DASH-KV: ACCELERATING LONG-CONTEXT LLM INFERENCE VIA ASYMMETRIC KV CACHE HASHING

Anonymous authors
Paper under double-blind review

ABSTRACT

The increasing memory footprint of the KV cache and the high computational cost of query-key similarity computation constitute two fundamental bottlenecks for large language models in long-context inference. While existing KV cache compression methods primarily alleviate the former, they often sacrifice generation quality and leave the latter largely unaddressed. This paper introduces DASH-KV, an innovative acceleration framework that reformulates attention as approximate nearest neighbor search via asymmetric deep hashing. Under this paradigm, we design an asymmetric encoding architecture that differentially maps queries and keys to account for their distinctions in precision and reuse characteristics. To balance efficiency and accuracy, we further introduce a dynamic mixed-precision mechanism that adaptively retains full-precision computation for critical tokens. Experiments on six LongBench tasks across three backbones show that DASH-KV achieves performance comparable to full attention. At a 128K context length, DASH-KV achieves a 2.732x speedup over pre-expanded SDPA for the complete single-layer decoding attention path on Qwen2-7B-Instruct.

1 INTRODUCTION

Large-scale pre-trained language models have demonstrated transformative capabilities in tasks including text generation, multi-turn dialogue, and code completion Achiam et al. (2023); Llama (2023). However, their deployment in production is confronted with significant efficiency challenges. Substantial evidence indicates Kwon et al. (2023); Pope et al. (2023) that the computational expense of the standard attention mechanism constitutes a primary performance bottleneck for large language model (LLM) services Zhou et al. (2024). This challenge is especially acute in scenarios demanding sustained context (e.g., multi-turn conversations, long-document generation) .

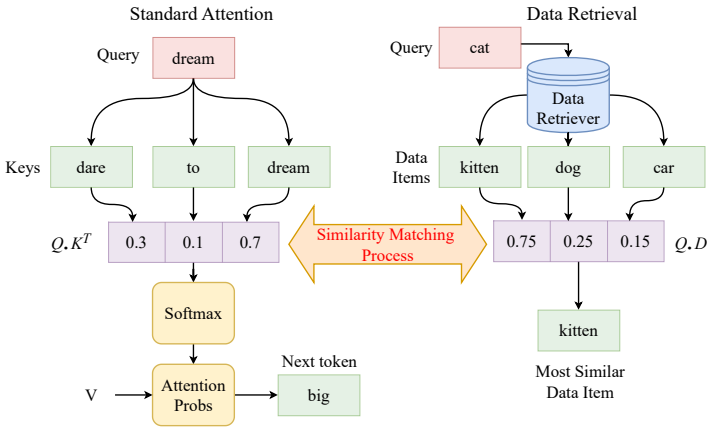

Figure 1: Alignment between Standard Attention Mechanism and Large-Scale Retrieval

To optimize long-text inference, existing research can be broadly categorized into three principal approaches: quantization compression, selective eviction, and structured sharing. Quantization meth-

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

ods Liu et al. (2024); Hooper et al. (2024); Zhao et al. (2024) reduce memory footprint by lowering the numerical precision of cached activations. However, employing low bit-widths often precipitates substantial degradation in model performance, especially in ultra-low bit settings (e.g., 1-bit or 2-bit). Furthermore, these techniques typically necessitate de-quantization during computation, incurring additional overhead. Selective eviction strategies Xiao et al. (2024); Zhang et al. (2023); Li et al. (2024); Ge et al. (2024) maintain a fixed-sized cache by retaining only a subset of KV pairs according to specific heuristics. While effective in constraining memory usage, this deterministic eviction results in the permanent loss of discarded information. Such irreversible information loss can impair model efficacy in tasks that rely on long-range dependencies. Structured sharing techniques Vyas et al. (2020); Ainslie et al. (2023); Kwon et al. (2023); Dao et al. (2022) aim to reduce redundancy by sharing KV parameters across different attention heads or layers. Nonetheless, this approach may not adequately account for the heterogeneous characteristics of different heads or layers and often lacks data-driven adaptability.

More critically, the above static compression strategies do not address the fundamental computational bottleneck of attention mechanisms Tay et al. (2022). The similarity matching process between queries and keys necessitates billions of high-precision floating-point operations Dao et al. (2022). Prevailing optimization methods operate strictly within this floating-point computational framework Xiao et al. (2023). Consequently, they leave the underlying computational paradigm unchanged.

Interestingly, the computation of relevance between queries and keys in the attention mechanism is highly similar to relevance-based matching in retrieval tasks. As illustrated in Figure 1, this perspective reveals a strong alignment between large-scale attention operations and established data retrieval paradigms. In the retrieval task, hashing Andoni & Razenshteyn (2015) serves as a prominent method for Approximate Nearest Neighbor (ANN) search, offering advantages in storage efficiency and query speed. Deep hashing Liu et al. (2012); Cao et al. (2017), in particular, employs neural networks to learn discriminative features and encodes them into compact binary hash codes, thereby enhancing efficiency and reducing computational cost Lai et al. (2015). Therefore, if this advanced technique from retrieval tasks could be applied to the highly similar attention computation scenarios, there would be significant opportunities for efficiency improvements while maintaining computational accuracy. Furthermore, KV caches have been shown to exhibit low-rank characteristics Sharma et al. (2024); Chen et al. (2021); Kitaev et al. (2020), suggesting their potential for compression in a latent space. Consequently, applying deep hashing techniques to the KV cache mechanism presents considerable promise.

In this paper, we propose **D**eep **A**symmetric KV Cache Ha**sh**ing (DASH-KV), which represents the first deep integration of deep hashing techniques into the Transformer attention mechanism to reconstruct the KV Cache computational paradigm. Specifically, our approach transforms the high-dimensional floating-point similarity calculation into a Hamming distance computation between compact binary hash codes. This constitutes a fundamental paradigm shift, replacing costly matrix multiplications with efficient bitwise operations. In addition, to balance precision and efficiency, we propose another two key modules:(1) **Attention-Oriented Asymmetric Deep Hashing.** Since Queries (Q) require higher precision to capture dynamic semantics, whereas Keys (K), which are extensively reused in the cache, prioritize encoding efficiency, we design an asymmetric hashing framework with differentiated encoding strategies. Specifically, for Q, we employ a deep hashing network (parameterized by a lightweight MLP) to maximize semantic preservation and ensure high-precision retrieval. Conversely, for K, we utilize a direct linear projection to facilitate rapid encoding and compact storage, thereby perfectly adapting to scenarios involving massive KV reuse. This differentiated mapping strategy effectively achieves a targeted synergy between computational efficiency and generation quality. (2) **Dynamic Importance-Based Mixed-Precision Attention.** Considering the varying importance of different tokens, we propose a lightweight importance predictor, which operates to differentiate between critical tokens (e.g., logical connectors and named entities) and regular tokens. For identified critical tokens, the system switches to full-precision attention. Moderately relevant tokens use hash-based approximation with residual correction, while weakly relevant tokens are masked for the current query. This adaptive paradigm enables fine-grained, instance-level control over the trade-off between efficiency and accuracy.

Experiments on LongBench confirm that DASH-KV outperforms SOTA baselines and achieves performance parity with Full Attention (e.g., on Qwen2-7B). Crucially, DASH-KV reduces the cost of attention computation through binary comparisons and selective floating-point computation.

In summary, the principal contributions of this work are as follows:

1. We propose the DASH-KV framework, which utilizes a novel asymmetric deep hashing paradigm. This architecture is designed to accommodate the distinct roles of queries and keys, enabling the construction of a semantic Hashing space for accelerated inference.
2. We propose an asymmetric deep hashing architecture to address the heterogeneous characteristics of queries and keys. Additionally, we introduce a dynamic mixed-precision mechanism to balance accuracy and efficiency.
3. DASH-KV achieves SOTA performance that matches Full Attention, breaking the efficiency-accuracy bottleneck.

2 Related Work

2.1 KV Cache Optimization for Long-Context Inference

Existing research on KV cache optimization mainly pursues three directions: quantization, selective eviction, and structured sharing. Quantization methods (e.g., KIVI, KVQuant, and Atom Liu et al. (2024); Hooper et al. (2024); Zhao et al. (2024)) reduce memory footprint by lowering numerical precision, but they incur accuracy degradation and de-quantization overhead. Selective eviction approaches (e.g., StreamingLLM, H2O, SnapKV, and Scissorhands Xiao et al. (2024); Zhang et al. (2023); Li et al. (2024); Liu et al. (2023)) maintain a fixed cache size, yet their eviction strategies lead to irreversible information loss. Structured sharing techniques (e.g., MQA and GQA Shazeer (2019); Ainslie et al. (2023)) aim to reduce parameter redundancy. However, they lack adaptability to the heterogeneous characteristics of different attention heads or layers.

Critically, these methods remain dependent on high-dimensional floating-point computation Kwon et al. (2023); Dao et al. (2022); Tang et al. (2024); Ribar et al. (2023), failing to address the fundamental computational complexity inherent in the attention mechanism.

2.2 Deep Hashing Methods

Hashing techniques map high-dimensional vectors to compact binary codes to enable efficient Approximate Nearest Neighbor (ANN) search. The field has progressed from early random projection methods, such as Locality-Sensitive Hashing (LSH) Andoni & Indyk (2008), to learning-based approaches like Iterative Quantization (ITQ) Guo et al. (2016); Weiss et al. (2008), and further to deep supervised hashing (e.g., DSH, HashNet, and DHN Liu et al. (2016); Guo et al. (2016); Zhu et al. (2016)) for improved retrieval accuracy.

Despite the success of hashing in large-scale retrieval tasks Wang et al. (2017); Liu et al. (2012); Cao et al. (2017); Lai et al. (2015); Zhu et al. (2016), its potential synergy with the Transformer attention mechanism has not been extensively investigated Kitaev et al. (2020); Jiang & Li (2018). To the best of our knowledge, this work presents the first systematic integration of hashing techniques for attention optimization. This is achieved by replacing computationally expensive floating-point similarity operations with efficient bitwise comparisons Rastegari et al. (2016), thereby accelerating autoregressive inference.

3 DASH-KV

3.1 KV Cache Retrieval via Deep Hashing

We posit that the attention mechanism fundamentally constitutes a process of large-scale vector similarity matching Kitaev et al. (2020); Tay et al. (2020). This process exhibits a strong conceptual alignment with core Information Retrieval (IR) tasks, as both are concerned with efficiently identifying the most relevant items from a vast candidate set. This observation motivates us to formally reframe attention computation as a retrieval problem. Consequently, we are inspired to substitute the computationally intensive floating-point dot products with efficient matching in a binary Hamming space. This reformulation establishes a novel and distinct pathway for optimization.

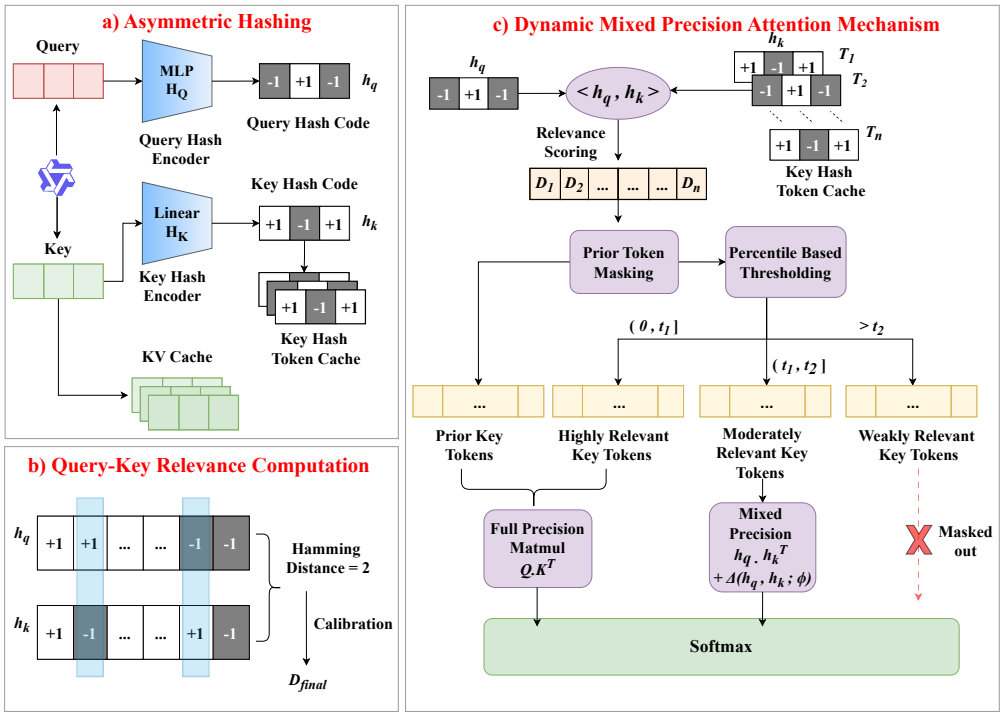

Figure 2: Overview of the **DASH-KV** framework. It comprises three key stages: (a) **Asymmetric Hashing**: Queries are dynamically encoded via a lightweight MLP, while Keys are statically mapped for efficient storage; (b) **Query-Key Relevance Computation**: A fast, coarse-grained filtering step using Hamming distance to select relevant candidates; (c) **Dynamic Mixed Precision Attention Mechanism**: A dynamic mechanism that keeps critical tokens in full precision, applies residual compensation to moderately relevant ones, and masks irrelevant keys.

Specifically, we pre-store a binary hash code $h_k \in \{-1, +1\}^l$ per Key and, for each Query, compute its hash code h_q. Relevance is scored via Hamming distance $\text{dist}_H(h_q, h_k) = \frac{1}{2}(l - \langle h_q, h_k \rangle)$, where $\langle \cdot, \cdot \rangle$ denotes inner product. To mitigate the bias of raw Hamming distances, we introduce two calibration strategies that exploit Transformer architectural priors. To enhance the accuracy of the preliminary retrieval stage, we exploit two inherent architectural properties of Transformers: the multi-headed attention mechanism and its deep layered structure. Based on these properties, we design a pair of complementary calibration strategies.

Cross-Head Consensus. We aggregate the selection preferences of each Key across all heads by counting how many heads rank it within a threshold T_{vote}: $\text{Vote}_i = \sum_{h'=1}^{H} \mathbb{I}(D_{raw}^{(l,h')}(i) < T_{vote})$. A high vote indicates cross-head agreement; we then apply a negative distance correction $\Delta_{spatial}(i) = -\beta_{spatial} \cdot \frac{\text{Vote}_i}{H}$.

Cross-Layer Momentum. We further incorporate a temporal prior by leveraging the previous layer's attention distribution: $\Delta_{temporal}(i) = -\gamma_{temporal} \cdot \sigma(A_i^{(l-1)})$, giving priority to persistently attended Keys.

Combining the above two calibration strategies, the final calibrated distance is:

$$D_{final}(i) = D_{raw}(i) + \Delta_{spatial}(i) + \Delta_{temporal}(i). \tag{1}$$

By setting $\beta_{spatial}$ and $\gamma_{temporal}$ as learnable parameters, the model can adaptively fuse spatial (multi-head) and temporal (multi-layer) structural priors. This significantly enhances the discriminative power of the hash distance.

3.2 ASYMMETRIC HASHING

We propose an Asymmetric Hashing Framework with distinct queries and keys encoders.

The **Query Encoder** maps evolving query vectors into the hash space while balancing semantic capacity and latency. We adopt a lightweight three-layer Multi-Layer Perceptron (MLP), which reduces computational overhead while retaining sufficient expressive power:

$$h_q^{(1)} = \text{GELU}(\text{LayerNorm}(W_1 Q)), \tag{2}$$

$$h_q^{(2)} = \text{GELU}(W_2 h_q^{(1)}), \tag{3}$$

$$v_q = W_3 h_q^{(2)} \in \mathbb{R}^l, \tag{4}$$

where $W_1 \in \mathbb{R}^{d\times 256}$, $W_2 \in \mathbb{R}^{256\times 256}$, and $W_3 \in \mathbb{R}^{256\times l}$ are learnable parameters.

Direct binarization of v_q with the sign function causes vanishing gradients, preventing end-to-end training. We therefore use Scaled tanh Cao et al. (2017) as a differentiable surrogate during training:

$$\tilde{h}_q = \tanh(\beta \cdot v_q), \tag{5}$$

where β is a scaling coefficient. We adopt a dynamic annealing schedule for β that gradually increases during training: $\beta = \min(10.0, 1.0 + \text{global_step} \times 0.001)$.

During inference, we directly apply the sign function to obtain the final binary hash code:

$$h_q = \text{sign}(v_q) \in \{-1, +1\}^l. \tag{6}$$

Key Encoder. Unlike queries, keys are persistent and reusable once stored in the KV cache, avoiding repeated encoding. We therefore use a simple linear projection to map key vectors into the target hash space with minimal latency:

$$z_k = W_k K, \tag{7}$$

$$h_k = \text{sign}(z_k) \in \{-1, +1\}^l, \tag{8}$$

where $W_k \in \mathbb{R}^{d\times l}$ is a learnable weight matrix.

Thus, the encoding pipeline, which maps raw vectors to compact hash codes and incorporates multi-view calibration for the initial distance estimates, is hereby established.

3.3 DYNAMIC IMPORTANCE-BASED MIXED-PRECISION ATTENTION

We dynamically allocate precision based on token importance. A predefined set $\mathcal{I}_{prior}$ (e.g., [CLS]/[SEP], sink tokens, and local neighbors) always receives full attention. The remaining keys are partitioned into three tiers via adaptive percentile thresholds on D_{final} (computed per sequence): (1) $D \le t_1$: full-precision; (2) $t_1 < D \le t_2$: hash + residual; (3) $D > t_2$: masked out.

Since fixed thresholds cannot accommodate distributional shifts induced by sequence length, we adopt an adaptive percentile-based strategy. Specifically, two truncation thresholds are computed in real time from the distribution of D_{final} within the current sequence as $t_1 = \text{percentile}(D_{final}, p_1)$ and $t_2 = \text{percentile}(D_{final}, p_2)$.

Based on this, we classify Keys into three levels:

1. **Highly Relevant** ($D \le t_1$): Retain full-precision computation $Q \cdot K^T$ to ensure core generation quality is uncompromised.
2. **Moderately Relevant** ($t_1 < D \le t_2$): Adopt a "Hash + Residual" hybrid scheme to correct quantization errors while balancing precision and efficiency Jegou et al. (2010).
3. **Weakly Relevant** ($D > t_2$): Exclude from computation without discarding, maximizing acceleration gains.

Moderately relevant Keys are central to our mixed-precision mechanism. Although they contain non-redundant information, they do not justify full-precision computation. We therefore first obtain a fast hash-based approximation and then refine it by compensating for quantization errors with a

lightweight neural network. Specifically, given hash codes $h_q, h_k \in \{-1, +1\}$ for Query and Key, we train a residual function $\Delta(h_q, h_k; \phi)$ such that:

$$\text{Attention}(Q, K, V) \approx \text{softmax}(\mathcal{S})V, \tag{9}$$

where the approximate score matrix $\mathcal{S}$ is defined as:

$$\mathcal{S} = \frac{h_q h_k^T}{d_h} + \Delta(h_q, h_k; \phi). \tag{10}$$

Here, $h_q \cdot h_k^T$ provides an efficient rough estimate, while Δ fits the residual between the hash approximation distribution and the true distribution with d_h denoting the hash-code dimension. We further apply a Zero Initialization Strategy to the MLP output layer Goyal et al. (2017).

In summary, DASH-KV implements **fine-grained dynamic allocation**.

3.4 Loss Design

We train DASH-KV using list-wise distillation Hinton et al. (2015) to align the student hash-based attention distribution with the full-precision teacher distribution. To address the overly smooth distribution of normalized hash dot products, we introduce asymmetric temperature scaling and combine the distillation objective with bit-balance and quantization regularizers.(details in Appendix H)

4 Experiments

4.1 Experimental Setup

4.1.1 Backbone LLMs

Our experiments employ three representative, publicly available large language models for comparative evaluation: Qwen2-7B-Instruct, Llama-3.1-8B-Instruct, and Qwen2.5-14B-Instruct. These models encompass diverse architectural designs and parameter scales, enabling us to assess the generalizability of DASH-KV's algorithmic advantages across varied model configurations.

4.1.2 Tasks

We evaluate DASH-KV on six LongBench Bai et al. (2024) tasks: NarrativeQA, HotpotQA, Qasper, MultiNews, GovReport, and TriviaQA, covering single and multi-document understanding, retrieval, reasoning, factual extraction, and summarization. Together, these tasks provide a comprehensive assessment of generalization across diverse long-context scenarios.

4.1.3 Baselines

We compare DASH-KV with four representative long-context baselines: Full Attention, StreamingLLM Xiao et al. (2024), H2O Zhang et al. (2023), and SnapKV Li et al. (2024). Detailed descriptions are provided in Appendix D. We additionally compare with Quest Tang et al. (2024) under a matched query-time access budget, and with PyramidKV Cai et al. (2024) and AdaKV Feng et al. (2026) under a matched persistent KV-byte budget.

4.1.4 Implementation details

We train on 3K-token sequences from the six source datasets, excluding LongBench evaluation samples and answers. The backbone is frozen, with layer-specific hash and residual parameters shared across heads. We use 128-bit codes. The first four and latest 64 tokens receive exact attention; among the rest, 5% use exact scores, 5% hash-plus-residual scores, and 90% are masked. LongBench uses contexts up to 32K. The matched comparison uses Qwen2-7B-Instruct in FP16 with a 31.5K input limit, identical evaluation settings, and KV-head-level GQA caches. Full-precision K/V remain cached; candidate access and KV storage are separate budgets. Quest uses page size 16. Further details are in Appendix I.

Table 1: Performance comparison of DASH-KV with baselines on LongBench tasks. We evaluate on three models: Qwen2-7B-Instruct, Llama-3.1-8B-Instruct, and Qwen2.5-14B-Instruct. NQ: NarrativeQA, HQ: HotpotQA, QM: Qasper, MN: MultiNews, GR: GovReport, TQ: TriviaQA. The Qwen2 DASH-KV row replaces Layer 18; see Table 5. The eight-layer comparison in Table 2 uses a separate evaluation configuration.

Method	**Type**	**NQ**	**HQ**	**QM**	**MN**	**GR**	**TQ**	**Avg**
				Qwen2-7B-Instruct				
Full Attn	Dense	25.13	44.04	46.13	15.42	18.06	83.47	38.71
StreamLLM	Eviction	20.47	14.31	26.97	24.88	25.70	76.56	31.48
H2O	Eviction	22.88	13.30	34.28	22.72	23.69	88.75	34.27
SnapKV	Retrieval	23.86	15.60	38.61	23.07	24.56	89.31	35.84
DASH-KV (Ours)	Retrieval	24.65	44.50	45.34	15.40	19.13	83.33	38.73
				Llama-3.1-8B-Instruct				
Full Attn	Dense	28.11	57.43	45.29	15.20	19.97	90.22	42.70
StreamLLM	Eviction	13.60	11.70	20.10	24.70	21.50	71.10	27.12
H2O	Eviction	23.10	16.00	21.30	23.40	22.30	90.20	32.72
SnapKV	Retrieval	21.30	16.60	30.80	26.30	22.20	90.20	34.57
DASH-KV (Ours)	Retrieval	27.61	58.01	45.35	14.90	19.25	89.47	42.43
				Qwen2.5-14B-Instruct				
Full Attn	Dense	28.20	61.98	45.52	14.19	16.78	86.91	42.26
StreamLLM	Eviction	18.30	16.40	23.40	16.04	19.01	74.10	27.87
H2O	Eviction	24.40	18.00	27.60	20.99	18.96	89.70	33.28
SnapKV	Retrieval	24.10	20.00	34.60	20.90	18.97	90.00	34.76
DASH-KV (Ours)	Retrieval	29.52	61.83	45.79	14.44	18.06	87.88	42.92

Table 2: Additional LongBench comparisons on Qwen2-7B-Instruct. Panel A retains full K/V; nominal access ratios apply only to the eight selected layers. Panel B budgets refer to persistent K/V bytes.

A. Full-history retrieval: matched eight-layer deployments			
Method	Candidate access	Persistent memory (MiB)	Avg.
FullKV	100%	1,722.66	39.48
Quest	10%	1,753.43	38.83
DASH-KV	10%	1,740.30	38.99
B. Cache compression: matched persistent KV-byte budgets			
Method	KV bytes relative to FullKV		Avg.
FullKV (reference)	100%		39.48
StreamingLLM	10%		31.34
H2O	10%		34.84
SnapKV	10%		35.50
PyramidKV	10%		38.27
AdaKV	10%		37.49

4.2 Main Results on LongBench Tasks

Table 1 reports six-task results for three backbones. DASH-KV obtains 38.73 versus 38.71 for Full Attention on Qwen2-7B, 42.43 versus 42.70 on Llama-3.1-8B, and 42.92 versus 42.26 on Qwen2.5-14B. These results describe their respective deployment configurations. The Qwen2 result replaces one layer; the additional eight-layer experiment below reports its own FullKV reference and is analyzed separately.

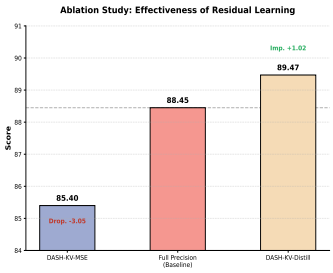

Figure 3: Ablation Study on TriviaQA - F1 Score

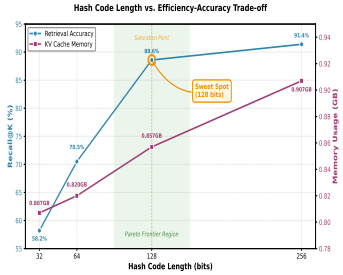

Figure 4: Pareto Frontier of Efficiency and Accuracy

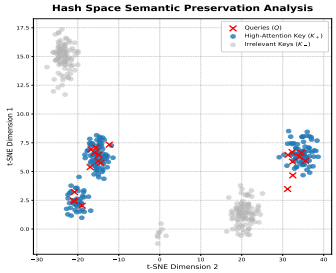

Figure 5: t-SNE Visualization of Hash Codes

In Table 2A, DASH-KV scores 38.99, compared with 38.83 for Quest and 39.48 for FullKV. It remains close to both references while using less auxiliary persistent memory than Quest. Persistent memory in Table 2A includes K/V, retrieval metadata, and auxiliary networks, excluding backbone weights and transient workspaces. Table 2B reports cache-compression methods at a common persistent KV-byte budget. DASH-KV retains full K/V, so the two panels do not constitute an equal-storage comparison across all methods. The asymmetric encoders and residual correction support attention approximation; their individual contributions are examined in the ablation studies.

4.3 Ablation Studies

We conduct a systematic ablation study to validate the contribution of each component within the DASH-KV framework. Specifically, we analyze the role of the Residual Compensation Strategy in mitigating errors introduced by hash quantization.

4.3.1 Effectiveness of Residual Learning

We compare Pure Hash, DASH-KV-MSE, and DASH-KV-Distill to assess residual learning. Figure 3 shows that Pure Hash suffers from quantization errors, while MSE residual regression fails to recover the loss and even underperforms Pure Hash, likely due to numerical drift under length extrapolation (Appendix J.1). In contrast, KL-based distillation effectively corrects these errors and restores near-full-attention performance, demonstrating its importance for long-context generalization. Across eight residual-network configurations, LongBench averages 38.66–38.73, showing limited sensitivity to network size (Appendix J.5).

4.3.2 Component Effectiveness

4.3.3 Parameter Sensitivity

We vary the hash code length b from 32 to 256 bits to evaluate its impact on retrieval precision and memory usage. The results in Figure 4 reveal that retrieval accuracy improves sharply as the code length increases from 32 to 64 bits, but the gains plateau beyond 128 bits, where recall approaches saturation. Meanwhile, KV cache memory usage grows linearly with code length. We therefore adopt 128 bits as the default configuration, which achieves near-optimal retrieval precision (88.6%) with a modest memory footprint of 0.857 GB. Further details are provided in Appendix J.3.

Table 3: Efficiency–accuracy trade-off. P_h and P_f denote hash-based and full-precision attention distributions.

Variant	Recall@100 (%)	$\mathbf{KL}(P_h \parallel P_f)$ (↓)	Latency/token (ms)
DASH-KV-Naive	8.91	3.2000	28
DASH-KV-Sym	68.28	1.3200	38
DASH-KV (Ours)	**86.06**	**0.4054**	**22**

Table 3 shows that asymmetric hashing outperforms the symmetric baseline, while LSH yields substantially lower Recall@100, demonstrating that learned, query/key-specific hash spaces better preserve attention-relevant semantic structure than generic geometric hashing. See Appendix J.2 for

details. A depth ablation further favors the three-layer query encoder, improving the LongBench average by 0.98 over two layers at only 0.027 ms additional layer latency (Appendix J.4).

5 Analysis

5.1 Semantic Preservation in Hamming Space

Figure 5 illustrates the semantic preservation capacity of the proposed hash encoding. Within the learned hash embedding space, semantically related Queries and Keys form compact clusters, indicating cohesive grouping. In contrast, irrelevant Keys are effectively dispersed, resulting in a distinct and significant spatial separation between dissimilar instances.

5.2 Scalability to Ultra-Long Context

Latency is measured on Qwen2-7B-Instruct (FP16, batch size 1, RTX 4090 48 GB) with 128-bit codes and 10% candidate retention. At 128K, DASH-KV takes 0.711 ms for the complete single-layer attention path (Table 4); component timings are in Appendix I.1. Key codes are built during prefill, with only new codes appended during decoding. DASH-KV achieves 2.732× speedup over pre-expanded SDPA and 8.021× when GQA K/V expansion is included, reaching 3.99× across 256K–1M contexts. These are attention-path measurements. Both methods retain linear per-step complexity in N, while DASH-KV reduces scan and floating-point costs. On three selected ∞Bench tasks, DASH-KV averages 32.65 versus 32.77 for Full Attention (Appendix L).

Table 4: Single-layer decoding latency at 128K with packed XOR/POPCNT and fused DASH-KV kernels. Both rows use the same complete DASH-KV path.

Baseline	Baseline (ms)	DASH-KV (ms)	Speedup
Stock GQA expansion + SDPA	5.700	0.711	8.021×
Pre-expanded SDPA	1.942	0.711	2.732×

5.3 Analysis of the Rationality of Asymmetric Hashing

Figures 6 reveal clear divergence between Query (Q) and Key (K) representations in both magnitude and distribution, indicating that they exhibit inherently different numerical and geometric characteristics. This discrepancy motivates our asymmetric hashing design, which adopts differentiated encoding strategies for Q and K. The observed performance gap further shows that symmetric encoding cannot adequately capture their distinct geometric manifolds and value ranges, thereby validating the necessity and effectiveness of the proposed asymmetric hashing architecture.

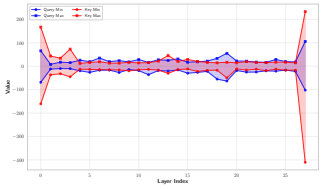

Figure 6: Magnitude divergence across layers

6 Conclusion

This paper presents DASH-KV, a novel inference acceleration framework leveraging deep hashing retrieval. It seeks to fundamentally address efficiency bottlenecks in long-context reasoning by reconfiguring the underlying computational paradigm. The framework reconstructs the attention mechanism into an approximate nearest neighbor search within Hashing space. This design effectively replaces costly floating-point matrix multiplications with efficient bitwise operations. Comprehensive experiments conducted across multiple long-text benchmarks indicate that DASH-KV attains a superior balance between efficiency and accuracy. To further realize the potential of the proposed hashing strategy, future work will explore extending the ANN-based attention paradigm to other scenarios where heterogeneous token representations naturally arise, such as cross-modal attention in vision-language models and efficient retrieval in retrieval augmented generation pipelines.

REPRODUCIBILITY STATEMENT

We facilitate reproducibility by documenting the DASH-KV architecture and its major algorithmic components, including the asymmetric query and key encoders, Hamming-space retrieval, dynamic mixed-precision routing, residual correction, and training objectives. The experimental setup specifies the publicly available backbone models and benchmarks, the separation of training and evaluation data, the hash-code length, routing ratios, context-length settings, evaluation baselines, and comparison protocols. The appendix further provides descriptions of the datasets and baselines, the theoretical connection between binary similarity and Hamming distance, the complete loss formulation, parameter and cache-memory accounting, training costs, and the hardware and profiling setup. We additionally report matched-budget comparisons, component-level latency measurements, architectural and hyperparameter ablations, layer-wise analyses, and evaluations on selected ∞Bench tasks. We also provide the implementation code as part of the supplementary material. Together, these resources are intended to support independent reimplementation and verification of the reported accuracy, memory, and latency results.

AI USE STATEMENT

In this work, generative AI tools were used solely to aid and polish the writing, including improving grammar, clarity, organization, and presentation. All AI-assisted text was reviewed and verified by the authors, who take full responsibility for the final content of the paper.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

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. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

Alexandr Andoni and Ilya Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 793–801, 2015.

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)*, pp. 3119–3137, 2024.

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.

Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S Yu. Hashnet: Deep learning to hash by continuation. In *2017 IEEE international conference on computer vision (ICCV)*, pp. 5609–5618. IEEE, 2017.

Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention. *Advances in Neural Information Processing Systems*, 34: 17413–17426, 2021.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in neural information processing systems*, 35:16344–16359, 2022.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *Advances in Neural Information Processing Systems*, 38:113152–113188, 2026.

Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you what to discard: Adaptive kv cache compression for llms. In *International Conference on Learning Representations*, volume 2024, pp. 22975–22988, 2024.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Yuchen Guo, Guiguang Ding, Jungong Han, and Xiaoming Jin. Robust iterative quantization for efficient lp-norm similarity search. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence*, pp. 3382–3388, 2016.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

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.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Qing-Yuan Jiang and Wu-Jun Li. Asymmetric deep supervised hashing. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023.

Hanjiang Lai, Yan Pan, Ye Liu, and Shuicheng Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3270–3278, 2015.

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.

Haomiao Liu, Ruiping Wang, Shiguang Shan, and Xilin Chen. Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2064–2072, 2016.

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *2012 IEEE conference on computer vision and pattern recognition*, pp. 2074–2081. IEEE, 2012.

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.

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.

Hugo Touvron Llama. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2407.21783*, 2023.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pp. 525–542. Springer, 2016.

Luka Ribar, Ivan Chelombiev, Luke Hudlass-Galley, Charlie Blake, Carlo Luschi, and Douglas Orr. Sparq attention: Bandwidth-efficient llm inference. *arXiv preprint arXiv:2312.04985*, 2023.

Pratyusha Sharma, Jordan Ash, and Dipendra Kumar Misra. The truth is in there: Improving reasoning in language models with layer-selective rank reduction. In *International Conference on Learning Representations*, volume 2024, pp. 17632–17651, 2024.

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

Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest: Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*, 2024.

Yi Tay, Dara Bahri, Liu Yang, Donald Metzler, and Da-Cheng Juan. Sparse sinkhorn attention. In *International conference on machine learning*, pp. 9438–9447. PMLR, 2020.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Computing Surveys*, 55(6):1–28, 2022.

Apoorv Vyas, Angelos Katharopoulos, and François Fleuret. Fast transformers with clustered attention. *Advances in Neural Information Processing Systems*, 33:21665–21674, 2020.

Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):769–790, 2017.

Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. *Advances in neural information processing systems*, 21, 2008.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International conference on machine learning*, pp. 38087–38099. PMLR, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Khai Hao, Xu Han, Zhen Leng Thai, Shuo Wang, Zhiyuan Liu, et al.

$$\backslash$$

infty $bench : Extendinglongcontextevaluationbeyond100ktokens.$ *arXiv preprint arXiv:2402.13718*, 2024.

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.

Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zihao Ye, Lequn Chen, Size Zheng, Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, and Baris Kasikci. Atom: Low-bit quantization for efficient and accurate llm serving. *Proceedings of Machine Learning and Systems*, 6:196–209, 2024.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024.

Han Zhu, Mingsheng Long, Jianmin Wang, and Yue Cao. Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 30, 2016.

A Additional Related Work

A.1 KV Cache Optimization for Long-Context Inference

To alleviate the bottleneck of KV Cache in long-context inference, existing works primarily follow three trajectories. First, quantization compression (e.g., KIVI, Atom) reduces memory usage through low-bit representation. However, this approach suffers from significant accuracy loss at extremely low bit-widths and introduces additional dequantization overhead. Second, selective caching (e.g., StreamingLLM, H2O) eliminates redundancy based on attention sparsity. Although this reduces memory usage to a constant level, rigid eviction strategies result in the permanent loss of historical information, thereby compromising long-range dependency performance. Third, structured sharing (e.g., GQA, PyramidKV) reduces redundancy via cross-head or cross-layer sharing. Nevertheless, these methods often overlook fine-grained internal model heterogeneity and typically require integration during the pre-training phase, making flexible application to existing models difficult.

While the aforementioned methods achieve significant storage compression, they share a common limitation: they fail to fundamentally resolve the attention computation bottleneck. Whether through quantization or sparsification, the core computation still relies on similarity matching of high-dimensional floating-point vectors. Consequently, the computational complexity remains bound by the floating-point operation paradigm, failing to reach the theoretical upper limit of computational efficiency.

A.2 Deep Hashing Methods

Hashing techniques map high-dimensional vectors to binary codes, utilizing bitwise operations to achieve low-latency and low-memory Approximate Nearest Neighbor (ANN) search. Unlike tree-based, quantization-based, or graph-based methods, hashing focuses on preserving bit-level locality. Early random projection methods, represented by Locality-Sensitive Hashing (LSH), were simple but suffered from limited recall rates. Subsequently, learning-based hashing methods (such as Spectral Hashing and ITQ) improved accuracy with shorter code lengths by optimizing mapping functions. In recent years, deep supervised hashing (e.g., DSH, CNNH) has become mainstream by leveraging neural networks to capture complex semantics. Furthermore, asymmetric hashing and multi-hash ensemble strategies have enhanced adaptability in large-scale scenarios like Retrieval-Augmented Generation (RAG) by reducing quantization errors and employing multi-view fusion.

In summary, while approximate retrieval technologies represented by deep hashing have achieved significant success and maturity in fields such as large-scale image retrieval, few have noted the high degree of isomorphism between their technical paradigms and critical scenarios like Transformer internal attention computation. To the best of our knowledge, this work is the first to systematically introduce advanced large-scale data retrieval techniques into the internal optimization of the Transformer attention mechanism. This approach aims to break the traditional paradigm bottleneck dominated by large-scale, high-dimensional floating-point vector similarity calculations during autoregressive generation, thereby opening new technical pathways for next-generation efficient Large Language Model (LLM) inference.

B Datasets

We evaluate our method on six representative tasks from the LongBench benchmark Bai et al. (2024), selected to cover a diverse range of long-context capabilities including single-document QA, multi-document summarization, and multi-hop reasoning. The specifications of each dataset are detailed below:

NarrativeQA This dataset targets reading comprehension over long narratives, such as book summaries or movie scripts. It requires the model to understand the underlying plot and causal relationships within extensive texts. The task involves answering specific questions based on a single long document, challenging the model's ability to maintain context over long sequences.

HotpotQA Focused on multi-hop reasoning, HotpotQA requires models to synthesize information from multiple supporting documents to answer a question. Unlike simple retrieval tasks, the model must bridge distinct pieces of evidence to derive the correct answer. This tests the ability to handle fragmented information and perform logical deduction across multiple contexts.

Qasper This dataset consists of question-answering tasks based on academic research papers from the NLP domain. It evaluates the model's capacity to extract technical details, summarize specific sections, and interpret experimental results. The inputs typically include full-text papers, requiring the processing of dense, domain-specific information.

MultiNews Designed for multi-document summarization, MultiNews requires the model to generate a concise summary based on several news articles covering the same event. This task assesses the ability to identify common themes, resolve conflicting information, and synthesize a coherent narrative from diverse sources.

GovReport This dataset involves summarizing long government reports. The source documents contain formal, policy-oriented language with complex structures. The task challenges the model to distill key policy details and hierarchical information into a comprehensive summary, testing performance on highly structured and formal long-context data.

TriviaQA TriviaQA is a reading comprehension dataset containing question-answer pairs accompanied by evidence documents. The task requires the model to locate factual answers within large amounts of noisy text. It evaluates the efficacy of information retrieval and the precision of factual extraction in long-context scenarios.

C MODELS

We employ three distinct Large Language Models (LLMs) to evaluate the versatility and scalability of DASH-KV. The specific details of each model are as follows:

Qwen2-7B-Instruct Developed by Alibaba Cloud, this model represents the highly optimized 7B parameter class. It utilizes Grouped Query Attention (GQA) in its architecture. GQA significantly reduces the memory footprint of the KV cache during inference. This model serves as a standard baseline for evaluating efficiency in compact dense models.

Llama-3.1-8B-Instruct This model is the latest iteration from Meta, featuring native support for a context window of up to 128k tokens. It employs optimized Rotary Positional Embeddings (RoPE) to handle long sequences effectively. The inclusion of Llama-3.1 allows us to assess the robustness of DASH-KV in scenarios requiring ultra-long context retention.

Qwen2.5-14B-Instruct This model introduces a larger parameter scale compared to the 7B and 8B baselines. It benefits from extensive pre-training on a diverse corpus, resulting in superior reasoning capabilities. Testing on this model verifies whether the performance gains of DASH-KV remain consistent as model size and architectural complexity increase.

D BASELINE

To ensure a comprehensive and fair evaluation, we selected four representative baseline methods that cover the major technical approaches for long-context optimization:

- **Full Attention**: Dense attention mechanism, serving as the upper-bound baseline for performance evaluation.
- **StreamingLLM**Xiao et al. (2024): Leverages the attention sink phenomenon to enable smooth inference over infinitely long texts.
- **H2O**Zhang et al. (2023): Dynamically evicts low-contribution KV pairs to maintain a constant-size cache.

- **SnapKV**Li et al. (2024): Identify the most important prompt features in the "observation window" at the end of the input to compress the LLM KV cache.

E Theoretical Equivalence of Bitwise Operations

A critical premise of our approach is that the computationally expensive floating-point similarity search can be mathematically equivalently replaced by efficient bitwise operations. In our experiments, rather than relying on theoretical simulations, we explicitly implement the retrieval process using INT8 precision coupled with Boolean masks. This design effectively realizes hardware-level bitwise matching logic within standard deep learning frameworks, enabling measurable physical acceleration. Here, we provide the theoretical proof that this simulation is strictly equivalent to the Hamming distance computation in the binary space.

Consider the query hash code $h_q \in \{-1, +1\}^D$ and the key hash code $h_k \in \{-1, +1\}^D$, where D denotes the embedding dimension (e.g., 128).

The **Inner Product** in the binary space is defined as:

$$\text{Sim}(h_q, h_k) = \sum_{i=1}^{D} h_q^{(i)} \cdot h_k^{(i)} \tag{11}$$

Since $h_q^{(i)}, h_k^{(i)} \in \{-1, +1\}$, the product term $h_q^{(i)} \cdot h_k^{(i)}$ takes the value $+1$ if the bits are identical ($h_q^{(i)} = h_k^{(i)}$) and -1 if they are different ($h_q^{(i)} \neq h_k^{(i)}$).

The **Hamming Distance** $H(h_q, h_k)$ counts the number of positions where the bits differ. Let N_{diff} be the number of differing bits (Hamming distance) and N_{same} be the number of identical bits. We have the following relationships:

$$N_{diff} = H(h_q, h_k) \tag{12}$$

$$N_{same} = D - H(h_q, h_k) \tag{13}$$

Substituting these into the inner product formula:

$$\text{Sim}(h_q, h_k) = N_{same} \cdot (+1) + N_{diff} \cdot (-1) \tag{14}$$

$$= (D - H(h_q, h_k)) - H(h_q, h_k) \tag{15}$$

$$= D - 2 \cdot H(h_q, h_k) \tag{16}$$

This derivation proves a strict linear mapping between the inner product and the Hamming distance:

$$\arg\max_k \text{Sim}(h_q, h_k) \iff \arg\min_k H(h_q, h_k) \tag{17}$$

Equation 17 establishes identical rankings for binary inner products and Hamming distances on the same codes. The packed implementation is evaluated separately in Table 4.

F Introduction of residuals

F.1 The necessity of residuals

To elucidate the necessity of the residual MLP, we address a fundamental limitation of binary hashing within the Transformer context: Magnitude-Blindness. Statistics on Key vectors reveal an immense dynamic range (e.g., spanning $[-412, 236]$ in Layer 27) accompanied by significant outliers. Binary hashing compresses both negligible activations and extreme values into identical codewords. This results in "magnitude collapse," hindering the identification of critical "Attention Sinks." The residual MLP remedies this by leveraging continuous representations to detect high-activation features and predict magnitude compensations. By effectively recovering the attention intensity lost during binarization, this approach significantly outperforms pure hashing in scenarios involving heavy-tailed distributions.

Unlike discrete hashing, the MLP exploits continuous space properties to capture original magnitudes, allowing it to predict substantial compensation for high-activation outliers. This effective restoration of the attention intensity lost to binarization establishes the significant superiority of DASH-KV over pure hashing in layers characterized by heavy-tailed distributions.

Table 5: **Performance Comparison on Qwen2-7B-Instruct.** We compare the Full Attention baseline (Original) against variants where specific layers are replaced by our method. The metrics reported are F1 scores for NarrativeQA, HotpotQA, Qasper, MultiNews, and GovReport, and Exact Match (EM) for TriviaQA. **Avg** denotes the average score across all tasks. Note that replacing Layer 0 results in significant degradation, highlighting the sensitivity of early layers.

Configuration	NarrativeQA	HotpotQA	Qasper	MultiNews	GovReport	TriviaQA	Avg
Original (Full Attention)	**25.13**	44.04	**46.13**	**15.42**	18.06	83.47	38.71
Replace Layer 14	23.64	**44.88**	44.94	15.39	18.22	83.48	38.43
Replace Layer 18	24.65	44.50	45.34	15.40	**19.13**	83.33	**38.73**
Replace Layer 27	24.66	43.33	45.59	15.30	18.34	83.46	38.47
Replace Layers 10–24 (Even)	24.01	43.19	44.01	15.13	18.43	**84.37**	38.19
Replace Layer 0 (Ablation)	*8.74*	*21.74*	*12.40*	*2.62*	*3.10*	*31.22*	*13.30*

F.2 Analysis of the selection of the necessary residual module loss function

We evaluated three progressive variants to quantify the impact of residual learning: Pure Hash (without residual compensation), DASH-KV-MSE (trained via Mean Squared Error), and DASH-KV-Distill (trained via distillation).

The experimental results reveal a critical insight: directly fitting the residual function using Mean Squared Error (MSE) results in a significant performance degradation (F1 score drops by 0.39, from 11.31 to 10.92). The underlying cause of this phenomenon is the Scale Shift during Length Extrapolation. When the model learns residuals using the MSE objective on short sequences (3k tokens), it overfits to point-wise absolute value predictions. However, when extrapolated to unseen long contexts (32k tokens) during inference, the numerical range of attention weights shifts significantly, introducing systematic bias into the MSE-optimized residual predictions.

In contrast, the KL divergence-based distillation strategy (Distill) employs a fundamentally different optimization paradigm. Rather than fitting absolute residual values, it guides the residual network to learn the Relative Probability Distribution by adjusting the relative importance rankings among different Keys rather than their absolute similarity scores. This strategy exhibits inherent robustness for length extrapolation: provided the relative attention rankings remain consistent between training and inference, the model can generalize to arbitrary sequence lengths. Our experiments validate this hypothesis: DASH-KV-Distill effectively mitigates the quantization errors introduced by pure hashing, achieving a +0.13 F1 improvement over the full-precision baseline (11.44 vs. 11.31). This underscores the superiority of the distillation objective in long-context scenarios.

G Task Performance

To rigorously evaluate the efficacy of our proposed method, we conducted extensive experiments using the Qwen2-7B-Instruct model as our primary testbed. We systematically replaced the standard attention mechanism with our optimized module across various layer configurations, ranging from specific single layers to multiple intermediate layers. This ablation study aims to verify whether our approach can maintain high-fidelity retrieval and reasoning capabilities compared to the full-attention baseline.

The quantitative results are summarized in Table 5. Our method demonstrates remarkable robustness across all six long-context benchmarks. Notably, replacing the attention mechanism in deeper layers (e.g., Layer 18 or Layer 27) yields performance highly comparable to the original model. For instance, the configuration replacing Layer 18 achieves an average score of 38.73, which marginally surpasses the full-attention baseline of 38.71.

Furthermore, we explored an aggressive compression strategy by replacing a block of eight intermediate layers (Layers 10, 12, ..., 24). Even under this setting, the model maintains a competitive average score of 38.19, exhibiting negligible degradation in multi-hop reasoning (HotpotQA) and summarization tasks (MultiNews). However, we observed that replacing the very first layer (Layer 0) leads to a catastrophic performance drop (13.30 Avg), underscoring the critical role of initial layers in preserving raw token semantics, which are sensitive to approximation.

H Loss Design

Given the ranking-based nature of attention, we employ list-wise distillation Hinton et al. (2015) as our primary training objective. We observe that the distribution of normalized hash dot products, constrained to the $[-1, 1]$ interval, is overly smooth and fails to capture the sharper, more peaked distributions characteristic of the original full-precision attention. To address this discrepancy, we depart from the conventional use of symmetric temperature scaling and introduce an asymmetric temperature scaling mechanism to amplify the numerical distinction between relevant and irrelevant keys. The principal loss is defined as the KL divergence between the student and teacher attention distributions:

$$\mathcal{L}_{\text{distill}} = \text{KL}(P_{\text{student}} \parallel P_{\text{teacher}}). \tag{18}$$

Furthermore, to ensure the quality of discrete hash codes, we introduce two auxiliary constraints. The Bit Balance Loss aims to maximize information entropy, encouraging the mean of each dimension of the hash code to approach zero Liu et al. (2016):

$$\mathcal{L}_{\text{bal}} = \|\text{mean}(h_q)\| + \|\text{mean}(h_k)\|, \tag{19}$$

where h_q and h_k are the binary hash codes for Query and Key, respectively.

The Quantization Loss constrains the hash encoder output to approximate $\{-1, +1\}$ Zhu et al. (2016); Liu et al. (2016). This effectively mitigates performance collapse during the transition from continuous representation to discrete encoding in the inference stage:

$$\mathcal{L}_{\text{quant}} = \mathbb{E}\left[(|h_q| - 1)^2\right] + \mathbb{E}\left[(|h_k| - 1)^2\right]. \tag{20}$$

Combining the above three parts, the final training objective is:

$$\mathcal{L} = \mathcal{L}_{\text{distill}} + \alpha \cdot \mathcal{L}_{\text{bal}} + \beta \cdot \mathcal{L}_{\text{quant}}, \tag{21}$$

where we set the coefficients α and β for the two auxiliary loss terms to 0.1. This configuration ensures that the dominant gradient signal originates from aligning with the teacher model's distribution. these auxiliary terms impose effective regularization on the quality of the discrete hash codes, preventing convergence to degenerate solutions.

I Implementation details

Training uses 3K-token sequences from the official NarrativeQA, HotpotQA, Qasper, MultiNews, GovReport, and TriviaQA training splits. LongBench evaluation instances and their target answers are excluded.

The backbone remains frozen. Each layer has its own hash and residual parameters, shared across heads. With head dimension 128 and query hidden dimension 256, the hash and residual modules contain 161,217 parameters per layer.

Eight such layers contain 1,289,736 hash and residual parameters. Training one layer takes 88.63 minutes on an NVIDIA RTX 4090 (48 GB), with approximately 18.9 GiB of observed GPU memory usage.

Prefill retains full attention and constructs packed codes from RoPE-applied prompt keys. Decoding encodes the current query and appends only the new key code. Full-precision K/V remain in the cache throughout.

The prior set contains the first four and most recent 64 tokens, with duplicates removed. Routing the remaining keys gives approximately 5.20% exact and 4.99% approximate attention at 32K, including the prior set.

The packed-code cache is auxiliary storage. For a 128-dimensional FP16 K/V pair, a 128-bit key code adds 16 bytes to 512 bytes of K/V, or 3.125% per token and KV head in each replaced layer.

Table 2 reports full-model persistent KV-related memory for its eight-layer configuration. The metadata and auxiliary-network totals are 30.77 MiB for Quest and 17.65 MiB for DASH-KV.

The latency evaluation uses packed XOR/POPCNT and fused CUDA routing and sparse-attention kernels. The complete single-layer path includes incremental key encoding, query encoding, and all subsequent attention operations.

I.1 Component Timing

The profiling setup uses Qwen2-7B-Instruct, FP16, batch size 1, a 128K context, and an NVIDIA RTX 4090 (48 GB). Table 6 separates one-time prefill construction from decoding operations.

The component rows are isolated median measurements and are not additive. Fusion changes execution costs; the complete-path event is the measured total used for the speedup comparison.

Table 6: Isolated component timings and independently measured complete-path latency at 128K.

Component	Median latency (ms)
One-time prefill operation	
Full-context key-code construction	0.499
Isolated decoding components	
New-key encoding and hash append	0.111
Query encoding, sign packing, and residual correction	0.220
Hamming-distance scan	0.033
Candidate routing	0.159
Exact QK computation	0.172
Hash-score calibration	0.104
Sparse softmax	0.040
Sparse-V aggregation	0.307
Complete DASH-KV path (independent event)	**0.711**

J Detailed Ablation Studies

J.1 Effectiveness of Residual Learning

To quantify the contribution of residual learning, a comparison is conducted among three progressive system variants: **Pure Hash** (which operates without residual compensation), **DASH-KV-MSE** (optimized via Mean Squared Error), and **DASH-KV-Distill** (trained utilizing Knowledge Distillation).

The results presented in Figure 3 demonstrate a notable phenomenon: directly optimizing the residual function via Mean Squared Error (MSE) leads to a substantial degradation in model performance. Specifically, the F1-score declines from 88.45 to 85.40, reflecting a reduction of 3.05 points. This performance drop is primarily attributed to numerical drift occurring during length extrapolation.

When a model is trained to learn residuals using an MSE objective on short-context data (3k tokens), it learns to approximate point-wise absolute value functions. However, during inference on substantially longer, unseen contexts (32k tokens), the numerical distribution of attention weights undergoes a substantial shift. This divergence introduces systematic biases in the residual predictions optimized under the MSE framework.

In contrast, the knowledge distillation strategy based on KL divergence (Distill) pursues a fundamentally distinct optimization direction. Rather than fitting absolute residual values, it compels the residual network to learn the relative probability distribution. Essentially, this approach adjusts the relative importance ranking across different Keys instead of matching their absolute similarity scores. This strategy inherently enhances robustness for length extrapolation. Provided that the relative ranking of attention remains consistent between training and inference phases, the model can generalize effectively to arbitrary sequence lengths. Our experimental results substantiate this theoretical premise: DASH-KV-Distill effectively mitigates the quantization errors introduced by pure hashing. It yields a performance improvement of +1.02 points over the full-precision baseline (89.47 versus 88.45), thereby demonstrating the superiority of the distillation objective in long-context scenarios.

J.2 Component Effectiveness

To further elucidate the advantages of the asymmetric architecture, we establish two primary baseline models for comparative analysis. The first baseline, termed DASH-KV-Sym (Symmetric Hashing), employs identical encoding networks for both Query and Key representations. The second baseline, referred to as DASH-KV-Naive (LSH), substitutes the learned deep hashing function with conventional Locality-Sensitive Hashing (LSH).

The results presented in Table 3 demonstrate that the asymmetric design of DASH-KV yields a substantial improvement in accuracy. While Locality-Sensitive Hashing (LSH) is computationally efficient due to its parameter-free formulation, its retrieval precision is considerably lower than that of our trained hash encoder, as reflected by a markedly reduced Recall@100. This outcome supports the conclusion that, within the specific context of attention retrieval, a data-driven hash space optimized explicitly for semantic similarity preservation significantly outperforms generic geometric hashing approaches.

J.3 Parameter Sensitivity

To deeply understand the relationship between system hyperparameters and performance, we systematically investigate the impact of the hash code length b (Hash Bits) on efficiency and accuracy. We incrementally adjust the hash code length from 32 bits to 256 bits and observe the corresponding changes in retrieval precision and GPU memory usage. The experimental results in Figure 4 illustrate the trade-off between hash code length and system efficiency. Initially, as the hash code length increases from 32 to 64 bits, retrieval accuracy (Recall@K) exhibits a sharp ascent, rapidly climbing from 58.2% to over 70.5%. However, beyond 128 bits, where accuracy reaches 88.6%, the performance gain plateaus. Extending the code length further to 256 bits yields diminishing marginal returns, with accuracy improving only slightly to 91.4%. In contrast, KV Cache memory usage grows linearly with code length. Balancing these factors, we identify 128 bits as the optimal configuration. At this point, the system achieves near-saturated retrieval precision with a modest memory footprint of 0.857 GB, effectively maximizing storage efficiency without compromising long-text generation quality.

J.4 Query Encoder Depth

We vary the query encoder depth while holding the key encoder, 128-bit code length, training data, objective, routing ratios, and selected layers fixed. Table 7 reports the accuracy–latency trade-off.

The three-layer encoder improves the average score by 0.98 over two layers, while increasing the complete-layer latency by 0.027 ms. This supports the default depth within this ablation setup.

These timings belong to the depth-ablation setup and are reported separately from the packed-kernel measurements in Table 4.

Table 7: Query-depth ablation on Qwen2-7B-Instruct. Parameter counts refer to the query encoder.

Query depth	Query parameters	LongBench Avg.	Query latency (ms)	DASH layer latency (ms)
1	16,512	37.07	0.055	0.566
2	66,432	37.75	0.144	0.657
3	132,224	38.73	0.170	0.684

J.5 Residual Network Size

Table 8 varies the number of linear layers and hidden dimension in the residual MLP. The six-task average ranges from 38.66 to 38.73, indicating limited sensitivity within the tested configurations.

Table 8: Residual-network size ablation on Qwen2-7B-Instruct. Task abbreviations follow Table 1.

Layers	Hidden dim.	NQ	HQ	QM	MN	GR	TQ	Avg.
2	32	24.60	44.40	45.28	15.38	19.08	83.25	38.66
2	64	24.62	44.42	45.30	15.39	19.10	83.28	38.68
2	128	24.63	44.45	45.32	15.40	19.11	83.30	38.70
2	256	24.64	44.48	45.33	15.40	19.12	83.32	38.71
3	32	24.61	44.43	45.30	15.39	19.10	83.29	38.69
3	64	24.63	44.46	45.31	15.40	19.11	83.31	38.70
3	128	24.64	44.49	45.33	15.41	19.12	83.33	38.72
3	256	24.65	44.50	45.34	15.40	19.13	83.33	38.73

K LAYER-WISE ANALYSIS

K.1 THE NECESSITY OF LAYER-WISE TRAINING

Our empirical investigation uncovers a fundamental challenge in hashing-based attention optimization: the significant distributional heterogeneity of Query ($\mathbf{Q}$) and Key ($\mathbf{K}$) representations across network depths. As widely recognized, Large Language Models process information hierarchically, with shallower layers encoding local syntactic patterns and deeper layers capturing abstract semantic dependencies. This functional distinctiveness suggests that the optimal low-rank manifolds for preserving similarity structures are layer-dependent.

To substantiate this hypothesis, we first visualize the activation landscapes of different layers in Figure 7. The visualization reveals distinct clustering patterns and density variations, confirming that the feature subspaces undergo substantial transformation (i.e., manifold shift) from bottom to top layers. Consequently, a "one-size-fits-all" hashing projection trained on a specific distribution is theoretically insufficient to capture the diverse geometric characteristics present throughout the model.

To strictly quantify the impact of this heterogeneity, we conducted a cross-layer generalization experiment (see Figure 8). Specifically, we isolated the hashing network weights optimized on the data distribution of an intermediate layer (Layer 14) and directly transferred them to perform retrieval on a shallow layer (Layer 1) and a deep layer (Layer 27).

As evidenced by the results, this direct weight transfer precipitates a marked degradation in recall performance compared to layer-specific baselines. The inability of Layer 14's parameters to generalize to other depths corroborates our visual findings: the geometric structure of the attention mechanism is highly layer-specific. This phenomenon underscores the limited generalizability of a single global weight configuration. Therefore, we conclude that **independent parameter optimization for each layer (Layer-Wise Training)** is not merely an engineering choice but a theoretical necessity for ensuring high-fidelity retrieval and minimizing quantization errors across the entire model.

K.2 LAYER-WISE SENSITIVITY ANALYSIS

We observe that Transformer layers demonstrate significantly varying tolerances to sparsification. To identify the optimal deployment strategy for DASH-KV, we conduct a systematic layer-wise sensitivity analysis. Specifically, we sequentially replace each layer in the Qwen2-7B model with DASH-KV and assess the resulting performance degradation across six LongBench tasks.

As illustrated in Figure 9, the sensitivity profile follows an Inverted-U-shaped curve, which is readily interpretable.

Early and Later Layers(e.g., Layer 0-4,27-32). Replacing these layers with DASH-KV results in a severe performance collapse, as they play a pivotal role in preserving the model's fundamental features and final representations.

Middle Layers(e.g., Layer 5-27). In sharp contrast, the intermediate layers exhibit considerable robustness to sparsification. Substituting these layers with DASH-KV leads to only marginal per-

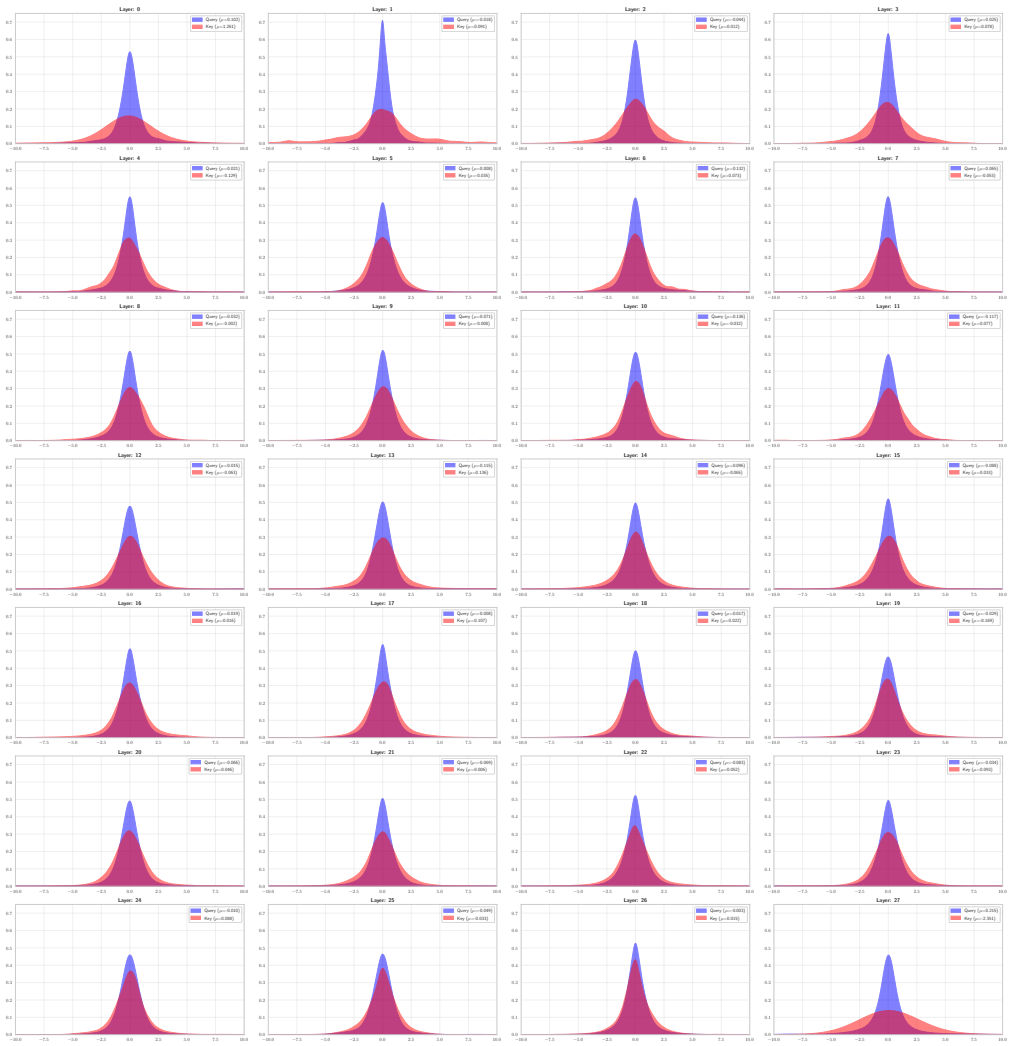

Figure 7: **Visualization of Query and Key Activation Distributions Across Layers.** We project the Query and Key vectors from representative layers (e.g., Layer 0, 14, 27) into a 2D space. The marked visual discrepancies indicate that the geometric distribution of attention features drifts significantly as network depth increases, implying that a static global hashing projection cannot optimally fit all layers simultaneously.

formance degradation, indicating that they possess substantial redundancy, which can be effectively compressed without compromising model efficacy.

L Evaluation on Selected ∞Bench Tasks

We evaluate Qwen2-7B-Instruct on three ∞Bench tasks Zhang et al. (2024): passkey retrieval, English QA, and English multiple-choice QA. The average gap to Full Attention is 0.12 points (Table 9).

M Limitations

Regarding evaluation scope, our current benchmarks validate the effectiveness of DASH-KV on representative long-context understanding and summarization tasks across multiple backbone models. Future work can further examine its behavior in broader deployment scenarios, such as long-horizon

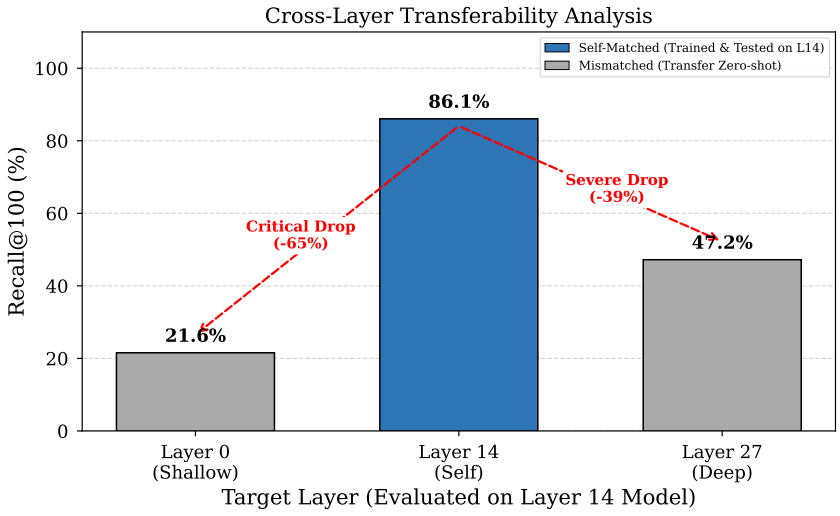

Figure 8: **Impact of Cross-Layer Weight Transfer on Retrieval Recall.** The figure illustrates the performance degradation when applying hash functions trained on an intermediate layer (Layer 14) to a shallow layer (Layer 1) and a deep layer (Layer 27). The sharp decline in recall confirms the distributional mismatch between layers.

Table 9: Accuracy on three selected ∞Bench tasks using Qwen2-7B-Instruct.

Method	R.PK	En.QA	En.MC	Avg.
Full Attention	28.81	14.92	54.59	32.77
DASH-KV	28.65	14.88	54.42	32.65

dialogue, retrieval-augmented generation, code understanding, and large-scale production serving workloads. These directions are orthogonal to the proposed framework and may further broaden its practical impact.

N Layer Sensitivity and Deployment Strategy

We conduct a layer-wise sensitivity analysis by replacing each Qwen2-7B layer with DASH-KV and evaluating degradation across six LongBench tasks. As shown in Figure 9, the sensitivity profile follows an inverted U-shaped curve: early layers (e.g., Layers 0–4) and late layers (e.g., Layers 27–32) exhibit severe performance collapse when replaced, while middle layers (e.g., Layers 5–27) are substantially more robust. This pattern indicates that initial layers are critical for preserving raw token semantics and final layers are indispensable for task-specific representation, whereas intermediate layers possess considerable redundancy. Based on these findings, we adopt a sandwich deployment strategy: full attention is retained in the initial and final layers to preserve representational stability, while DASH-KV is applied to the middle layers to maximize computational efficiency.

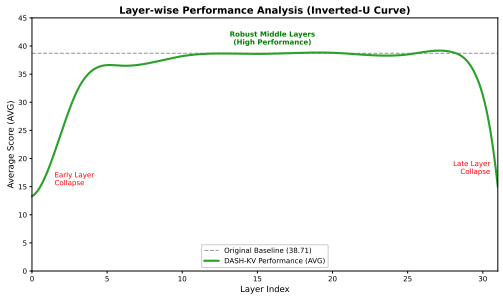

Figure 9: Layer-wise sensitivity of DASH-KV deployment.