	**88.00**	78.00	**17.00**	**71.25**	**93.50**	31.40	0.10	67.67	**82.00**	**50.00**	58.53
Mistral-7B-Instruct-v0.3														
Compression Ratio = 0.5														
FullKV	100.0	92.00	100.0	84.00	80.00	57.00	92.25	92.25	95.80	80.70	89.67	65.00	47.00	82.74
ThinK*	21.00	58.00	2.00	32.00	17.00	5.00	20.25	14.00	44.20	17.10	89.67	60.00	42.00	32.47
Palu	**100.0**	**97.00**	98.00	76.00	51.00	26.00	88.50	80.75	83.00	32.10	85.00	41.00	33.00	68.56
CommonKV	86.00	46.00	36.00	27.00	18.00	0.00	7.00	12.75	66.60	15.10	75.00	46.00	44.00	36.88
EchoKV	**100.0**	88.00	99.00	82.00	59.00	19.00	62.75	67.50	85.20	46.60	81.67	65.00	43.00	69.13
EchoKV-Hybrid	**100.0**	89.00	**100.0**	**83.00**	**77.00**	**42.00**	**92.50**	**90.25**	**90.80**	**66.30**	**93.67**	**67.00**	**47.00**	**79.89**
Compression Ratio = 0.3														
Palu	40.00	27.00	0.00	4.00	0.00	0.00	1.75	1.00	8.40	4.30	37.67	12.00	24.00	12.32
CommonKV	11.00	0.00	0.00	2.00	0.00	0.00	0.00	1.25	1.60	12.30	27.00	15.00	25.00	7.32
EchoKV	77.00	64.00	28.00	54.00	14.00	1.00	6.00	9.00	39.80	15.30	51.00	56.00	41.00	35.08
EchoKV-Hybrid	**99.00**	**80.00**	**88.00**	**76.00**	**53.00**	**10.00**	**48.00**	**50.75**	**80.00**	**27.10**	**82.33**	**61.00**	**45.00**	**61.55**
Mistral-Small-24B-Instruct-2501														
Compression Ratio = 0.5														
FullKV	98.00	100.0	100.0	95.00	98.00	94.00	99.75	99.00	100.0	95.90	92.00	83.00	63.00	93.67
ThinK*	0.00	1.00	1.00	1.00	0.00	0.00	2.75	2.50	0.40	12.80	25.33	59.00	46.00	11.68
Palu	98.00	**100.0**	68.00	**97.00**	84.00	76.00	94.50	96.25	71.20	34.20	80.00	55.00	58.00	77.86
CommonKV	99.00	99.00	98.00	98.00	92.00	90.00	94.50	**98.00**	89.60	71.10	**94.00**	81.00	**62.00**	89.71
EchoKV	99.00	**100.0**	**100.0**	95.00	**98.00**	**95.00**	85.00	96.25	**98.00**	75.20	92.33	83.00	60.00	90.52
EchoKV-Hybrid	**100.0**	**100.0**	**100.0**	94.00	97.00	94.00	97.50	96.75	97.20	**75.50**	92.00	**84.00**	60.00	**91.38**
Compression Ratio = 0.3														
Palu	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	4.00	4.00	0.62
CommonKV	4.00	36.00	4.00	6.00	1.00	0.00	7.00	0.75	0.00	5.60	32.33	30.00	39.00	12.74
EchoKV	**99.00**	**95.00**	**84.00**	**90.00**	**96.00**	**6.00**	**72.25**	**91.25**	**93.60**	**46.40**	**90.33**	**80.00**	**59.00**	**77.14**
EchoKV-Hybrid	5.00	4.00	1.00	3.00	7.00	0.00	0.00	2.75	9.80	17.20	61.00	61.00	47.00	16.83

peak-memory threshold during prefill, then continues with EchoKV. Appendix C.5 reports routing performance and switching quality.

C Additional results

C.1 Results on RULER

In this section, we present the detailed experimental results on the RULER benchmark. Table 9 summarizes the performance of Llama-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Mistral-Small-24B-Instruct-2501 across varying compression ratios (0.5 and 0.3).

C.2 Comparison with TransMLA

We further compare EchoKV with TransMLA [21], a post-training method that converts existing GQA attention modules into an MLA-style architecture. For fairness, we report two TransMLA settings on LongBench: direct conversion without additional adaptation ("Converted"), and a continued-training variant using the same LongAlpaca data as EchoKV ("FT"). For both Llama-3.1-8B and Mistral-7B, we use the same conversion parameters: at a compression ratio of 0.5, we set `qk-mqa-dim` to 128 and

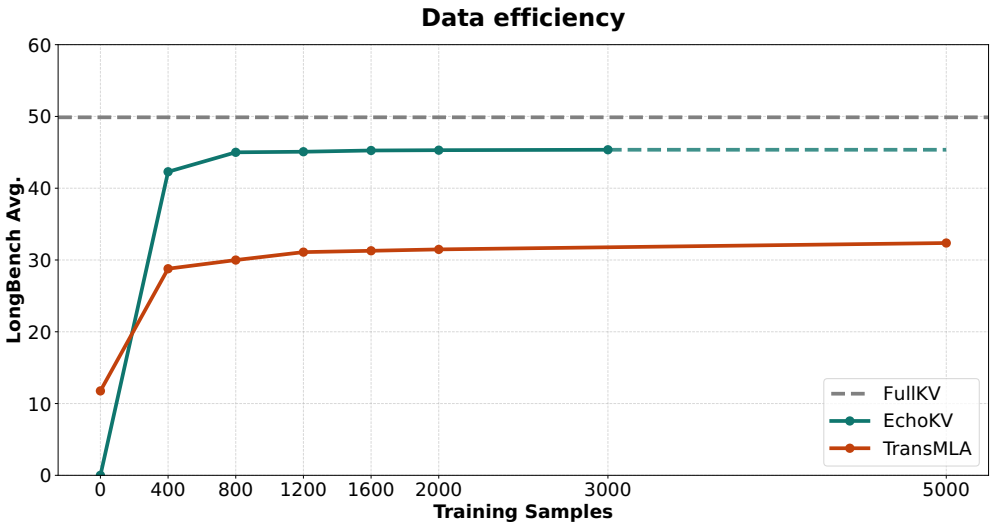


Figure 4: LongBench average as a function of training data size on Llama-3.1-8B at a compression ratio of 0.3. EchoKV reaches strong performance with substantially fewer training samples than TransMLA.

`kv-lora-rank` to 896; at a compression ratio of 0.3, we set `qk-mqa-dim` to 128 and `kv-lora-rank` to 512.

Table 3 in the main text summarizes the average LongBench results. Direct conversion leads to severe quality degradation on both backbones, while continued training substantially recovers performance. However, even after this additional adaptation, EchoKV still maintains a clear advantage across both compression ratios on Llama-3.1-8B and Mistral-7B.

Figure 4 further compares data efficiency on Llama-3.1-8B at a compression ratio of 0.3. EchoKV improves rapidly with a small amount of data and saturates early, whereas TransMLA starts from a much weaker converted model and requires substantially more training data to recover. This comparison highlights a practical distinction between the two approaches: TransMLA must first restore a modified attention module after architecture conversion, whereas EchoKV preserves the original attention projections and trains only a lightweight reconstruction module.

C.3 Hybrid results

We additionally report the results of EchoKV-Hybrid in the appendix, since this variant is intended as an exploratory extension rather than the main method. The motivation comes from the asymmetric reconstruction difficulty analyzed in §5.2: Values are generally easier to reconstruct with EchoKV, whereas Keys are more sensitive to prediction errors. Combined with prior observations that Keys are often more amenable to low-rank compression [6], this suggests a heterogeneous design in which Keys and Values need not share the same compression mechanism. Accordingly, EchoKV-Hybrid applies ThinK-style low-rank compression to Keys and EchoKV reconstruction to Values.

The results in Tables 9 and 10 support this motivation, but also reveal its limits. At moderate compression ratios, the hybrid design often provides additional gains over plain EchoKV, especially on several LongBench settings and on the RULER evaluations of Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3. These improvements are consistent with the hypothesis that the inter-cache similarity exploited by EchoKV is more pronounced for Values, while Keys can benefit more from direct low-rank compression. At the same time, the hybrid variant is not uniformly superior: under more aggressive compression, and particularly on the stronger 24B backbone, plain EchoKV is often the more robust choice. Taken together, these results suggest that differentiated treatment of Keys and Values is promising, but that the best hybrid strategy likely depends on both the compression regime and the backbone model.

Table 10: Experimental results on LongBench including the exploratory EchoKV-Hybrid variant. * Since ThinK only compresses Keys, we double the compression rate calculation to account for Value storage. We report results at 0.6 because the 0.5 ratio is infeasible. Compression Ratio = Size of Compressed Cache / Size of Full Cache.

		Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		
	Methods	**Nrtv**	**Qasp**	**MF**	**Hotpot**	**2Wiki**	**Musiq**	**Gov**	**QMS**	**MN**	**TREC**	**TQA**	**SAM**	**PC**	**PR**	**LCC**	**RB**	**Avg.**
Llama-3.1-8B-Instruct	FullKV	31.38	46.60	56.85	58.10	50.01	32.52	34.66	25.19	27.11	73.00	92.11	43.70	6.56	100.0	65.52	54.75	49.88
	Compression Ratio = 0.7																	
	MiniCache	17.87	22.26	29.51	26.72	11.47	9.71	24.61	21.85	23.29	65.00	75.05	31.46	5.83	60.50	41.34	37.19	31.48
	ThinK	**32.77**	45.44	**58.83**	**58.30**	**51.09**	**32.65**	32.63	24.96	26.19	**73.00**	91.72	42.10	**6.96**	99.50	**65.94**	55.64	**49.86**
	Palu	28.96	**46.73**	53.26	53.08	45.35	29.91	32.51	24.69	25.42	**73.00**	89.10	43.77	2.50	96.50	57.90	**56.78**	47.47
	CommonKV	29.87	41.16	53.86	54.90	45.42	29.96	27.91	24.21	25.68	66.00	90.03	43.10	6.67	**100.0**	61.92	53.41	47.13
	EchoKV	32.24	44.53	57.84	57.90	49.89	32.08	29.29	25.06	25.02	**73.00**	**92.52**	41.67	6.87	99.00	63.57	54.72	49.08
	EchoKV-Hybrid	32.04	45.97	58.15	58.14	50.16	32.42	**33.27**	**25.12**	**26.82**	**73.00**	92.28	**43.80**	6.27	97.00	65.12	55.13	49.67
	Compression Ratio = 0.5																	
	ThinK*	28.51	22.92	37.42	52.90	47.84	28.01	19.67	21.96	19.01	38.00	88.11	40.72	7.50	**99.50**	59.97	49.92	41.37
	Palu	19.27	34.64	40.14	41.99	26.71	22.45	22.60	23.70	23.21	66.00	25.95	39.75	4.00	49.50	29.86	30.19	31.25
	CommonKV	25.24	40.64	54.45	54.56	45.58	29.82	25.23	23.85	24.92	59.00	89.98	**42.50**	**10.12**	**99.50**	62.46	52.50	46.27
	EchoKV	**33.06**	44.80	57.51	57.91	**50.25**	31.79	29.22	24.10	25.21	**73.00**	**91.92**	41.45	7.17	89.50	64.61	54.91	48.53
	EchoKV-Hybrid	31.93	**45.44**	**57.68**	**58.53**	50.13	**32.69**	**31.81**	**24.81**	**25.83**	**73.00**	91.87	42.48	6.92	96.50	**65.74**	**55.11**	**49.40**
	Compression Ratio = 0.3																	
	Palu	1.90	2.42	4.36	1.00	1.70	0.66	3.52	9.28	5.95	1.25	3.19	6.31	0.00	0.00	18.20	18.79	4.91
	CommonKV	11.20	29.58	36.89	31.21	31.30	14.79	13.86	21.22	21.82	47.50	83.38	40.25	0.22	8.50	54.43	51.10	31.08
	EchoKV	**31.22**	**42.47**	**54.27**	57.03	**49.58**	**30.75**	**24.23**	**23.84**	**22.32**	49.00	**91.79**	40.90	**7.25**	84.50	62.09	**53.03**	45.27
	EchoKV-Hybrid	30.89	38.41	50.83	**57.28**	49.07	30.72	23.21	23.25	21.39	**57.50**	90.85	**41.77**	6.77	**95.50**	**62.77**	51.69	**45.74**
Mistral-7B-Instruct-v0.3	FullKV	29.80	39.12	50.45	50.28	36.31	26.45	34.11	25.93	26.56	76.00	88.89	46.97	5.50	97.00	61.47	62.57	47.34
	Compression Ratio = 0.7																	
	MiniCache	14.49	21.16	26.26	18.38	18.16	6.33	19.12	21.06	23.22	61.00	82.68	32.60	2.67	24.17	44.28	40.75	28.52
	ThinK	**30.42**	37.87	50.13	50.35	34.55	25.92	**33.81**	**25.66**	26.40	**76.00**	88.43	45.81	**5.50**	95.00	61.90	61.95	46.86
	Palu	28.47	36.08	**52.56**	46.57	**36.12**	26.37	33.59	25.16	**26.54**	73.50	87.77	45.11	4.00	94.50	59.04	61.06	46.03
	CommonKV	26.02	37.13	48.21	47.61	29.87	24.43	30.05	24.75	25.76	74.00	88.12	44.54	4.00	92.50	61.16	62.10	45.02
	EchoKV	29.93	37.65	49.43	**51.50**	34.54	24.98	32.29	25.11	25.99	**76.00**	**89.47**	**47.44**	**5.50**	**95.50**	**62.26**	62.18	46.86
	EchoKV-Hybrid	30.24	**38.90**	50.27	50.37	35.43	**26.73**	33.46	25.05	26.02	**76.00**	**89.47**	46.76	4.00	**95.50**	60.79	**62.51**	**46.97**
	Compression Ratio = 0.5																	
	ThinK*	29.27	27.37	39.90	45.95	31.83	20.41	22.24	23.21	19.80	65.50	87.27	43.43	**6.00**	80.00	59.03	59.32	41.28
	Palu	25.84	35.28	47.43	46.69	31.93	**27.21**	29.32	24.32	25.37	74.50	86.31	42.95	4.50	61.00	48.19	47.90	41.17
	CommonKV	27.27	34.71	46.70	44.74	27.02	24.21	26.36	23.52	24.42	51.00	88.56	43.82	4.50	88.00	60.14	59.94	42.18
	EchoKV	30.26	36.92	**49.71**	**51.33**	34.26	24.51	31.54	**25.28**	25.36	**76.00**	89.32	**46.93**	3.50	**93.50**	**61.72**	61.66	46.36
	EchoKV-Hybrid	**31.32**	**38.69**	49.55	49.05	**34.95**	25.10	**32.88**	25.02	**25.83**	**76.00**	**89.57**	46.55	3.50	93.00	61.00	**62.78**	**46.55**
	Compression Ratio = 0.3																	
	Palu	11.04	11.91	23.13	16.00	14.48	8.77	10.60	20.50	16.87	58.50	58.98	26.55	**4.50**	4.08	23.34	25.56	20.93
	CommonKV	11.96	21.25	29.05	21.47	14.49	8.40	22.30	22.55	**24.43**	56.50	72.81	37.21	2.63	14.50	52.46	39.16	28.20
	EchoKV	27.99	35.01	**47.29**	**49.77**	31.63	**24.01**	25.88	23.86	23.63	67.00	89.06	**45.00**	4.00	70.00	**59.27**	61.05	42.78
	EchoKV-Hybrid	**28.01**	**37.11**	47.08	47.98	**33.35**	23.63	**27.11**	**23.95**	22.93	**75.50**	**89.71**	44.41	**4.50**	**72.00**	58.96	**61.62**	**43.62**
Mistral-Small-24B-Instruct-2501	FullKV	33.72	47.19	54.95	67.24	64.93	49.04	33.22	24.54	25.47	75.50	93.71	48.86	21.00	100.0	63.67	70.66	54.61
	Compression Ratio = 0.7																	
	MiniCache	31.70	35.78	44.52	61.25	53.13	39.06	25.38	22.30	22.92	72.00	92.39	37.20	17.50	67.50	37.97	48.19	44.30
	ThinK	34.43	41.31	51.85	67.44	64.09	47.23	24.61	23.98	20.62	61.00	93.24	46.86	19.00	100.0	61.30	69.05	51.63
	Palu	34.71	**47.09**	55.35	66.61	63.78	44.51	**32.68**	**25.72**	25.53	74.50	92.12	48.22	14.50	100.0	54.65	65.63	52.85
	CommonKV	**36.16**	46.59	54.82	66.00	62.80	46.29	32.13	24.85	**25.55**	**76.00**	93.16	48.20	18.00	99.50	**68.02**	**71.11**	54.32
	EchoKV	33.74	46.73	**55.47**	67.18	64.07	**49.50**	31.71	25.28	24.92	75.50	93.64	48.22	19.50	100.0	64.38	70.98	54.43
	EchoKV-Hybrid	33.72	47.04	55.20	**67.46**	**64.89**	49.09	32.06	25.54	25.20	75.50	**93.71**	**48.55**	**21.00**	100.0	63.33	71.04	**54.58**
	Compression Ratio = 0.5																	
	ThinK*	26.96	30.79	35.81	56.43	57.52	39.26	19.81	21.56	16.84	37.00	91.51	44.48	17.50	99.00	56.30	61.10	44.49
	Palu	28.82	44.48	47.35	61.33	53.18	37.76	23.24	24.79	22.87	73.00	91.12	43.68	9.50	97.00	22.62	38.07	44.93
	CommonKV	**35.98**	46.14	54.49	66.59	63.45	47.10	30.78	24.58	**25.52**	74.00	92.91	47.59	19.00	100.0	**66.90**	**70.02**	54.07
	EchoKV	32.78	**46.33**	56.23	**67.14**	**64.25**	49.07	**30.88**	24.69	24.38	**75.50**	93.39	**48.37**	**20.00**	100.0	64.14	69.87	**54.19**
	EchoKV-Hybrid	33.62	45.36	**56.38**	65.84	63.51	**49.45**	29.68	**25.17**	23.70	**75.50**	**93.81**	48.29	**20.00**	100.0	63.45	69.72	53.97
	Compression Ratio = 0.3																	
	Palu	4.77	6.85	16.03	7.47	5.98	6.47	7.06	15.09	10.45	36.50	22.94	16.84	1.00	0.00	21.35	22.41	12.58
	CommonKV	27.41	41.87	48.97	55.45	53.02	28.62	26.16	22.74	**24.72**	72.50	90.17	45.51	1.23	95.00	**70.49**	68.78	48.29
	EchoKV	30.77	**44.80**	**54.43**	**66.78**	**62.74**	**49.01**	**27.57**	**24.66**	23.05	**74.50**	93.23	**45.76**	18.50	**99.50**	63.25	**69.95**	**53.03**
	EchoKV-Hybrid	**31.21**	34.22	43.76	62.49	58.14	41.53	20.42	22.87	17.58	41.00	**93.33**	45.41	**21.00**	99.00	58.13	62.71	47.05

C.4 Visualization of Needle-in-a-Haystack

To provide a more intuitive comparison of the retrieval capabilities of different methods under low memory budgets, we visualize the results of the "Needle-in-a-Haystack" (NIAH) task. Figure 6 and Figure 7 illustrate the performance of Palu, CommonKV, and EchoKV on Llama-3.1-8B-Instruct and Mistral-7B-Instruct-v0.3, respectively, at a compression ratio of 0.3. Green regions indicate successful retrieval, while yellow and red regions denote failures. As observed, EchoKV maintains consistently superior coverage across the context window on both models, demonstrating strong generalization capabilities. In contrast, while CommonKV performs relatively well on Llama-3.1, it suffers from severe information loss on Mistral-7B, further highlighting the robustness of our approach across different models.

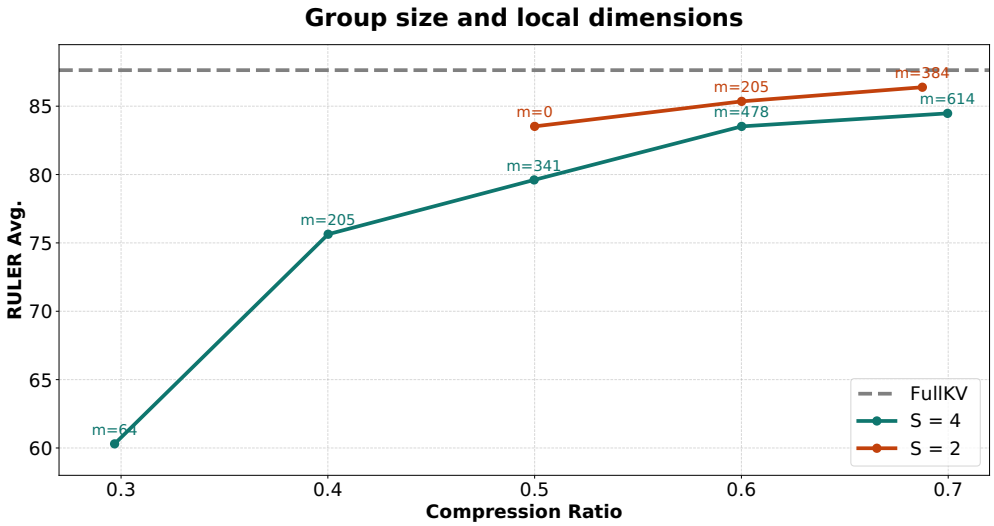


Figure 5: RULER average for different group sizes S and local dimensions m on Llama-3.1-8B-Instruct. The gray dashed line denotes the FullKV baseline, and each point is annotated with its retained local dimension m. Smaller S corresponds to a shorter prediction distance to the retained full layer.

C.5 Adaptive inference and switching

Table 11 evaluates the policies in Appendix B.4. Both adaptive policies complete all 20 batches while retaining FullKV-like TPOT for short prompts. The memory policy switches during prefill at layers 19/14 for 32K/36K, costing 193.52/165.52 ms; the length policy selects EchoKV before prefill.

Table 11: Adaptive workload on Llama-3.1-8B-Instruct, Pre-RoPE, nominal ratio 0.5, batch size 8, and 16 output tokens. Latencies are averages over five batches per length (20 batches total). Short/long prompts correspond to 4K/8K and 32K/36K. *The evaluated official Palu implementation stores reconstructed full-dimensional K/V in the standard attention cache.

Policy	Short TPOT (ms) 4K/8K	Long TPOT (ms) 32K/36K	Routing short / long	Switch (ms) 32K/36K	KV cache (GiB) short; long	Completed batches
FullKV-only	36.33 / 50.52	OOM / OOM	FullKV only	-	4.015 / 8.015; OOM / OOM	10/20
Echo static	74.27 / 129.65	469.54 / 528.20	EchoKV only	-	2.079 / 4.079; 16.079 / 18.079	20/20
Echo-adaptive (length)	35.00 / 50.16	472.67 / 530.62	FullKV / EchoKV	-	4.015 / 8.015; 16.079 / 18.079	20/20
Echo-adaptive (memory)	36.08 / 50.19	475.72 / 534.09	FullKV / FullKV→EchoKV	193.52 / 165.52	4.015 / 8.015; 16.079 / 18.079	20/20
Palu static*	43.71 / 51.47	OOM / OOM	Palu only	-	4.015 / 8.015; OOM / OOM	10/20

Table 12 reports quality after switching during generation. Switch@32/64 yields macro averages of 26.63–27.09, versus 26.18 for static EchoKV and 28.99 for FullKV; later switching retains more of the FullKV trajectory.

In the separate batch-1 Switch@32 test, converting 16K/32K caches takes 24.04/45.97 ms with Pre-RoPE and 3.21/3.53 ms with Post-RoPE.

Table 12: LongBench summarization results for Llama-3.1-8B-Instruct at ratio 0.5. Static EchoKV stores the compressed cache from prefill; Switch@t starts compression after t generated tokens.

Mode	GovReport	QMSum	MultiNews	Average
FullKV	34.66	25.19	27.11	28.99
Static EchoKV	29.22	24.10	25.21	26.18
Switch@32, Pre-RoPE	29.72	24.75	25.54	26.67
Switch@32, Post-RoPE	29.78	24.58	25.52	26.63
Switch@64, Pre-RoPE	30.34	25.27	25.67	27.09
Switch@64, Post-RoPE	30.16	25.07	25.71	26.98

C.6 Ultra-long context

We evaluate Llama-3.1-8B-Instruct at 64K and 128K on all 13 RULER tasks, using greedy decoding and macro averaging. EchoKV and Palu use ratio 0.5; ThinK uses an overall KV ratio of 0.6. As shown in Table 13, EchoKV exceeds Palu by 17.32/26.24 points at 64K/128K, with gaps to FullKV of 4.50/12.92 points. EchoKV completes the 128K evaluation, where ThinK runs out of memory.

Table 13: RULER macro averages at 64K and 128K on Llama-3.1-8B-Instruct. Ratios are shown in parentheses.

Context	FullKV	Palu (0.5)	ThinK (0.6)	EchoKV (0.5)
64K	84.13	62.31	10.77	**79.63**
128K	75.46	36.30	OOM	**62.54**

C.7 Seed robustness

We independently train Llama-3.1-8B-Instruct predictors with seeds 42–45 at ratios 0.7 and 0.5. Table 14 reports means and sample standard deviations over these four predictors; greedy evaluation is deterministic for a fixed predictor. The standard deviations of the LongBench and 32K RULER averages are at most 0.14 and 0.27, respectively. Individual LongBench categories vary more, particularly synthetic tasks at ratio 0.7.

Table 14: Predictor-training seed variation on Llama-3.1-8B-Instruct. EchoKV entries are mean $\pm$ sample standard deviation over seeds 42–45; FullKV is deterministic.

Metric	FullKV	EchoKV (0.7)	EchoKV (0.5)
Single-document QA	44.94	45.02 ± 0.15	44.81 ± 0.21
Multi-document QA	46.88	46.80 ± 0.12	46.75 ± 0.07
Summarization	28.99	27.39 ± 0.62	26.26 ± 0.12
Few-shot learning	69.60	69.20 ± 0.12	68.82 ± 0.17
Synthetic	53.28	50.02 ± 2.10	47.84 ± 0.35
Code	60.14	59.91 ± 0.54	59.83 ± 0.22
LongBench average	49.88	49.07 ± 0.14	48.45 ± 0.08
RULER average	87.63	86.27 ± 0.10	83.50 ± 0.27

D Additional analysis

D.1 Analysis of QK-KL divergence loss

In this section, we provide the detailed formulation of the standard KL divergence loss (QK-KL) referenced in §3.3, and analyze the computational bottlenecks that led us to adopt the Output MSE loss instead.

Formulation. The primary objective of the QK-KL loss is to ensure that the attention probability distribution generated by the reconstructed Key cache closely approximates the ground truth distribu-

tion. Given a query matrix $\mathbf{Q} \in \mathbb{R}^{L\times d}$ and the original Key matrix $\mathbf{K} \in \mathbb{R}^{L\times d}$, the standard attention weights $\mathbf{A}$ are computed via the softmax operation:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right), \tag{8}$$

where $\mathbf{A} \in \mathbb{R}^{L\times L}$ represents the full attention matrix, and d denotes the head dimension.

Similarly, utilizing the reconstructed Key matrix $\tilde{\mathbf{K}}$, which is composed of the retained local cache and the predicted dropped cache (as defined in Eq. 6), we obtain the approximated attention weights $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}} = \text{softmax}\left(\frac{\mathbf{Q}\tilde{\mathbf{K}}^T}{\sqrt{d}}\right). \tag{9}$$

The QK-KL loss is then defined as the Kullback-Leibler divergence between the target distribution $\mathbf{A}$ and the approximated distribution $\tilde{\mathbf{A}}$:

$$\begin{aligned}\mathcal{L}_{\text{KL}} &= \frac{1}{L}\sum_{i=1}^{L} D_{\text{KL}}(\mathbf{A}_i \parallel \tilde{\mathbf{A}}_i) \\ &= \frac{1}{L}\sum_{i=1}^{L}\sum_{j=1}^{L} \mathbf{A}_{i,j}\log\left(\frac{\mathbf{A}_{i,j}}{\tilde{\mathbf{A}}_{i,j}}\right),\end{aligned} \tag{10}$$

where $\mathbf{A}_i$ and $\tilde{\mathbf{A}}_i$ represent the attention distribution for the i-th token.

Complexity analysis. While $\mathcal{L}_{\text{KL}}$ effectively aligns the attention distributions, it imposes significant computational and memory overhead in long-context scenarios. Calculating the loss requires explicitly materializing the full $L \times L$ attention probability matrices $\mathbf{A}$ and $\tilde{\mathbf{A}}$. This results in a quadratic memory complexity of $O(L^2)$, which negates the memory efficiency benefits of sparse KV caching. Furthermore, this explicit materialization is incompatible with memory-efficient attention kernels such as FlashAttention [29], which avoid storing the intermediate attention matrix to achieve linear memory complexity. Consequently, we opt for the Output MSE loss (Eq. 7), which operates on the attention output $\mathbf{O} \in \mathbb{R}^{L\times d}$ and maintains compatibility with efficient kernels.

D.2 Group size, local dimension, and prediction distance

We further study the interaction between the group size S and the retained local dimension m.

Cache budget and construction. For a complete group of S layers, each token in the compressed region stores $D = H_{\text{KV}} d_{\text{head}}$ features at the anchor and m at each other layer. The nominal cache ratio is

$$r_{\text{nom}}(S, m) = \frac{D + (S-1)m}{SD}. \tag{11}$$

K and V share this ratio under the same configuration and precision. Anchor storage sets a lower bound of $1/S$. For a target $0 < r_{\text{target}} < 1$, we choose the smallest feasible group and allocate the remaining budget to local features:

$$S = \left\lceil \frac{1}{r_{\text{target}}} \right\rceil, \qquad m^* = \frac{D(Sr_{\text{target}} - 1)}{S-1}. \tag{12}$$

The continuous budget m^* is rounded to a supported dimension m; Appendix B.2 gives the configurations and resulting ratios. FullKV corresponds to ratio 1. Choosing a small S keeps the anchor close to its target layers, while m preserves same-layer information within the remaining budget. Appendix B.4 describes full-token storage and other memory components.

Empirical analysis. To examine this design choice, we evaluate multiple (S, m) combinations on Llama-3.1-8B-Instruct and report their RULER averages in Figure 5.

Two trends are clear. First, under comparable compression ratios, smaller group sizes tend to perform better. For example, around a compression ratio of 0.5, the $S = 2, m = 0$ setting outperforms the

Table 15: Comparison of local-dimension selection strategies on Llama-3.1-8B-Instruct. Prefix denotes the default contiguous selection used by EchoKV.

Ratio	Benchmark	Prefix	Random	Per-head	Oracle
0.7 ($S = 2, m = 384$)	LongBench	49.075	49.509	49.177	**49.553**
	RULER	**86.386**	84.765	82.805	85.440
0.3 ($S = 4, m = 64$)	LongBench	45.267	45.916	**46.038**	45.205
	RULER	**60.302**	48.446	43.125	60.159

Table 16: Comparison of prediction network architectures on Llama-3.1-8B-Instruct at a compression ratio of 0.5.

Architecture	LongBench Avg.	RULER Avg.
Linear	48.53	83.52
MLP (param-matched)	48.80	82.70
MLP (hidden = input dim)	**48.84**	**83.94**

nearby $S = 4, m = 341$ setting, and around 0.6, the $S = 2, m = 205$ setting also exceeds the corresponding $S = 4, m = 478$ setting. This suggests that reducing the prediction distance between the target layer and the retained full layer improves reconstruction quality. Second, when S is fixed, increasing m consistently improves performance, since more local information from the current layer is preserved. At the same time, S cannot be made arbitrarily small for all budgets: for instance, with $S = 2$, the lowest feasible compression ratio is 0.5 when no local dimensions are retained. Therefore, lower compression ratios require larger group sizes in order to satisfy the target memory budget.

This trade-off provides a simple interpretation of the (S, m) design space. The group size S controls how far the predictor must extrapolate across layers, while the local dimension m controls how much same-layer information is directly retained. Smaller S is therefore preferable whenever the memory budget allows it, and larger m should be used within a fixed S because it reduces the amount of information that must be reconstructed.

D.3 Local-dimension selection

We also compare different strategies for selecting the retained local dimensions. Besides the default prefix strategy that keeps the first m flattened dimensions, we evaluate three alternatives: random selection, a per-head allocation strategy that spreads the retained dimensions across heads, and an oracle-style greedy strategy based on similarity to the dropped dimensions. We report results on Llama-3.1-8B-Instruct under two representative settings: a moderate-budget configuration ($S = 2, m = 384$) with a compression ratio of 0.7, and a lower-budget configuration ($S = 4, m = 64$) with a compression ratio of 0.3.

Table 15 shows that the simple prefix strategy is already highly competitive. At the moderate-budget setting, more sophisticated strategies bring only marginal gains on LongBench and do not improve RULER, indicating that contiguous retention is sufficient when enough local information is preserved. At the more aggressive setting, prefix selection remains the most robust choice on RULER and is close to the best LongBench result. These results suggest that the main benefit comes from retaining a small amount of same-layer information, while the exact choice of dimensions is less critical than the overall budget. Given this small performance gap, the prefix strategy remains attractive because it avoids additional selection overhead and preserves contiguous memory access.

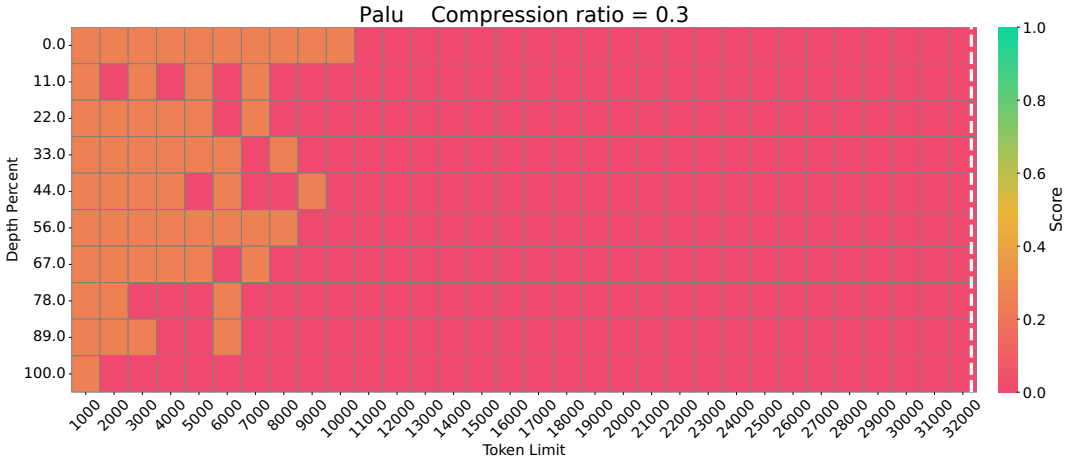
D.4 Prediction network architecture

We further examine whether a more expressive predictor architecture is beneficial. In addition to the default linear predictor, we evaluate two one-hidden-layer MLP variants on Llama-3.1-8B-Instruct at a compression ratio of 0.5: a parameter-matched MLP whose width is chosen to keep the parameter count comparable to the linear predictor, and a wider MLP whose hidden size equals the predictor input dimension.

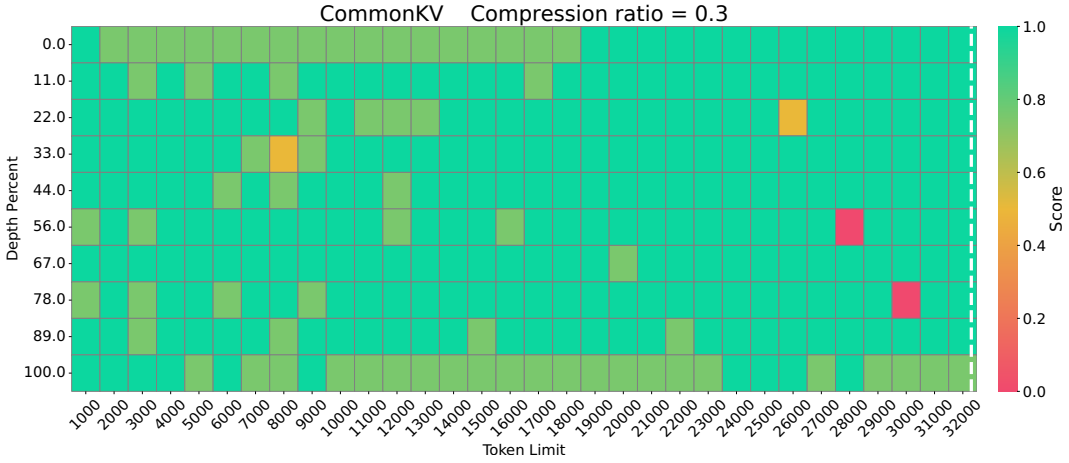
Table 16 shows that increasing predictor complexity yields only modest gains. On LongBench, both MLP variants remain within 0.31 average points of the linear baseline, and on RULER the best MLP improves the average by only 0.42 points. Moreover, the parameter-matched MLP does not consistently outperform the linear model, indicating that additional nonlinearity alone is not sufficient to deliver reliable gains. Given that larger predictors also introduce extra reconstruction overhead during inference, these results do not justify moving away from the simple linear architecture. We therefore retain the linear predictor in all main experiments.

E Use of AI tools

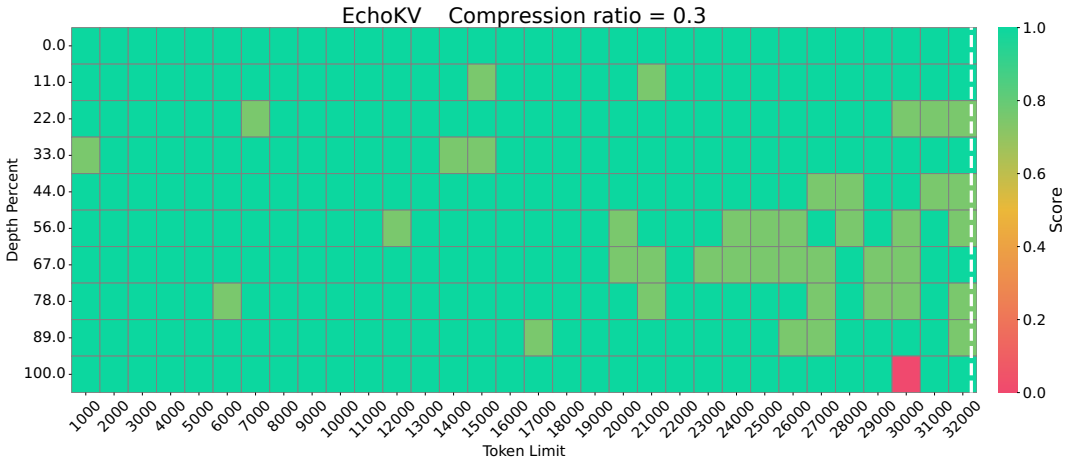
We acknowledge the use of AI tools solely for the purpose of language polishing and grammatical refinement of this manuscript. All core methodologies, experimental designs, and data analyses are conducted independently by the authors without the involvement of AI-generated content.



(a) Palu (Score = 4.3)

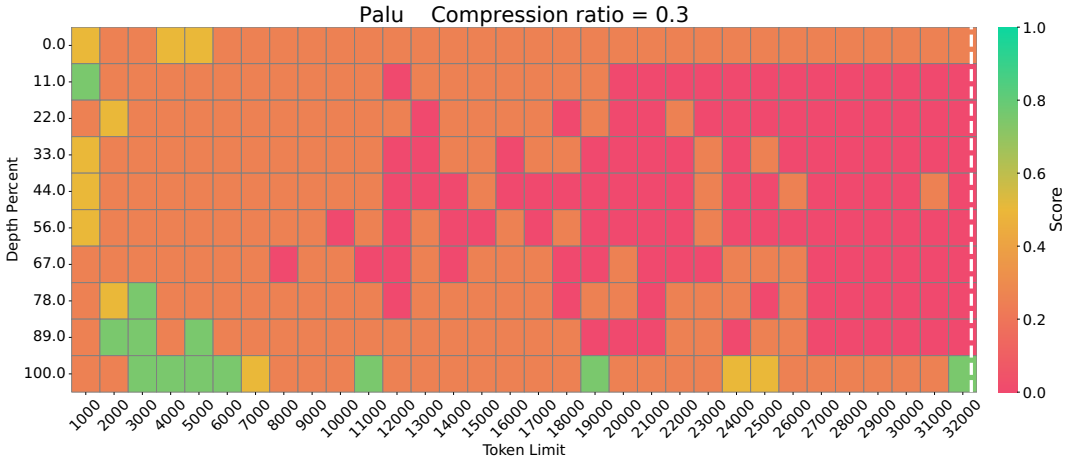


(b) CommonKV (Score = 94.0)

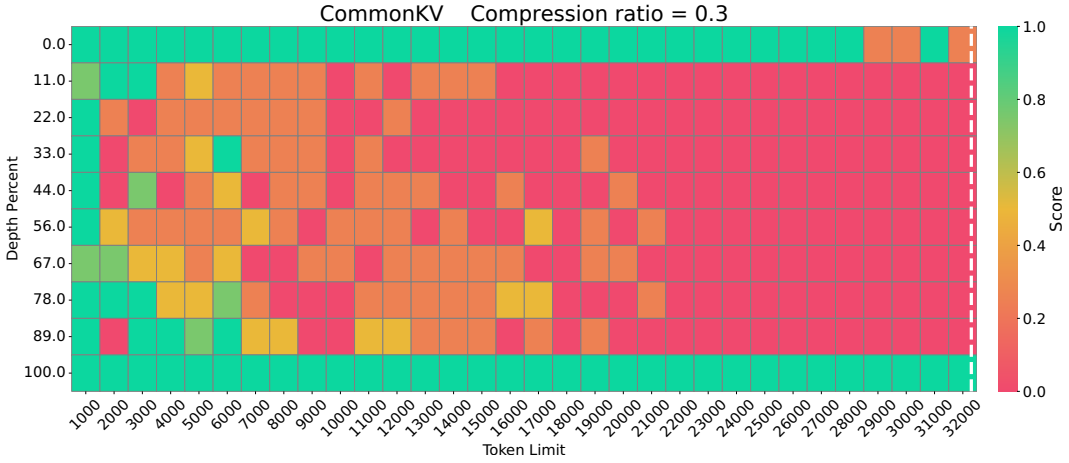


(c) EchoKV (Score = 96.6)

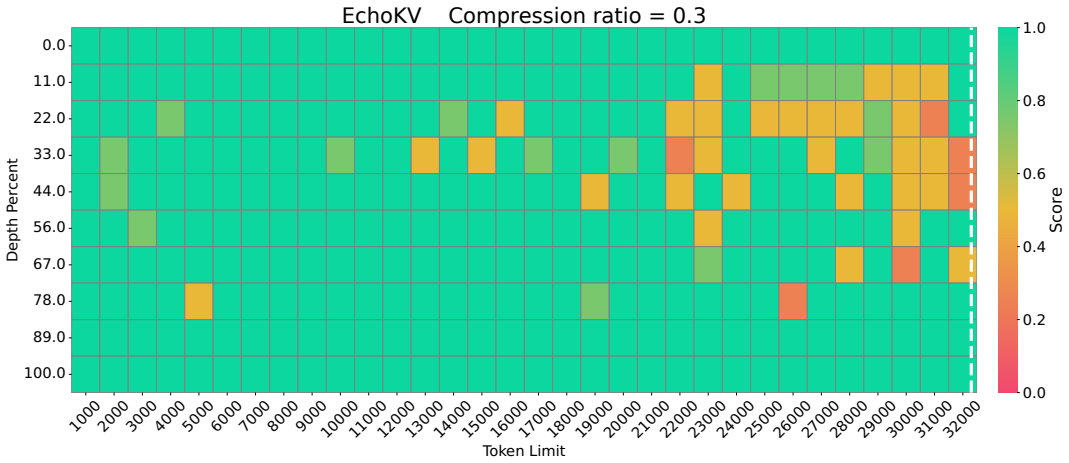
Figure 6: Visualization of NIAH results on Llama-3.1-8B-Instruct with a compression ratio of 0.3.



(a) Palu (Score = 18.9)



(b) CommonKV (Score = 32.8)



(c) EchoKV (Score = 92.8)

Figure 7: Visualization of NIAH results on Mistral-7B-Instruct-v0.3 with a compression ratio of 0.3.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes].

   Justification: The abstract and introduction describe the proposed EchoKV method, its flexible KV-cache compression setting, and its empirical gains on LongBench and RULER. These claims are supported by the experiments and analysis in the main paper.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes].

   Justification: The paper includes a dedicated "Discussion and Limitations" section in the appendix. It discusses the heuristic local-dimension selection strategy and the current limitations of the hybrid key-value compression design.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A].

Justification: The paper does not present theorem-style theoretical results or formal proofs. The mathematical content defines the method and training objectives.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes].

Justification: The paper specifies the backbone models, benchmarks, baselines, compression settings, predictor design, and training objectives used for the main results. Additional configuration and training details are provided in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes].

   Justification: The paper states that the code implementation is provided in the supplementary materials in anonymized form for submission. The manuscript and appendix describe the datasets, training setup, and evaluation protocol used to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes].

   Justification: The experimental setting section describes the benchmarks, models, baselines, compression ratios, and retained sink/local-window settings. The appendix further provides hyperparameters, optimizer, scheduler, training steps, batch size, and random seed.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes].

   Justification: Appendix C.7 reports means and sample standard deviations over predictor-training seeds 42–45 for Llama-3.1-8B at ratios 0.7 and 0.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes].

   Justification: The paper reports the approximate training cost of EchoKV on an NVIDIA A100-SXM4-80GB GPU and the throughput analysis specifies an NVIDIA A100-SXM4-80GB GPU. The appendix also provides the optimization settings and training-step counts used in the reported experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes].

   Justification: The work uses existing models and public benchmarks, without participant recruitment or new collection of personal data.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [N/A].

Justification: This is foundational research on KV-cache compression without a specific application or deployment. We identify no direct societal impacts specific to the method.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A].

Justification: The paper does not release a new pretrained language model, image generator, or scraped dataset. It proposes an inference-time compression method for existing models.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes].

Justification: The paper cites the existing models, datasets, and baseline methods used in the experiments.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes].

    Justification: The paper introduces the implementation of the proposed EchoKV method as a new research artifact and states that it is provided in the supplementary materials. The paper and appendix document the main settings needed to use and evaluate it.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A].

    Justification: The research does not involve crowdsourcing experiments or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

Justification: The research does not involve human subjects and therefore does not require IRB approval or an equivalent review.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A].

Justification: LLMs were not used as an original or non-standard component of the core methodology. The paper separately states that AI tools were used only for language polishing and grammatical refinement.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.