PrismKV: Harmonizing Heterogeneous Compression Strategies through Layer-adaptive Partition Scoring

Dongcheng Shi†
Beihang University
Beijing, China
shidongcheng@buaa.edu.cn

Yongxiang Cao†
Beihang University
Beijing, China
caoyongxiang@buaa.edu.cn

HongXu Jiang*
Beihang University
Beijing, China
jianghx@buaa.edu.cn

***Abstract*—Large Language Models (LLMs) encounter significant scalability bottlenecks in long-context scenarios due to the prohibitive memory overhead of the Key-Value (KV) cache. Current optimization paradigms often overlook the hierarchical heterogeneity of Transformer layers, leading to suboptimal resource allocation. To address this, we propose PrismKV, a novel KV cache compression framework that leverages adaptive partition scoring to orchestrate heterogeneous compression strategies. PrismKV employs an unsupervised, layer-adaptive scoring function that jointly evaluates compositional redundancy and attention sensitivity to dynamically bifurcate Transformer layers into distinct operational zones. For shallow regions characterized by high directional alignment, we implement a reconstructible codebook-based replacement strategy to achieve high-ratio structured compression by clustering redundant vectors. Conversely, for deep regions exhibiting sparse, task-specific attention, we transition to an importance-aware dynamic eviction policy that quantifies spatiotemporal saliency for precise information retention. Experimental results across three prevailing LLMs demonstrate that PrismKV reduces the KV cache to 10.8% of its original size with a marginal performance degradation of only 1.9%. Our approach consistently outperforms existing baselines in long-context benchmarks and achieves 65.8% decoding acceleration in 32K context scenarios. By precisely aligning heterogeneous compression primitives with the intrinsic evolution of layer-wise representations, PrismKV achieves a superior balance between hardware-level inference efficiency and high-fidelity generation.**

***Index Terms*—KV Cache Compression, layer-adaptive partition scoring, codebook-based replacement, importance-aware eviction.**

I. Introduction

Large Language Models (LLMs) offer exceptional versatility but are significantly hindered by the prohibitive memory and computational costs associated with long-context inference.A primary bottleneck is the KV cache, whose memory footprint scales linearly with sequence length and quadratically with model size. This scaling behavior frequently leads to GPU memory exhaustion, severely limiting the maximum supported context window.

Despite recent progress in KV cache optimization, existing strategies face several limitations. First, uniform eviction mechanisms [1]–[3] apply a monolithic strategy across all Transformer layers, disregarding inter-layer functional diversity. This often leads to semantic fragmentation in shallow layers and information loss in deeper layers, compromising performance under tight memory constraints. Although layer-aware methods [4], [5] attempt to address this, they typically rely on static, heuristic-based budgets that lack adaptability to diverse architectures and inputs. Consequently, they fail to fully exploit the inherent redundancy in shallow layers. Furthermore, even advanced replacement mechanisms like SpindleKV [6] utilize fixed partitioning, which fails to generalize across different models; for instance, a partition optimized for Llama-3 may cause under-compression or performance degradation on Mistral.

To address these limitations, we propose PrismKV, a heterogeneous KV cache compression framework that orchestrates layer-specific strategies through an unsupervised layer-adaptive scoring function. This function jointly quantifies compositional redundancy via off-diagonal cosine similarity and attention sensitivity through the standard deviation of attention weights. During the prefilling stage, PrismKV performs real-time scoring to dynamically bifurcate Transformer layers into shallow and deep regions; this is achieved in a parameter-free manner by detecting whether score deviations exceed intrinsic historical fluctuations. For shallow layers, a codebook-based replacement strategy is leveraged to cluster redundant vectors into a compact shared codebook, enabling high-ratio structured compression. In contrast, deep layers employ an importance-aware dynamic eviction policy that balances cumulative semantic significance with instantaneous sensitivity, maximizing information retention under stringent memory budgets.

The main contributions of this paper are as follows:

- We establish a heterogeneous compression architecture, tailoring reconstructible codebook-based replacement to exploit shallow-layer redundancy while deploying importance-aware eviction to navigate deep-layer saliency.
- We introduce a layer-adaptive scoring mechanism that jointly quantifies compositional redundancy and attention sensitivity to dynamically orchestrate the transition of compression strategies across different model regions.
- By embedding scoring mechanism and compression strategy switching into the prefilling forward pass, our approach eliminates reliance on generation-stage feedback and transforms adaptive compression from offline analysis into online inference.

†These authors contributed equally to this work.
*Corresponding author: HongXu Jiang (jianghx@buaa.edu.cn).

II. RELATED WORK

A. Token Eviction

Prevailing KV cache optimization primarily focuses on token eviction, utilizing heuristic-based criteria such as attention concentration [7] and positional importance [8] to prune redundant tokens. A conceptual comparison of the budget allocation policies employed by these methods is illustrated in Fig. 1.

Early heuristics like attention sinks [3], [9] prioritize initial and recent tokens, yet this bias potentially neglects critical semantic information within middle-sequence segments. Subsequent refinements have introduced sophisticated scoring metrics, including cumulative attention [1], [10], last-token salience [11], and clustering-based filtering [2]. Despite their precision, these methods predominantly employ monolithic budget allocation across all Transformer layers and fail to account for the heterogeneous importance inherent in different layers.

To transcend the limitations of uniform allocation, recent research has explored non-uniform budgeting. Static-pattern methods, such as PyramidInfer [4] and PyramidKV [5], adopt a power-law-inspired pyramid structure, favoring shallow layers with larger budgets. Conversely, D2O [12] adjusts capacity based on instantaneous attention density. Nevertheless, these strategies often rely on predefined heuristics or isolated layer-wise observations, which fail to capture the complex inter-layer dynamics and lack a robust quantitative foundation for cross-layer resource distribution.

To overcome these heuristic constraints, more advanced dynamic allocation frameworks [13]–[15] have emerged to optimize global budget distribution through layer-wise importance quantification. For instance, ZigZagKV [16] leverages uncertainty estimation to identify the minimum cache size required for target fidelity, while CAKE [17] evaluates spatiotemporal attention divergence to guide proportional allocation. By replacing fixed patterns with data-driven metrics, these methods provide the flexibility needed to respond to varying input characteristics. Despite their improved efficiency, these approaches tend to over-allocate resources to shallow layers, thereby under-utilizing the substantial compression potential embedded within these initial stages.

B. KV Cache Merging

KV cache merging [18]–[20] offers a compelling alternative to eviction by consolidating semantically similar KV vectors rather than discarding them. This paradigm is predicated on the observation of compositional redundancy, where KV vectors across different tokens exhibit high directional alignment as measured by cosine similarity and share a common semantic basis despite varying magnitudes. By employing clustering, weighted averaging, or codebook-based replacement, these methods compress redundant states while theoretically preserving broader contextual information.

However, existing merging techniques face critical bottlenecks. Locality-constrained methods, such as Cache Merging [21], utilize probabilistic sampling and learnable masks within fixed local windows. While this mitigates immediate information loss, the window-bound nature inhibits the modeling of long-range dependencies, as critical information cannot be transferred across window boundaries. In contrast, similarity-based clustering approaches like KVMerger [22] employ greedy algorithms and Gaussian kernel weighting to fuse representations. Although they preserve local semantic structures, their uniform application across all layers disregards the evolutionary nature of redundancy; furthermore, the cumulative approximation errors from kernel fusion often degrade attention precision during iterative decoding.

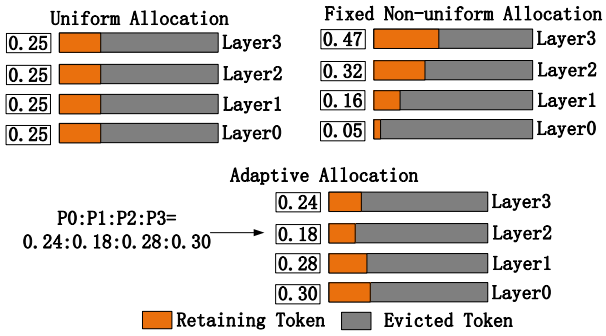

Fig. 1. Budget allocation strategies for various token eviction methods. Px represents the importance score of layer x.

To address inter-layer variability, SpindleKV [6] introduces a bifurcated strategy that applies codebook-based quantization to exploit compositional redundancy in shallow layers and transitions to sparse eviction in deeper and more discriminative layers. Nevertheless, its reliance on static and empirical partitioning limits its efficacy. By designating shallow layers through fixed heuristics, SpindleKV lacks input adaptivity and architectural awareness. Such rigid boundaries fail to accommodate varying redundancy patterns including those found in repetitive prose versus structured code, which leads to either sub-optimal compression or semantic distortion.

In summary, while KV cache merging reduces memory usage, current frameworks remain constrained by a lack of dynamic layer-awareness and content-sensitive adaptability, necessitating a more robust, online mechanism for strategy modulation.

III. METHOD

A. Partitioned Architecture Integrating Codebook Replacement and Dynamic Eviction

LLMs exhibit distinct structural behaviors across Transformer layers [17]. Shallow layers primarily capture low-level constitutive features characterized by high compositional redundancy where KV vectors share a common semantic basis. Conversely, deep layers specialize in high-level task-specific semantics and manifest highly sparse attention along with pronounced token-wise importance variance. This hierarchical heterogeneity renders a monolithic compression strategy suboptimal. While token eviction effectively prunes redundant

states in deep layers without compromising the semantic backbone, its application in shallow layers where attention is more dispersed often triggers the erroneous removal of structurally critical tokens.

To reconcile these divergent behaviors, we propose PrismKV, a framework that harmonizes heterogeneous compression strategies as illustrated in Fig. 2. By adaptively segmenting the model hierarchy, PrismKV tailors its compression mechanisms to the varying information density of the Transformer layers. In the shallow regions, we apply reconstructible codebook-based replacement to exploit high compositional redundancy. This process clusters directionally aligned KV vectors into a compact shared codebook, decoupling directional semantics from original magnitudes to facilitate high-fidelity reconstruction. As the model transitions to deeper layers characterized by sparse attention patterns, we switch to an importance-aware dynamic eviction policy governed by cascaded budget allocation. This strategy quantifies layer-wise saliency across spatiotemporal dimensions to prioritize tokens with the highest generative impact. Through this approach, PrismKV systematically aligns structural reconstruction and saliency-aware pruning with the specific redundancy profiles encountered across the Transformer hierarchy.

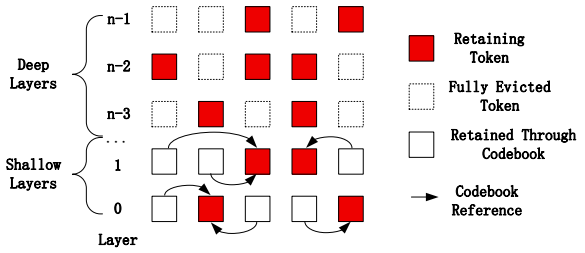

Fig. 2. Layer-differentiated compression:codebook-based replacement for shallow layers and token eviction with dynamic allocation for deep layers.

1) Importance-Aware Dynamic Eviction for Deep Layers: In deep-layer regions, we implement an inter-layer differentiated eviction strategy that dynamically modulates KV cache budgets according to layer-wise sensitivity, a design motivated by the observation that LLM layers exhibit profound spatiotemporal heterogeneity in attention patterns [17]. Specifically, layers characterized by high spatial dispersion or high temporal deviation are more susceptible to critical information loss under compression, meaning that a monolithic budget allocation inevitably leads to resource misallocation where sensitive layers are starved and redundant layers are over-provisioned. By tailoring the retention policy to the specific functional role of each layer, our approach ensures that the most semantically dense tokens are preserved even under strict memory constraints.

We define a layer importance score to quantify the true KV cache demand of each layer l as defined in (1). This score jointly evaluates spatial and temporal attention characteristics based on the attention weights $a_{t,s}^{(l)}$ observed during the prefilling stage where spatial dispersion is measured through attention entropy and temporal deviation is tracked over T query steps. Higher attention entropy indicates that a layer attends to a broader context of length S which necessitates a larger cache to preserve dispersed semantic signals. Simultaneously, high attention variance reflects tokens that are intermittently salient and measuring this temporal deviation prevents the premature eviction of tokens that may become critical during specific decoding stages.

$$P^{(l)} = \underbrace{\frac{1}{T}\sum_{t=1}^{T}\left(-\sum_{s=1}^{S} a_{t,s}^{(l)} \log a_{t,s}^{(l)}\right)}_{\text{Spatial Dispersion (Average Attention Entropy)}} + \underbrace{\frac{1}{S}\sum_{s=1}^{S}\left(\frac{1}{T}\sum_{t=1}^{T}\left(a_{t,s}^{(l)} - \frac{1}{T}\sum_{t'=1}^{T} a_{t',s}^{(l)}\right)^2\right)}_{\text{Temporal Deviation (Average Attention Variance)}} \tag{1}$$

To circumvent the $O(S \cdot L)$ peak memory bottleneck of conventional prefilling, we introduce a cascaded cache management mechanism, as shown in Fig. 3. Unlike static methods, PrismKV performs sequential layer-wise forward propagation. Upon completing the attention computation for the k-th layer, the total budget is proportionally reallocated among the first k active layers according to (2).The per-layer budget $B_l^{(k)}$ dynamically determines the number of tokens retained at the current stage. This iterative refinement ensures that as the number of processed layers increases, the cache configuration progressively converges toward a globally optimal distribution while maintaining a strictly bounded memory footprint.

$$B_l^{(k)} = B \cdot L \cdot \frac{P_l}{\sum_{i=1}^{k} P_i}, \quad \forall l \le k \tag{2}$$

To balance prefilling efficiency with generative fidelity, we utilize a query window mechanism where full attention computation is restricted to the last w tokens where the number of query steps T is strictly equal to w. These tokens are inherently immune to eviction due to their high local dependency while eviction for the remaining $S - w$ tokens is governed by a comprehensive token importance score defined in (3). This score integrates long term semantic contribution and dynamic stability by calculating the mean attention weight within the query window alongside the temporal variance of attention weights across the same window.

By incorporating temporal variance, our mechanism preserves "intermittently active" tokens that are frequently misidentified as redundant by simple averaging methods. Tokens are ranked by this composite score, with only the top-$B_l^{(k)}$ entries retained, thereby maximizing the preservation of high-value historical information under constrained budgets.

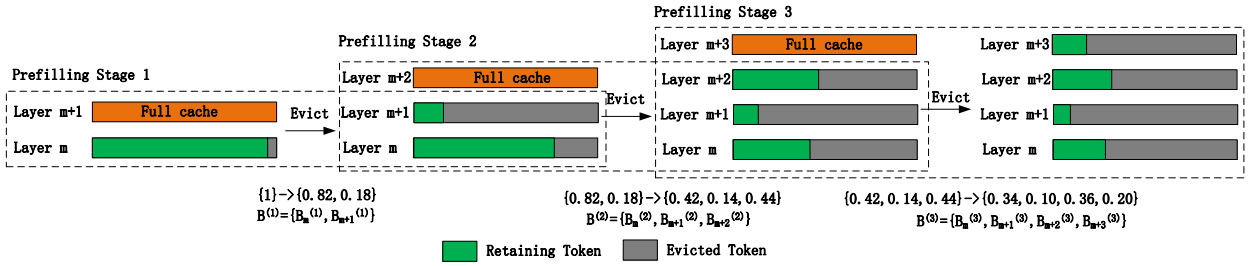

Fig. 3. Illustration of cascading budget allocation.

$$\mathbf{I}_l[n] = \bar{a}_n^{(l)} + \frac{1}{w}\sum_{t=S-w+1}^{S}\left(a_{t,n}^{(l)} - \bar{a}_n^{(l)}\right)^2, \qquad \bar{a}_n^{(l)} = \frac{1}{w}\sum_{t=S-w+1}^{S} a_{t,n}^{(l)} \tag{3}$$

2) Reconstructible Codebook-based Replacement for Shallow Layers: Shallow Transformer layers are characterized by high representational redundancy and diffuse attention, where KV vectors exhibit significant semantic alignment. To exploit this, we implement a codebook-based replacement strategy based on similarity clustering. This approach facilitates a Partitioned Representation in shallow layers through direction-amplitude decoupling. By partitioning the original high-dimensional KV space into a shared semantic codebook and individual magnitude scalars, we prevent redundant tokens from saturating the budget.

During the prefilling stage, we construct a shared codebook for both key and value vectors. We define an adjacency matrix $G \in \{0,1\}^{N\times N}$ to represent pairwise merging relationships among N tokens. For each layer l, we compute the cosine similarity between all vector pairs according to (4). A connection is established in G if the similarity exceeds an empirical threshold ($\tau_K = 0.98$ for key vectors, $\tau_V = 0.95$ for value vectors), effectively modeling the KV cache as an undirected graph where edges denote semantic redundancy.

$$\frac{\mathbf{k}_i^{(l)}\cdot\mathbf{k}_j^{(l)}}{\|\mathbf{k}_i^{(l)}\|_2\|\mathbf{k}_j^{(l)}\|_2} \geq \tau_K, \quad \frac{\mathbf{v}_i^{(l)}\cdot\mathbf{v}_j^{(l)}}{\|\mathbf{v}_i^{(l)}\|_2\|\mathbf{v}_j^{(l)}\|_2} \geq \tau_V \tag{4}$$

To minimize the codebook size while maintaining reconstruction fidelity, we identify a representative set of central nodes using a greedy dominating set strategy. This selection process begins by identifying the node with the highest degree as a codebook entry before removing both the selected central node and all its adjacent neighbors from the graph via a masking operation. We then iteratively refine the selection until all nodes are processed. Ultimately, each token is represented by a reference index to the codebook and its original magnitude, which ensures that the directional semantics are preserved with minimal storage overhead.

During decoding, approximate KV vectors are efficiently reconstructed by retrieving the indexed codebook entry and scaling it by the stored magnitude. For incremental tokens, the system first attempts to reuse existing codebook entries based on similarity thresholds; if no match is found, a local reconstruction process is triggered.

As shown in (5), this strategy reduces the memory footprint from $2Nd$ to $O((M_K + M_V)d + N)$, where $M \ll N$. However, the quadratic complexity $O(N^2 d)$ of the adjacency matrix construction poses a computational bottleneck. To maintain an optimal balance between inference throughput and compression ratio, we introduce an evaluation function to dynamically determine the transition point, ensuring this high-ratio compression is only applied to layers where the redundancy-to-overhead ratio is maximized.

$$\underbrace{(M_K + M_V)d}_{\text{Codebook}} + \underbrace{N(\log_2 M_K + \log_2 M_V)/32}_{\text{Reference (bit)}} + \underbrace{2N}_{\text{Magnitudes}} \tag{5}$$

B. Online Layer Partitioning via Adaptive Scoring

To effectively orchestrate the transition between codebook-based replacement and dynamic eviction, we propose an unsupervised layer-adaptive scoring function. Relying solely on compositional redundancy is insufficient; if a layer possesses high redundancy but concentrated attention, codebook-based averaging may blur critical signals. Therefore, our scoring mechanism introduces attention sensitivity as a necessary constraint to ensure optimal strategy selection.

1) Dual-Metric Quantification: The fitness of a layer l for structured compression is determined by two complementary signals:

1. **Compositional Redundancy ($\mathcal{S}_l$):** As defined in (6), this metric calculates the average pairwise cosine similarity of key and value vectors. A high $\mathcal{S}_l$ indicates that the layer's representation space is spanned by a few shared basis vectors, favoring codebook-based approximation.

$$\mathcal{S}_l = \frac{1}{S(S-1)}\sum_{i\neq j}^{S}\left(\frac{\mathbf{k}_i^{(l)}\cdot\mathbf{k}_j^{(l)}}{\|\mathbf{k}_i^{(l)}\|\|\mathbf{k}_j^{(l)}\|} + \frac{\mathbf{v}_i^{(l)}\cdot\mathbf{v}_j^{(l)}}{\|\mathbf{v}_i^{(l)}\|\|\mathbf{v}_j^{(l)}\|}\right) \tag{6}$$

2. **Attention Sensitivity (σ_l):** Defined in (7) as the standard deviation of average attention weights, this metric captures the "discriminative power" of a layer. A high σ_l reflects a concentrated attention pattern where specific tokens dominate, making the layer an ideal candidate for importance-aware eviction. Conversely, a low σ_l suggests a flat distribution where tokens are near-equally attended, rendering the model insensitive to individual token identities.

$$\sigma_l = \sqrt{\frac{1}{S-w}\sum_{n=1}^{S-w}\left(\bar{a}_n^{(l)} - \mu_{\bar{a}}^{(l)}\right)^2}, \mu_{\bar{a}}^{(l)} = \frac{1}{S-w}\sum_{n=1}^{S-w}\bar{a}_n^{(l)} \tag{7}$$

By fusing these metrics, we define the layer fitness score in (8) as the ratio Score_l. A high score identifies layers that are simultaneously redundant in representation and diffuse in attention, which represents the optimal regime for codebook-based replacement.

$$\text{Score}_l = \frac{s_l}{\sigma_l} \tag{8}$$

2) Online Dynamic Partitioning Algorithm: To enable real-time adaptation during inference, we implement an online dynamic partitioning algorithm based on statistical trend shifts. During the prefilling stage as the model performs sequential forward propagation, we monitor the evolution of the fitness score sequence where μ_{k-1} and σ_{k-1} denote the running mean and standard deviation of scores for the first $k-1$ layers respectively. Upon computing the k-th layer for $k \geq 2$, we assess the stability of the sequence to trigger the transition from shallow to deep layers whenever the inclusion of the current score Score_k causes a systematic drop in the mean that exceeds the historical volatility as formulated in the criterion of (9).

$$\mu_{k-1} - \mu_k > \sigma_{k-1} \tag{9}$$

This approach models the intrinsic representational transition of LLMs because shallow layer scores remain high and stable with fluctuations primarily driven by stochastic noise. As the network progresses to deeper layers where semantic abstraction increases and attention focuses on task-relevant tokens, the fitness score undergoes a precipitous decline. By detecting the specific juncture where the decrease in the mean significantly outpaces the historical standard deviation, PrismKV captures a statistically significant structural shift and enables the system to adaptively switch to dynamic eviction precisely when the representation becomes too sparse for clustering.

IV. EXPERIMENTS

A. Experimental Setup

To evaluate the generalizability of PrismKV, we conduct experiments on three representative backbone models including Llama2-7B-Chat [23], Llama3-8B-Instruct [24], and Mistral-7B-Instruct-v0.3 [25] which feature diverse architectural characteristics and maximum context windows ranging from 4K to 32K tokens. All experiments are executed on NVIDIA A100 80GB GPUs to ensure consistent computational performance. We compare PrismKV against five prevailing KV cache optimization methods categorized into two primary groups where the first group consists of Uniform Allocation strategies including StreamingLLM [3] which balances initial and recent tokens, H2O [1] which utilizes cumulative attention for eviction, TOVA [11] which adopts last-token attention, and SnapKV [2] which clusters recent attention to identify important tokens. The second group comprises Non-Uniform Allocation strategies such as PyramidKV [5] which employs a fixed pyramid-shaped budget distribution integrated with the SnapKV eviction indicator. Model performance is rigorously assessed using LongBench [26] which is a comprehensive benchmark for long-context understanding encompassing 16 knowledge-intensive subsets across six categories such as single-document and multi-document QA, summarization, few-shot learning, synthesis tasks, and code completion. These evaluation sequences predominantly range from 5k to 15k tokens to provide a robust measure of long-context inference capabilities under constrained memory environments.

B. Validation of Layer-Adaptive Scoring Function's Rationality

To evaluate our adaptive scoring mechanism, we compared it against a fixed boundary scheme designating the first three layers as shallow. Using average LongBench scores as the primary metric, Table I shows that the Adaptive approach consistently outperforms the Fixed baseline across all models. These results demonstrate a superior ability to navigate architectural heterogeneity by dynamically aligning compression with the specific representational evolution of each model.

Furthermore, we measured three models' average number of shallow layers calculated by adaptive partitioning strategy across the sixteen datasets in LongBench,as presented in the final column of Table I. These results indicate that the optimal compression boundary is highly model-specific. The adaptive strategy identifies an average of 5.8 shallow layers for Llama-3-8B-Instruct [24], suggesting that its refined normalization mechanisms allow the model to maintain high compositional redundancy up to layers 5-6. The 1.9% performance improvement over the fixed baseline proves that treating only the first 3 layers as shallow leaves excessive redundant patterns that introduce noise into the subsequent importance-ranking process. In contrast, Mistral-7B-v0.3 [25] yields an average of only 3.5 shallow layers, signifying that the model achieves semantic discriminativity as early as the fourth layer. In this case, the adaptive approach outperforms the fixed scheme by 3.1%, which confirms that forcing these discriminative layers into codebook compression leads to decline in accuracy due to the erasure of critical context fragments.

TABLE I
COMPARISON OF FIXED AND ADAPTIVE PARTITIONING STRATEGIES

Model	Avg Score on Longbench		Avg Num of Shallow Layers
	Fixed	Adaptive	
Llama-2-7b-Chat	32.67	32.99	4.2
Llama-3-8b-Instruct	47.24	48.12	5.8
Mistral-7b-Instruct-v0.3	41.38	42.67	3.5

C. Performance Evaluation on Longbench

PrismKV demonstrates superior performance on the LongBench benchmark, achieving the highest average scores across all three models compared to five prevailing KV cache optimization baselines as shown in Table II. Specifically, on Llama2-7B-Chat, our method improves the average score by 1.4% compared to the second-best performer PyramidKV while maintaining a minimal performance loss of only 0.2% relative to the FullKV baseline. For Llama3.1-8B-Instruct and Mistral-7B-Instruct-v0.3, PrismKV achieves score improvements of 0.4% and 1.2% over the runner-up SnapKV respectively. These gains are accomplished while restricting the performance gap to FullKV within just 1.9% and 1.2% for the two models. This advantage stems from the layer-differentiated approach to information preservation where PrismKV adaptively matches compression techniques to layer-wise characteristics to avoid the cumulative semantic erosion caused by premature token expulsion.

Aside from maintaining high accuracy, PrismKV achieves substantial memory efficiency as detailed in TableIII. Under a typical 8K context setting with a 1024-token deep-layer budget, our method achieves an additional 0.9% to 1.7% compression beyond the 12.5% baseline compression ratio. This improvement demonstrates that the structured compression of shallow representations in PrismKV reduces redundant storage overhead more effectively than token eviction. Furthermore, by optimizing the memory footprint at the representational level rather than through granular token removal, PrismKV maintains better semantic continuity within the cache. This cumulative reduction in memory consumption translates directly to increased batch sizes and higher inference throughput in resource-constrained environments. Consequently, PrismKV offers a more scalable solution for deploying large-scale models where both precision and hardware utilization are critical constraints.

Finally, to isolate the efficacy of our importance-aware dynamic eviction in deep layers, we conducted an ablation study on Mistral-7B-Instruct-v0.3 by fixing the first three layers as the shallow region and comparing the average LongBench scores of PrismKV against the token eviction methods of H2O and PyramidKV. The results in Table IV indicate that PrismKV achieves a score improvement of 7.9% compared to the H2O-based deep layer configuration and a 1.7% increase over the PyramidKV-based setup. These findings demonstrate that the token eviction strategy in PrismKV effectively enhances overall model performance by optimizing the allocation of cache resources across different layers.

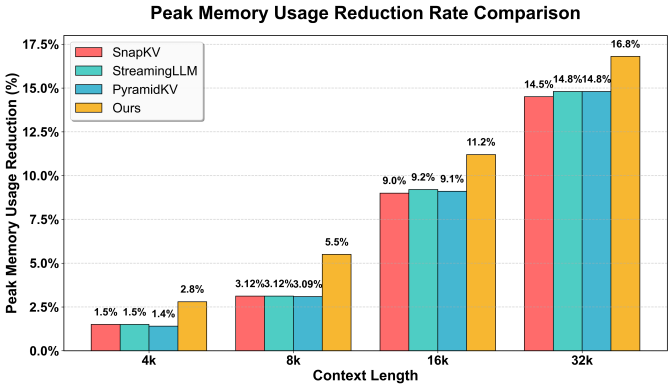

Fig. 4. Peak memory usage compression relative to FullKV across 4k-32k context lengths.

D. Memory Usage and Throughput Assessment

To evaluate the operational efficiency of the proposed framework, we conducted a rigorous analysis of peak memory consumption and decoding throughput using the Mistral-7B-Instruct-v0.3 model. Benchmarked against SnapKV, StreamingLLM, and PyramidKV under a standardized budget of 1024 tokens per layer, our approach consistently achieves the most substantial reductions in peak memory and the highest end-to-end acceleration.

As illustrated in Fig. 4, when the context length scales from 4K to 32K, the reduction in peak memory consumption compared to the baseline methods expands from 1.3% to 2.3%. These results demonstrate that the codebook-based replacement strategy in shallow layers effectively mitigates peak memory pressure by repartitioning the memory footprint into a compact set of basis vectors and scalar indices, thereby enhancing hardware utility for processing extended context windows under fixed resource constraints.

Furthermore, PrismKV achieves significant gains in inference throughput that scale effectively with sequence length, as illustrated in Fig. 5. In short-context scenarios of 4K tokens, our framework delivers an 18.2% increase in throughput, outperforming several baseline methods whose high computational complexity results in performance even lower than that of FullKV. This advantage becomes increasingly pronounced in 32K long-context scenarios, where the throughput improvement reaches 65.8%. These results provide empirical evidence that the cascaded architecture of PrismKV effectively reduces computational complexity. Unlike conventional methods that require per-layer importance ranking during the decoding stage, our cascaded mechanism allows shallow layers to operate in a static indexing mode post-prefilling. This strategy drastically minimizes bandwidth requirements and alleviates memory wall bottlenecks, thereby achieving superior hardware-level acceleration while maintaining high semantic fidelity.

TABLE II
PERFORMANCE COMPARISON OVER 16 DATASETS OF LONGBENCH(SINGLE-LAYER KV CACHE BUDGET=1024 TOKENS). THE BEST RESULT IS HIGHLIGHTED IN BOLD, THE SECOND BEST IN UNDERLINE.

Method	Single-Doc QA			Multi-Doc QA			Summarization			Few-shot			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	PCount	PR-en	Lcc	RB-P	
Llama2-7B-Chat																	
FullKV	20.68	20.19	36.69	27.83	31.45	8.21	27.09	20.71	26.24	64.0	83.09	41.41	2.94	7.75	58.51	52.29	33.07
StreamingLLM	13.73	16.75	26.28	25.97	25.67	5.87	21.92	20.47	24.07	61.00	80.88	40.90	2.28	1.00	56.71	48.79	29.46
H2O	16.58	17.67	32.04	26.01	28.73	7.81	22.87	20.99	25.05	60.00	83.02	40.06	2.90	3.50	57.57	52.02	31.05
TOVA	15.00	17.32	29.57	27.48	**31.71**	7.04	21.86	20.54	24.78	**63.50**	**83.35**	**41.78**	2.63	8.00	56.69	52.11	31.46
SnapKV	18.05	19.73	38.07	27.57	30.86	8.41	23.41	20.79	25.22	**63.50**	82.13	40.98	3.33	7.50	**58.20**	52.23	32.50
PyramidKV	17.94	20.68	**38.85**	27.25	29.99	7.91	23.17	**21.55**	25.45	**63.50**	82.97	40.79	**3.57**	7.50	57.72	51.93	32.55
PrismKV	**18.63**	**21.06**	38.47	**27.74**	31.15	**8.74**	**23.96**	21.02	**25.65**	**63.50**	83.16	41.15	3.08	**10.00**	57.71	**52.80**	**32.99**
Llama3.1-8B-Instruct																	
FullKV	31.06	45.43	53.78	55.04	47.14	31.29	34.87	25.33	27.49	72.5	91.65	43.81	6.00	99.50	63.36	56.65	49.06
StreamingLLM	26.64	30.77	35.59	47.31	42.03	24.17	25.81	21.31	25.66	63.50	88.84	42.76	**6.50**	88.00	61.36	53.47	42.73
H2O	29.57	36.15	45.94	54.43	44.81	29.04	27.64	23.31	**26.47**	62.00	91.83	43.14	6.36	99.00	62.74	55.39	46.11
TOVA	30.66	40.95	51.09	54.58	46.51	30.62	28.12	23.61	26.24	68.00	91.49	**43.80**	5.92	**99.50**	60.73	52.64	47.15
SnapKV	**30.95**	44.74	52.58	55.09	46.83	30.37	27.87	24.57	25.99	68.00	92.03	42.60	**6.50**	**99.50**	**63.00**	56.50	47.95
PyramidKV	30.54	43.64	**52.73**	55.29	46.29	**31.28**	27.53	24.50	26.00	68.00	**92.09**	41.75	6.05	**99.50**	62.35	55.44	47.69
PrismKV	30.92	**44.95**	52.44	**55.48**	**46.96**	30.84	**28.68**	**24.92**	26.35	**69.00**	91.94	42.60	6.00	**99.50**	62.65	**56.88**	**48.13**
Mistral-7B-Instruct-v0.3																	
FullKV	28.26	39.33	49.47	45.29	33.21	24.57	32.25	22.16	24.58	75.50	88.36	43.78	0.50	83.00	49.79	51.30	43.21
StreamingLLM	20.96	28.05	30.03	37.06	27.56	16.03	24.03	19.07	22.79	67.00	87.61	40.96	**1.50**	21.50	48.05	48.87	33.82
H2O	23.78	31.63	41.31	43.24	31.07	20.43	26.74	20.41	**23.93**	67.50	**88.84**	42.62	**1.50**	72.00	50.60	49.87	39.72
TOVA	**26.97**	34.51	45.58	44.32	32.58	22.83	26.91	20.75	23.49	**75.00**	88.66	**43.17**	1.00	91.00	47.51	47.06	41.96
SnapKV	26.63	35.78	48.11	45.75	32.20	23.37	26.71	**21.84**	23.18	70.50	88.61	41.37	0.50	88.00	50.60	**51.79**	42.18
PyramidKV	25.51	36.02	47.72	44.74	**33.16**	**23.91**	26.55	21.83	23.27	70.50	88.41	40.94	1.00	87.00	50.17	50.93	41.98
PrismKV	25.07	**36.51**	**48.75**	**46.47**	32.44	22.96	**27.56**	21.43	23.58	72.50	88.61	42.78	0.00	**91.50**	**51.10**	51.45	**42.67**

TABLE III
KV CACHE COMPRESSION RATIO OF DIFFERENT MODELS

Model	KV Cache Compression Ratio
Mistral-7B-Instruct-v0.3	11.6%
Llama-2-7b-Chat	11.2%
Llama-3-8b-Instruct	10.8%

TABLE IV
COMPARISON OF DIFFERENT DEEP LAYER COMPRESSION METHODS

Method	Avg Score on Longbench
H2O	39.53
PyramidKV	41.94
PrismKV	42.65

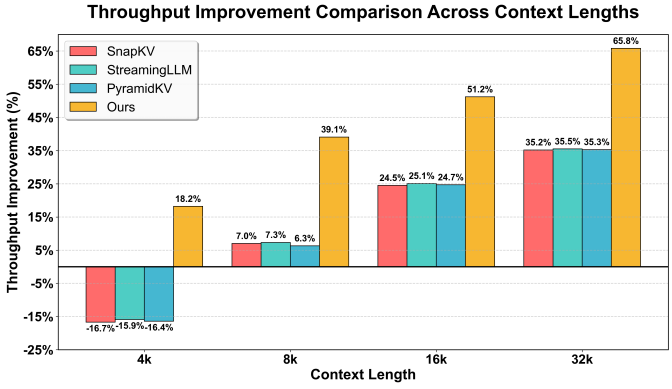

Fig. 5. Throughput improvement relative to FullKV across 4k-32k context lengths.

V. CONCLUSION

In this paper, we present PrismKV, which leverages an unsupervised, layer-adaptive scoring function to identify the hierarchical heterogeneity of Transformer architectures and dynamically partition models into distinct operational zones. Specifically, PrismKV applies codebook-based replacement in shallow layers to exploit compositional redundancy, while implementing importance-aware dynamic eviction in deep layers to manage sparse attention patterns. Experimental evaluations across several prevailing LLMs demonstrate that PrismKV

achieves a superior balance between inference efficiency and semantic fidelity. By providing a robust, model-agnostic solution for large-scale model deployment, this framework paves the way for more efficient and scalable long-context AI applications.

VI. ACKNOWLEDGMENT

This work is supported by the China Higher Education Institution Industry-University-Research Innovation Fund (No. 2024HY001).

REFERENCES

[1] 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, vol. 36, pp. 34661–34710, 2023.

[2] 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, vol. 37, pp. 22947–22970, 2024.

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

[4] Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao, "Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference," arXiv preprint arXiv:2405.12532, 2024.

[5] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, et al., "Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling," arXiv preprint arXiv:2406.02069, 2024.

[6] Zicong Tang, Shi Luohe, Zuchao Li, Baoyuan Qi, Liu Guoming, Lefei Zhang, and Ping Wang, "Spindlekv: A novel kv cache reduction method balancing both shallow and deep layers," in Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 28428–28442.

[7] Payman Behnam, Yaosheng Fu, Ritchie Zhao, Po-An Tsai, Zhiding Yu, and Alexey Tumanov, "Rocketkv: Accelerating long-context llm inference via two-stage kv cache compression," arXiv preprint arXiv:2502.14051, 2025.

[8] Junyoung Park, Dalton Jones, Matthew J Morse, Raghavv Goel, Mingu Lee, and Chris Lott, "Keydiff: Key similarity-based kv cache eviction for long-context llm inference in resource-constrained environments," arXiv preprint arXiv:2504.15364, 2025.

[9] Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang, "Lm-infinite: Zero-shot extreme length generalization for large language models," in Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), 2024, pp. 3991–4008.

[10] 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, vol. 36, pp. 52342–52364, 2023.

[11] Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz, "Transformers are multi-state rnns," arXiv preprint arXiv:2401.06104, 2024.

[12] Zhongwei Wan, Xinjian Wu, Yu Zhang, Yi Xin, Chaofan Tao, Zhihong Zhu, Xin Wang, Siqi Luo, Jing Xiong, and Mi Zhang, "D2o: Dynamic discriminative operations for efficient generative inference of large language models," arXiv preprint arXiv:2406.13035, 2024.

[13] Wei Wu, Zhuoshi Pan, Kun Fu, Chao Wang, Liyi Chen, Yunchu Bai, Tianfu Wang, Zheng Wang, and Hui Xiong, "Tokenselect: Efficient long-context inference and length extrapolation for llms via dynamic token-level kv cache selection," in Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, 2025, pp. 21275–21292.

[14] Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W Lee, Sangdoo Yun, and Hyun Oh Song, "Kvzip: Query-agnostic kv cache compression with context reconstruction," arXiv preprint arXiv:2505.23416, 2025.

[15] Kunxi Li, Zhonghua Jiang, Zhouzhou Shen, ZhaodeWang ZhaodeWang, Chengfei Lv, Shengyu Zhang, Fan Wu, and Fei Wu, "Madakv: Adaptive modality-perception kv cache eviction for efficient multimodal long-context inference," in Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2025, pp. 13306–13318.

[16] Meizhi Zhong, Xikai Liu, Chen Zhang, Yikun Lei, Yan Gao, Yao Hu, Kehai Chen, and Min Zhang, "Zigzagkv: Dynamic kv cache compression for long-context modeling based on layer uncertainty," in Proceedings of the 31st International Conference on Computational Linguistics, 2025, pp. 8897–8907.

[17] Ziran Qin, Yuchen Cao, Mingbao Lin, Wen Hu, Shixuan Fan, Ke Cheng, Weiyao Lin, and Jianguo Li, "Cake: Cascading and adaptive kv cache eviction with layer preferences," arXiv preprint arXiv:2503.12491, 2025.

[18] Guangda Liu, Chengwei Li, Jieru Zhao, Chenqi Zhang, and Minyi Guo, "Clusterkv: Manipulating llm kv cache in semantic space for recallable compression," in 2025 62nd ACM/IEEE Design Automation Conference (DAC). IEEE, 2025, pp. 1–7.

[19] 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, vol. 37, pp. 139997–140031, 2024.

[20] 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.

[21] Yuxin Zhang, Yuxuan Du, Gen Luo, Yunshan Zhong, Zhenyu Zhang, Shiwei Liu, and Rongrong Ji, "Cam: Cache merging for memory-efficient llms inference," in Forty-first international conference on machine learning, 2024.

[22] Zheng Wang, Boxiao Jin, Zhongzhi Yu, and Minjia Zhang, "Model tells you where to merge: Adaptive kv cache merging for llms on long-context tasks," arXiv preprint arXiv:2407.08454, 2024.

[23] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al., "Llama 2: Open foundation and fine-tuned chat models," arXiv preprint arXiv:2307.09288, 2023.

[24] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al., "The llama 3 herd of models," arXiv e-prints, pp. arXiv–2407, 2024.

[25] Devendra Singh Chaplot, "Albert q. jiang, alexandre sablayrolles, arthur mensch, chris bamford, devendra singh chaplot, diego de las casas, florian bressand, gianna lengyel, guillaume lample, lucile saulnier, lélio renard lavaud, marie-anne lachaux, pierre stock, teven le scao, thibaut lavril, thomas wang, timothée lacroix, william el sayed," arXiv preprint arXiv:2310.06825, vol. 3, 2023.

[26] 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), 2024, pp. 3119–3137.