HAPPI: Accurate 2-bit KV cache compression with Low-error truncated SVD approximation

Anonymous ACL submission

Abstract

Truncated Singular Value Decomposition (SVD) has recently attracted renewed attention for its effectiveness in model optimization, such as LoRA initialization and KV-cache compression. However, exact SVD remains computationally expensive, while approximate methods like power iteration often introduce non-negligible errors. In this paper, we present Hadamard PCA-based Power Iteration (HaPPI), a new algorithm that significantly improves the accuracy of low-rank approximation while retaining computational efficiency. Compared to prior methods, HaPPI achieves the lowest approximation error at a practical computational cost. Building on this foundation, we further propose HaPPI-KV, which combines HaPPI with key whitening and residual quantization to deliver high compression ratios for key–value caches. HaPPI-KV compresses KV-cache memory by up to $6.37\times$ relative to the FP16 baseline, surpasses all SVD-based state-of-the-art methods, and recovers near full-precision accuracy on downstream benchmarks.

1 Introduction

Truncated Singular Value Decomposition (SVD) has recently re-emerged as a powerful tool for optimizing transformer-based models (Wang et al., 2024, 2025a; Yuan et al., 2023; Hsu et al., 2022; Wong et al., 2025; Chen et al., 2021; Sy et al., 2024). By representing data in compact low-rank structures, it significantly reduces storage overhead and has long served as a cornerstone of model compression and lightweight adaptation. Its practical utility is further demonstrated in recent work: initializing LoRA adapters (Meng et al., 2024; Yang et al., 2024) with SVD markedly improves fine-tuning accuracy, while exploiting the inherent low rank of key and value matrices enables highly effective KV-cache compression (Yankun et al., 2025; Kang et al., 2024; Chang et al., 2025, 2024; Zhang et al., 2024; Wang et al., 2025b; Yan et al., 2025; Lin et al., 2024a). Together, these results underscore the growing importance of truncated SVD in modern optimization.

To advance the quality of low-rank approximation, we propose Hadamard PCA-based Power Iteration (HaPPI), a novel algorithm that achieves precise truncated SVD approximations with practical efficiency. HaPPI builds upon power iteration, a widely used approximate technique, by integrating Hadamard transformations and PCA-inspired initialization. Empirically, HaPPI achieves the lowest per-tensor error among existing methods and recovers fine-tuning performance in LoRA adapters even in scenarios where previous approaches fail due to significant approximation errors.

Going a step further, we propose HaPPI-KV, a novel method for KV-cache compression. HaPPI-KV extends HaPPI with two additional techniques, key whitening and residual quantization, enabling both rapid compression and state-of-the-art reconstruction quality under tight memory constraints.

In summary, our contributions are threefold: 1) We propose HaPPI, a fast and accurate algorithm for truncated SVD approximation. 2) We provide comprehensive theoretical, quantitative, and empirical validation of HaPPI, including its application to LoRA initialization. 3) We design HaPPI-KV, a high-quality and efficient KV-cache compression technique that outperforms existing approaches. We believe that HaPPI opens new avenues for model optimization, with HaPPI-KV serving as a compelling demonstration of its potential to advance KV-cache compression.

2 Related Work

2.1 KV Cache Compression

KV cache compression has been widely explored through two main approaches: quantization and low-rank approximation. Quantization-based techniques, such as KIVI (Liu et al.,

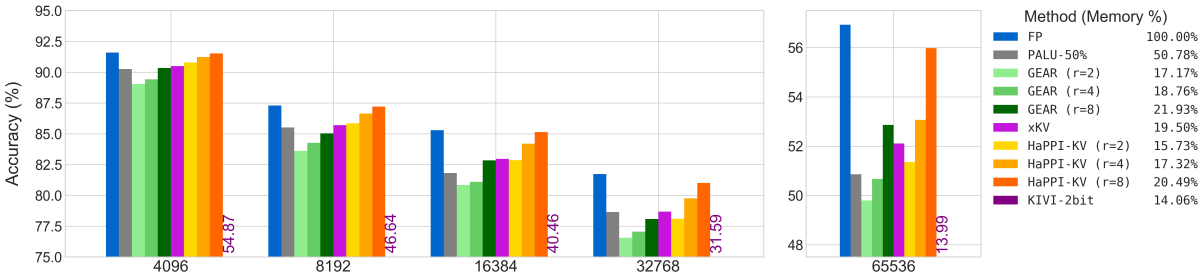

Figure 1: Accuracy across different sequence lengths on Qwen2-7B-Instruct using the RULER dataset. Values in parentheses indicate the relative KV-cache size compared to the FP16 baseline.

2024b), KVQuant (Hooper et al., 2024), and WKVQuant (Yue et al., 2024), reduce memory footprint by representing each element with reduced precision. On the other hand, as KV caches are inherently amenable to compression via low-rank structures, several methods leverage this property, including GEAR (Kang et al., 2024), PALU (Chang et al., 2024), and SVDq (Yankun et al., 2025).

2.2 Truncated SVD-Based Methods

Beyond KV cache compression, truncated SVD plays a crucial role in various optimization tasks, such as weight compression for inference (e.g., SVD-LLM (Wang et al., 2024), SVD-LLMv2 (Wang et al., 2025a)) and LoRA initialization (e.g., PiSSA (Meng et al., 2024), CorDA (Yang et al., 2024)). The accuracy of the low-rank approximation significantly influences the final performance of these techniques. For instance, PiSSA reports that LoRA adapters initialized with a naive approximation algorithm exhibit degraded quality after fine-tuning compared to those initialized using an exact truncated SVD. The fast and precise approximation provided by HaPPI enhances the practicality of such methods as well, as we will demonstrate in the Appendix.

2.3 SVD Approximation Methods

To make low-rank decomposition computationally affordable, many approximation techniques have been proposed. The randomized SVD framework (`torch.svd_lowrank` (Halko et al., 2011)) generates low-rank approximations by projecting onto random subspaces, while power iteration methods (Vogels et al., 2019) iteratively refine estimates through matrix-vector products. Block Krylov methods (Musco and Musco, 2015) achieve faster convergence by building richer subspaces, and recent theoretical advances have established tight complexity bounds for approximate SVD (Allen-Zhu and Li, 2016). In this work, we present an improved power iteration algorithm that introduces a novel initialization strategy leveraging the Hadamard transform and PCA. Our theoretical analysis (Section 4.1) shows that this approach improves numerical stability under finite-precision arithmetic, particularly for matrices with outliers, which are a common characteristic of KV cache tensors in transformers.

3 Preliminaries

We briefly introduce the key building blocks of HaPPI; full definitions and derivations are provided in Appendix A.

Singular Value Decomposition. Any matrix $A \in \mathbb{R}^{m\times n}$ admits the decomposition $A = U\Sigma V^T$, where U, V are orthonormal and Σ contains singular values in descending order. A rank-r truncation $A_r = U_{[:,1:r]}\Sigma_{1:r,1:r}V^T_{[:,1:r]}$ yields the optimal low-rank approximation under the Frobenius norm, but computing it exactly costs $O(mn^2)$.

Hadamard Transform The Hadamard transform (HT; (Sylvester, 1867)) projects data into the frequency domain using the orthogonal Walsh–Hadamard matrix H, satisfying $HH^T = I$. Recent studies (Xi et al., 2023; Lin et al., 2024b; Tseng et al., 2024; Kim et al., 2025; Ashkboos et al., 2024; Liu et al., 2024a; Ashkboos et al., 2025; Xiang and Zhang, 2024) have shown that HT effectively mitigates activation outliers, a property central to HaPPI.

4 Hadamard PCA-based Power Iteration

Existing approximate methods either incur high computational cost (`torch.svd_lowrank`) or produce a non-trivial error level regardless of the iteration count (power iteration, Figure 2). This residual error propagates to downstream tasks and is especially harmful for transformer tensors with pronounced outliers. To solve this problem, we

Algorithm 1 Power Iteration ((Kang et al., 2024))

Require: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, Rank r, Loop L
Ensure: Matrix $\mathbf{P} \in \mathbb{R}^{n \times r}$, $\mathbf{Q} \in \mathbb{R}^{m \times r}$

```
Initialize P with random distribution.
Initialize Q with random distribution.
for l = 0 to L − 1 do
    if l = L − 1 then
        [P, ∼] ← QR(P)
    end if
    Q ← AP
    if l = L − 1 then
        [Q, ∼] ← QR(Q)
    end if
    P ← A^T Q
end for
return P, Q
```

Algorithm 2 HaPPI (Ours)

Require: Matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$, Rank r, Loop L
Ensure: Matrix $\mathbf{P} \in \mathbb{R}^{n \times r}$, $\mathbf{Q} \in \mathbb{R}^{m \times r}$

```
1:  A_H ← AH
2:  C_H ← A_H^T A_H
3:  P, _ ← PowerIter(C_H, r, min(2, L))
4:  for l = 0 to L − 1 do
5:      if l = L − 1 then
6:          [P, ∼] ← QR(P)
7:      end if
8:      Q ← A_H P
9:      if l = L − 1 then
10:         [Q, ∼] ← QR(Q)
11:     end if
12:     P ← A_H^T Q
13: end for
14: return H^{−1} P, Q
```

propose Hadamard PCA-based Power Iteration (HaPPI), an algorithm that delivers near-exact truncated SVD approximation while preserving the efficiency of power iteration. HaPPI closes this gap with two complementary innovations layered on top of standard power iteration: (i) a Hadamard transform on the input that suppresses outlier-induced numerical errors, and (ii) a PCA-based initialization that accelerates convergence beyond what random initialization can achieve. Both additions are lightweight preprocessing steps, keeping the per-iteration cost on par with plain power iteration.

Outlier suppression via HT. HaPPI first projects the input tensor into the frequency domain via the Hadamard transform, $A_H = AH$ (line 1 of algorithm 2). The Hadamard transform redistributes per-row energy uniformly across components, reducing the magnitude of channel-wise outliers that destabilize power iteration under finite-precision arithmetic. This property is critical for KV cache tensors whose outliers would dominate the approximation error. After iterative refinement on A_H, the recovered singular vectors are mapped back to the original space via the inverse Hadamard transform (line 14 of algorithm 2).

Accelerated convergence via PCA initialization. Rather than initializing $\mathbf{P}$ with random vectors, HaPPI initializes it with approximate principal components of A_H. Concretely, it computes the covariance matrix $C_H = A_H^T A_H$ and applies $\min(2, L)$ power-iteration steps to obtain its top-r singular vectors (lines 2, 3 of algorithm 2). Since power iteration on C_H converges at a quadratically faster rate than on A_H, even two iterations produce vectors well aligned with the dominant directions of A_H, providing a substantially better starting point at negligible overhead (formalized in Section 4.1).

4.1 Theoretical Analysis of HaPPI

We next provide theoretical justification for HaPPI's Hadamard transform and PCA initialization strategy. While HaPPI and standard power iteration are mathematically equivalent in infinite precision, their behavior differs significantly under finite-precision arithmetic. Full proofs and detailed derivations are deferred to Appendix B.

Outlier Factor and Numerical Condition. To quantify the impact of outliers on numerical stability, we define the *outlier factor*:

$$\gamma(A) = \frac{\|A\|_{\max}^2}{\frac{1}{mn}\|A\|_F^2} = \frac{mn \cdot \max_{i,j} |a_{ij}|^2}{\|A\|_F^2} \tag{1}$$

KV cache tensors in transformers typically have $\gamma(A) \gg 1$ due to attention-related activation patterns. We define the *numerical effective condition number* as $\kappa_{\text{eff}}(A) = \kappa(A) \cdot \gamma(A)$. This product arises naturally from floating-point error analysis: in each matrix-vector product Ap, the per-element rounding error scales with $\|A\|_{\max}$ (captured by γ), while $\kappa(A)$ governs the amplification of such errors across iterations (see Appendix B.2 for derivation).

HT Reduces Numerical Error. The Hadamard transform reduces the outlier factor by redistributing energy across components:

Proposition 1. *Let $A \in \mathbb{R}^{m \times n}$ be a matrix where each row's energy is concentrated in at most k entries (i.e., the top-k entries account for the dominant fraction of the row's ℓ_2 norm). Let $A_H = AH$.*

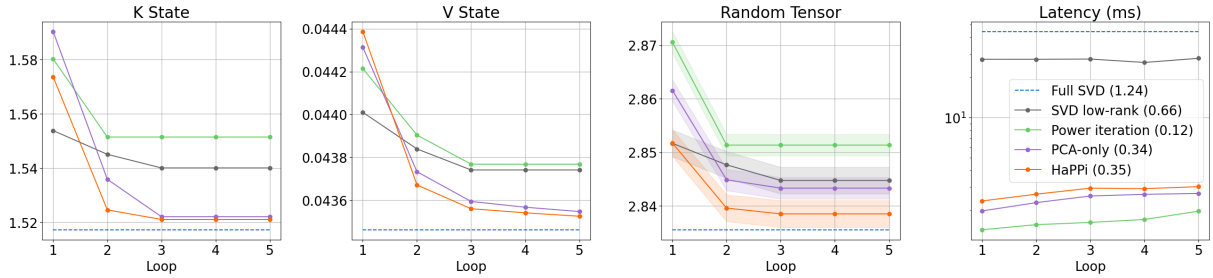

Figure 2: MSE across different numbers of iterations of various low-rank approximation methods. K & V states (left, center) are from the 11th layer of Llama3-8B-Instruct. Note that PCA-only is HaPPI without Hadamard Transformation, and the number next to each method represents GFLOPs. Solid lines indicate the mean over 5 runs; shaded regions denote standard deviation.

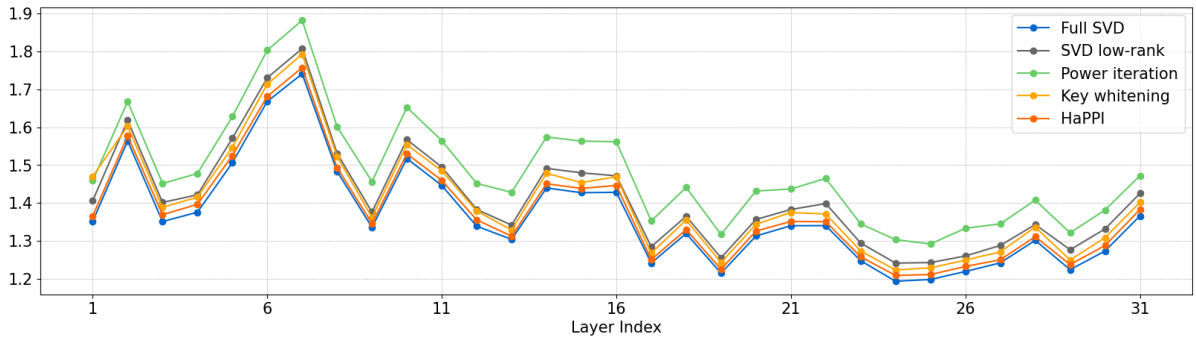

Figure 3: MSE comparison per layer of Llama3-8B-Instruct with the GSM8K dataset.

Then $\gamma(A_H) \leq \frac{k^2}{n}\gamma(A)$. *In the extreme case* $k = 1$, *this yields* $\gamma(A_H) \approx \frac{1}{n}\gamma(A)$.

This reduction directly improves numerical stability: the accumulated error scales as $O(\epsilon_{\text{mach}} \cdot L \cdot \kappa_{\text{eff}}(A_H))$ for HaPPI versus $O(\epsilon_{\text{mach}} \cdot L \cdot \kappa_{\text{eff}}(A))$ for standard power iteration.

PCA Initialization Accelerates Convergence. HaPPI initializes using approximate principal components from $C_H = A_H^T A_H$. Since C_H has eigenvalues σ_i^2, power iteration on C_H converges at rate $(\sigma_2/\sigma_1)^2$ rather than σ_2/σ_1, providing an effective head start of 4 iterations compared to random initialization (see Appendix B for derivation).

Combined Error Bound. Combining both effects, the approximation error of HaPPI after L iterations can be bounded as:

$$\begin{aligned}\|A - \hat{A}_r^{\text{HaPPI}}\|_F \leq \|A - A_r\|_F \\ + \underbrace{O\left(\left(\frac{\sigma_{r+1}}{\sigma_r}\right)^{L+4}\right)}_{\text{convergence error}} \\ + \underbrace{O\left(\epsilon_{\text{mach}} \cdot L \cdot \kappa(A) \cdot \frac{k^2\,\gamma(A)}{n}\right)}_{\text{numerical error}}\end{aligned} \tag{2}$$

where A_r is the optimal rank-r approximation. This bound combines the convergence and numerical error terms independently; cross-terms of order $O(\epsilon_{\text{mach}} \cdot (\sigma_{r+1}/\sigma_r)^L \cdot \kappa(A))$ are omitted as they are dominated by the individual terms when $\epsilon_{\text{mach}} \cdot \kappa_{\text{eff}}(A) \ll 1$, which holds for standard FP16/BF16 arithmetic on the models we evaluate. Compared to standard power iteration, HaPPI improves the convergence exponent from L to $L + 4$ due to PCA initialization, and reduces the numerical error by a factor of up to n/k^2 due to Hadamard-based outlier suppression.

5 Empirical Analysis of HaPPI

Building on the theoretical analysis that established a tighter error bound for HaPPI over standard power iteration, we now provide empirical validation. In this section, we demonstrate experimentally that HaPPI attains an approximation error nearly indistinguishable from that of exact SVD while retaining a latency comparable to that of plain power iteration. We substantiate this through Mean Squared Error (MSE) comparisons against widely used approximate and exact SVD baselines on both real KV cache tensors and random matrices, complemented by a layer-wise analysis across an entire LLM.

First, we conducted a series of Mean Squared Error (MSE) comparisons across different low-rank

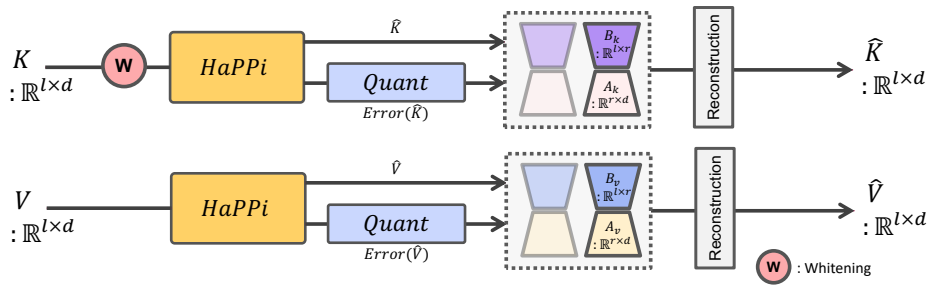

Figure 4: Overview of the pipeline of HaPPI-KV.

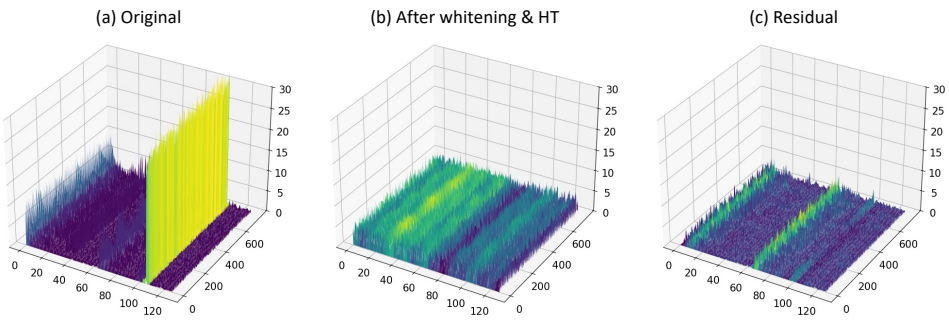

Figure 5: 3D plot of K state of layers.0.attention of Llama3-8B-Instruct.

approximation methods (Figure 2). The evaluation used real K and V tensors extracted from the 11th layer of the Llama3-8B-Instruct model (Grattafiori et al., 2024) on the GSM8K dataset (Cobbe et al., 2021), along with a random tensor of an identical shape (802 × 128) to assess generalization under a rank-2 approximation. Additionally, Figure 3 presents layer-wise MSE results across all 32 layers of the model. We compared five approaches: (1) Full SVD (torch.svd), representing the theoretically most accurate baseline; (2) SVD low-rank (`torch.svd_lowrank`), a memory-efficient approximation variant; (3) power iteration proposed in GEAR; (4) PCA-only, which uses principal vectors of the covariance matrix without HT; and (5) HaPPI, the proposed scheme.

As illustrated in Figure 2, HaPPI achieves approximated results that are consistently closer to the ground-truth SVD solution than prior methods on both the key tensor and the random tensor. While power iteration exhibits the lowest per-iteration computational overhead, its error remains stagnant even with additional iterations. In contrast, HaPPI not only converges more rapidly to a lower error level than `torch.svd_lowrank` but also achieves higher approximation accuracy. The layer-wise analysis (Figure 3) confirms that HaPPI achieves the lowest MSE across all 32 layers, with its performance nearly overlapping with Full SVD and SVD low-rank.

Notably, HaPPI achieves this accuracy at only 0.35 GFLOPs, which is significantly lower than SVD low-rank (0.66) and full SVD (1.24) while remaining close to the latency of power iteration (see Appendix C for a detailed overhead analysis).

6 HaPPI-KV

Building on HaPPI, we introduce HaPPI-KV, a KV-cache compression framework that pairs low-rank approximation with residual quantization to achieve a state-of-the-art compression ratio without sacrificing accuracy. The core insight is that once HaPPI extracts the dominant low-rank structure of the KV cache, the remaining residual is outlier-free and amenable to aggressive 2-bit quantization, yielding compression ratios well beyond what either technique attains in isolation. To make this synergy work on the key tensor, whose channel-wise outliers would otherwise degrade low-rank approximation, HaPPI-KV additionally introduces a lightweight key whitening step that runs before HaPPI decomposition.

6.1 2-bit Residual Quantization

Figure 4 illustrates the HaPPI-KV pipeline. For both key and value tensors, HaPPI-KV applies HaPPI-based low-rank approximation and stores the decomposed factors (A_k, B_k for the key and A_v, B_v for the value) as the primary compressed representation. The residual tensors are then quan-

tized to 2 bits using KIVI (Liu et al., 2024b). Because HaPPI has already absorbed the high-energy components and outliers, the residual exhibits a near-uniform distribution that 2-bit quantization can represent with high fidelity, enabling aggressive memory savings without sacrificing accuracy.

The pipeline treats key and value asymmetrically. The value tensor has a relatively uniform distribution and is fed directly into HaPPI. The key tensor, in contrast, contains extreme channel-wise outliers (Figure 5 (a)) that severely degrade low-rank approximation quality if applied naively. HaPPI-KV therefore applies a lightweight whitening step on the key before HaPPI, which we detail next.

6.2 Key Whitening

Algorithm 3 HaPPI with Key Whitening

Require: Key $\mathbf{K} \in \mathbb{R}^{l \times d}$, Rank r_1, r_2, Loop L
Ensure: Compressed Key $\mathbf{K}' \in \mathbb{R}^{l \times d}$
1: $\mathbf{C} \leftarrow \mathbf{K}^T\mathbf{K}$
2: $\mathbf{P_C}, _ \leftarrow \mathrm{HaPPI}(\mathbf{C}, r_1, \min(2, L))$
3: $\mathbf{K_C} \leftarrow \mathbf{K}\mathbf{P_C}$
4: $\mathbf{P}, \mathbf{Q} \leftarrow \mathrm{HaPPI}(\mathbf{K_C}, r_2, L)$
5: $\mathbf{K'_C} \leftarrow \mathbf{Q}\mathbf{P^T}$
6: $\mathbf{K}' \leftarrow \mathbf{K'_C}\mathbf{P^T_C}$
7: **return** $\mathbf{K}'$

The goal of key whitening is to suppress channel-wise outliers in the key tensor before HaPPI decomposition. Conceptually, it transforms the key into a coordinate system in which the covariance is isotropic, so that the subsequent low-rank approximation operates on a well-conditioned matrix with uniform variance across dimensions. Prior work (Wang et al., 2024, 2025a) has applied similar whitening to input activations for weight compression; we extend the idea to the key tensor and, crucially, replace full SVD with HaPPI to make it practical for on-the-fly KV-cache compression. Algorithm 3 summarizes the resulting procedure, which we derive below.

Optimization objective. To minimize the error introduced into the attention mechanism, the compressed key should produce attention scores as close as possible to those of the original key for a given query. Introducing a whitening matrix S, the objective becomes:

$$\min \left(\left\| QK^T - QS\,\mathrm{HaPPI}(S^{-1}K^T) \right\|_F \right) \tag{3}$$

where $Q, K \in \mathbb{R}^{l \times d}$ denote the query and key, and l, d are the sequence length and hidden dimension.

Choice of whitening matrix. Ideal whitening requires the transformed key to have unit covariance, i.e., $(S^{-1}K^T)(S^{-1}K^T)^T = I$, leading to the condition $K^TK = SS^T$. Applying SVD to K^TK yields $K^TK = U\Sigma^2U^T$, so $S = U\Sigma$, meaning that normalization corresponds to projecting along the principal axes of the key. This suppresses outlier-dominated channels while preserving the essential structure of the tensor.

Practical algorithm. Full SVD on K^TK is impractical for KV caches generated dynamically during inference. Algorithm 3 therefore uses HaPPI itself to approximate the eigenvectors at practical cost, retaining the theoretical foundation of full whitening. We adopt a rank r_1 larger than the final compression rank r_2 used for the key (empirically, $r_1 = 4r_2$ yields the best performance), and limit the whitening-stage iterations to at most 2, which is sufficient to obtain a reliable whitening basis with minimal additional overhead. The principal components P_C transform the key, HaPPI performs low-rank approximation in this transformed space, and the result is mapped back by multiplying by P_C^T. To avoid redundant computation, we compute the covariance $C = K^TK$ once and reuse it across both the whitening step and the inner HaPPI call: the PCA-initialization step of the inner HaPPI requires $K_C^TK_C$, which can be rewritten as $K_C^TK_C = P_C^TK^TKP_C = P_C^TCP_C$, eliminating the need to recompute the covariance on the transformed key.

Effect. As shown in Figure 5 (b) and (c), the residual after whitening exhibits a far more uniform distribution than the original key, directly translating to improved final compression quality.

7 Experimental Results of HaPPI-KV

Having established HaPPI's low approximation error and HaPPI-KV's strategy for compressing outlier-heavy tensors, we now turn to evaluating the practical effectiveness of HaPPI-KV. In this section, we measure how these algorithmic advantages translate into downstream accuracy, memory savings, and inference latency. To this end, we evaluate HaPPI-KV across three open-source LLMs and three benchmark categories (chain-of-thought reasoning, long-context understanding, and very-long-sequence evaluation), and compare the memory versus latency trade-off against the strongest prior method.

Table 1: Experimental results of LLM inference with various long context datasets. Note that GEAR and HaPPI-KV are quantized to 2-bit integers. Bold numbers indicate the best performance among compression methods, excluding the FP16 baseline. Relative memory usage of each method is reported in Figure 1.

(%)		Chain of Thought			LongBench					
Model	**Method**	**GSM8K**	**AQuA**	**BBH**	**Single**	**Multi**	**Sum.**	**Few-shot**	**Synth.**	**Code**
Llama3-8B -Instruct	FP	78.01	50.79	58.62	29.93	32.14	23.41	57.78	47.12	54.09
	KCVT-3bit	58.23	35.43	43.70	16.50	15.19	20.56	48.47	4.77	40.70
	KIVI-2bit	58.38	30.31	38.10	21.22	19.22	20.66	52.97	7.94	40.86
	PALU-50%	71.87	43.70	54.32	30.79	**32.68**	22.80	57.76	45.99	53.79
	xKV	76.59	49.90	54.10	30.02	32.61	23.44	57.22	**47.67**	52.02
	GEAR (r=4)	75.89	47.64	52.35	29.24	32.51	23.53	57.85	47.06	**53.97**
	GEAR (r=2)	73.69	45.47	52.04	29.20	32.61	23.44	**57.87**	47.50	53.91
	HaPPI-KV (r=4)	**78.08**	**50.00**	**55.27**	30.99	32.38	**23.82**	57.28	47.06	52.73
	HaPPI-KV (r=2)	77.56	48.82	54.11	**31.44**	32.44	23.24	57.02	**47.67**	52.42
Mistral-7B -Instruct	FP	55.95	33.46	47.74	39.02	37.40	26.20	63.05	66.67	59.03
	KCVT-3bit	44.81	33.86	42.84	32.12	25.79	25.37	58.88	43.94	52.36
	KIVI-2bit	41.09	25.20	35.88	31.74	29.16	24.77	57.14	28.48	52.86
	PALU-50%	54.13	**37.80**	46.56	**39.30**	37.56	25.38	62.97	**66.67**	58.87
	xKV	53.87	31.70	46.04	38.92	37.04	24.56	62.97	66.31	59.62
	GEAR (r=4)	54.06	22.05	42.80	38.92	37.33	25.84	62.97	66.50	58.92
	GEAR (r=2)	53.15	24.80	42.04	38.74	37.12	25.75	**63.02**	66.50	59.07
	HaPPI-KV (r=4)	**56.16**	33.83	**47.25**	38.92	**37.97**	**26.18**	62.52	**66.67**	**59.94**
	HaPPI-KV (r=2)	55.63	32.29	45.10	38.90	37.87	25.89	62.35	63.31	59.62
Qwen2-7B -Instruct	FP	71.65	57.09	50.63	38.42	15.37	25.56	63.76	51.42	59.42
	KCVT-3bit	65.05	56.69	40.47	31.95	16.66	23.40	57.75	18.81	50.80
	KIVI-2bit	57.77	45.67	37.61	31.52	**19.74**	22.45	55.54	15.55	47.77
	PALU-50%	65.43	**57.48**	48.31	38.05	15.72	24.98	63.57	51.00	59.17
	xKV	70.40	53.94	48.37	38.51	15.94	24.84	63.57	51.33	59.11
	GEAR (r=4)	66.11	53.94	46.29	38.51	15.15	24.78	63.69	50.92	**59.36**
	GEAR (r=2)	68.76	53.94	45.54	38.35	15.11	24.73	63.66	51.00	59.11
	HaPPI-KV (r=4)	**72.93**	56.70	**50.07**	**38.59**	15.34	**25.61**	63.74	51.33	59.21
	HaPPI-KV (r=2)	71.10	55.18	49.74	**38.59**	15.11	25.49	**63.79**	**51.67**	59.11

7.1 Experimental Setup

We select three widely used open-source models, Llama3-8B-Instruct (Grattafiori et al., 2024), Mistral-7B (Jiang et al., 2024), and Qwen2-7B (Team, 2024). For these three models, we evaluate diverse tasks across three major categories. For mathematical reasoning, we use GSM8K (Cobbe et al., 2021) and AQuA (Ling et al., 2017); for symbolic reasoning, we adopt BigBench Hard (BBH) (Suzgun et al., 2022). All reasoning experiments employ Chain-of-Thought (CoT) prompting (Wei et al., 2022) in English, which generates extended reasoning traces to evaluate compression under complex workloads.

For long-context understanding, we use LongBench (Bai et al., 2023), which includes tasks such as document understanding, summarization, and question answering, allowing us to test HaPPI-KV's performance in realistic application scenarios. Additionally, we leverage RULER (Hsieh et al., 2024) to systematically analyze accuracy, memory usage, and latency across a range of sequence lengths, verifying that HaPPI-KV consistently delivers effective compression to very long contexts.

All experiments are conducted on LM-Eval-harness (Gao et al., 2023) with NVIDIA A100-SXM4 GPUs (40GB and 80GB). Details of the experimental setup and hyperparameters are provided in the appendix.

7.2 Results on LLM Inference

The results are summarized in Table 1. In CoT reasoning tasks, HaPPI-KV (r=4) achieved state-of-the-art performance across most CoT tasks. On Llama3-8B-Instruct, it reached 78.08% on GSM8K, nearly matching full-precision (78.01%) and outperforming xKV (76.59%). Similar trends hold on AQuA and BBH, where HaPPI-KV (r=4) surpasses xKV and GEAR by 0.1–2.9%p.

In LongBench evaluations, HaPPI-KV maintained competitive or superior performance across all task categories. Notably, on Llama3, HaPPI-KV (r=2) achieved 31.44% on Single Document QA, surpassing even the full-precision baseline (29.93%), suggesting that the compression process may introduce a beneficial regularization effect. These results confirm that HaPPI-KV offers robust performance across diverse reasoning domains without specializing to any particular task type.

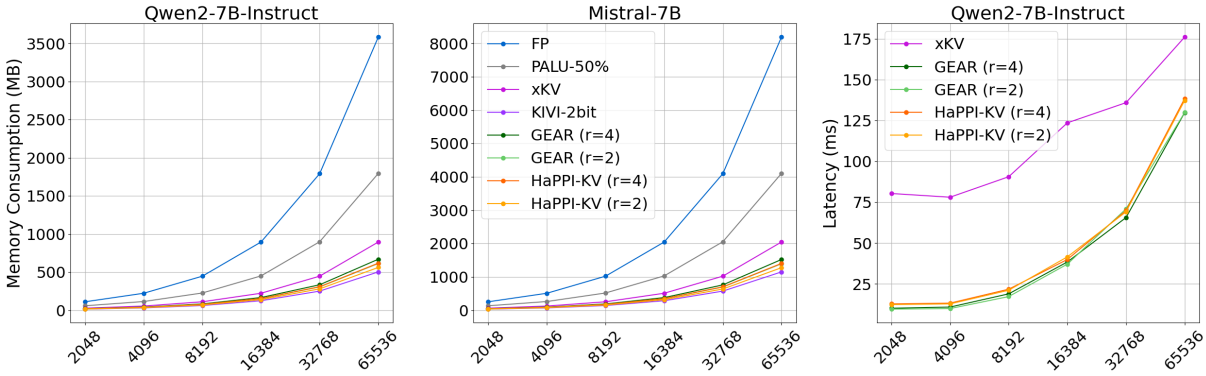

Figure 6: Memory consumption (left and middle) on Qwen2-7B and Mistral-7B, and KV-cache compression latency (right) on Qwen2-7B at each sequence length.

7.3 Results on Various Sequence Lengths

We further evaluated HaPPI-KV on RULER with sequence lengths from 4k to 64k tokens (Figure 1). HaPPI-KV consistently outperforms all methods, with the gap widening at longer contexts. Compared to xKV, HaPPI-KV (r=4) achieves 0.7–1.2%p higher accuracy from 4k to 32k tokens and maintains 53.07% at 64k versus xKV's 52.11%, delivering a 1–3%p improvement over GEAR and xKV under comparable compression ratios.

7.4 Memory and Latency

HaPPI-KV achieves the highest memory efficiency among all evaluated methods at a modest latency premium over GEAR. The trade-off is favorable: a small per-step overhead buys substantially better compression accuracy, which translates directly into higher downstream quality (Table 1, Figure 1).

Memory. We measured KV-cache memory across various sequence lengths on Qwen2-7B and Mistral-7B (Figure 6, left and middle). HaPPI-KV (r=2) is the most memory-efficient across all configurations: at 65,536 tokens on Qwen2-7B, it uses only 562.68 MB, achieving a 6.37× compression ratio (84.30% savings) over the FP16 baseline (3,584 MB) and 8.4% smaller than the strongest competitor, GEAR (r=2). While all methods grow proportionally with sequence length, HaPPI-KV exhibits a shallower growth slope, confirming its scalability to long contexts.

Latency. HaPPI-KV incurs slightly higher overhead than GEAR (e.g., 11.83 ms vs. 10.83 ms at 2,048 tokens on Qwen2-7B). This added cost reflects the whitening and PCA-initialization steps and is the price for HaPPI-KV's accuracy advantage. Detailed latency measurements are provided in Appendix D.

8 Conclusion

In this work, we proposed HaPPI, a novel Hadamard-based power iteration algorithm that significantly improves the accuracy of truncated SVD approximation. Building upon this foundation, we further introduced HaPPI-KV, an advanced KV-cache compression framework that combines key whitening and residual quantization to achieve state-of-the-art trade-offs between memory savings, model quality, and latency. Extensive experiments on multiple large language models and diverse benchmarks demonstrated that our method consistently outperforms existing approaches across a wide range of settings. We believe that HaPPI and HaPPI-KV pave the way for future research on scalable and efficient model compression, facilitating the practical deployment of large language models under stringent resource constraints.

Limitations

We acknowledge several limitations of this work. First, all experiments are conducted on 7–8B parameter models due to computational constraints. Since attention patterns and outlier distributions may differ at larger scales, the generalization of HaPPI-KV to models with 70B+ parameters remains an important direction for future investigation. Second, while HaPPI-KV demonstrates strong performance on standard decoder-only transformer architectures, its applicability to Multi-Head Latent Attention (MLA)-based models has not been empirically evaluated. Nonetheless, since HaPPI-KV effectively mitigates the outlier features that are commonly observed across transformer architectures, we expect it to remain effective in the MLA setting as well. Finally, the key whitening step requires computing the covariance matrix $K^T K$, which in-

troduces additional overhead that, although modest in practice, may become more significant for extremely long sequences.

References

Zeyuan Allen-Zhu and Yuanzhi Li. 2016. Lazysvd: Even faster svd decomposition yet without agonizing pain. In *Advances in neural information processing systems*, volume 29.

Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. 2024. Quarot: Outlier-free 4-bit inference in rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240.

Saleh Ashkboos, Mahdi Nikdan, Soroush Tabesh, Roberto L Castro, Torsten Hoefler, and Dan Alistarh. 2025. Halo: Hadamard-assisted lower-precision optimization for llms. *arXiv preprint arXiv:2501.02625*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2023. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*.

Chi-Chih Chang, Chien-Yu Lin, Yash Akhauri, Wei-Cheng Lin, Kai-Chiang Wu, Luis Ceze, and Mohamed S Abdelfattah. 2025. xkv: Cross-layer svd for kv-cache compression. *arXiv preprint arXiv:2503.18893*.

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

Patrick Chen, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2021. Drone: Data-aware low-rank compression for large nlp models. *Advances in neural information processing systems*, 34:29321–29334.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, and 1 others. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2023. A framework for few-shot language model evaluation.

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

Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. 2011. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. 2024. Kvquant: towards 10 million context length llm inference with kv cache quantization. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, pages 1270–1303.

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

Yen-Chang Hsu, Ting Hua, Sungen Chang, Qian Lou, Yilin Shen, and Hongxia Jin. 2022. Language model compression with weighted low-rank factorization. *arXiv preprint arXiv:2207.00112*.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, and 1 others. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Hao Kang, Qingru Zhang, Souvik Kundu, Geonhwa Jeong, Zaoxing Liu, Tushar Krishna, and Tuo Zhao. 2024. Gear: An efficient kv cache compression recipe for near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*.

Seonggon Kim, Juncheol Shin, Seung-taek Woo, and Eunhyeok Park. 2025. Hot: Hadamard-based optimized training. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 4787–4796.

Bokai Lin, Zihao Zeng, Zipeng Xiao, Siqi Kou, Tianqi Hou, Xiaofeng Gao, Hao Zhang, and Zhijie Deng. 2024a. Matryoshkakv: Adaptive kv compression via trainable orthogonal projection. *arXiv preprint arXiv:2410.14731*.

Haokun Lin, Haobo Xu, Yichen Wu, Jingzhi Cui, Yingtao Zhang, Linzhan Mou, Linqi Song, Zhenan Sun, and Ying Wei. 2024b. Duquant: Distributing outliers via dual transformation makes stronger quantized llms. *Advances in Neural Information Processing Systems*, 37:87766–87800.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. *arXiv preprint arXiv:1705.04146*.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *Advances in neural information processing systems*, 36:34892–34916.

Zechun Liu, Changsheng Zhao, Igor Fedorov, Bilge Soran, Dhruv Choudhary, Raghuraman Krishnamoorthi, Vikas Chandra, Yuandong Tian, and Tijmen Blankevoort. 2024a. Spinquant: Llm quantization with learned rotations. *arXiv preprint arXiv:2405.16406*.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. 2024b. Kivi: a tuning-free asymmetric 2bit quantization for kv cache. In *Proceedings of the 41st International Conference on Machine Learning*, pages 32332–32344.

Fanxu Meng, Zhaohui Wang, and Muhan Zhang. 2024. Pissa: Principal singular values and singular vectors adaptation of large language models. *Advances in Neural Information Processing Systems*, 37:121038–121072.

Cameron Musco and Christopher Musco. 2015. Randomized block krylov methods for stronger and faster approximate singular value decomposition. In *Advances in neural information processing systems*, volume 28.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, and 1 others. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. *arXiv preprint arXiv:2210.09261*.

Yaya Sy, Christophe Cerisara, and Irina Illina. 2024. Lillama: Large language models compression via low-rank feature distillation. *arXiv preprint arXiv:2412.16719*.

James Joseph Sylvester. 1867. Lx. thoughts on inverse orthogonal matrices, simultaneous signsuccessions, and tessellated pavements in two or more colours, with applications to newton's rule, ornamental tile-work, and the theory of numbers. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 34(232):461–475.

Qwen Team. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2.

Albert Tseng, Jerry Chee, Qingyao Sun, Volodymyr Kuleshov, and Christopher De Sa. 2024. Quip# even better llm quantization with hadamard incoherence and lattice codebooks. In *Proceedings of the 41st International Conference on Machine Learning*, pages 48630–48656.

Thijs Vogels, Sai Praneeth Karimireddy, and Martin Jaggi. 2019. Powersgd: Practical low-rank gradient compression for distributed optimization. *Advances in Neural Information Processing Systems*, 32.

Xin Wang, Samiul Alam, Zhongwei Wan, Hui Shen, and Mi Zhang. 2025a. Svd-llm v2: Optimizing singular value truncation for large language model compression. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4287–4296.

Xin Wang, Yu Zheng, Zhongwei Wan, and Mi Zhang. 2024. Svd-llm: Truncation-aware singular value decomposition for large language model compression. *arXiv preprint arXiv:2403.07378*.

Yixuan Wang, Haoyu Qiao, Lujun Li, Qingfu Zhu, and Wanxiang Che. 2025b. Commonkv: Compressing kv cache with cross-layer parameter sharing. *arXiv preprint arXiv:2508.16134*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837.

Jeffrey TH Wong, Cheng Zhang, Xinye Cao, Pedro Gimenes, George A Constantinides, Wayne Luk, and Yiren Zhao. 2025. A3: an analytical low-rank approximation framework for attention. *arXiv preprint arXiv:2505.12942*.

Haocheng Xi, Changhao Li, Jianfei Chen, and Jun Zhu. 2023. Training transformers with 4-bit integers. *Advances in Neural Information Processing Systems*, 36:49146–49168.

Jingyang Xiang and Sai Qian Zhang. 2024. Dfrot: Achieving outlier-free and massive activation-free for rotated llms with refined rotation. *arXiv preprint arXiv:2412.00648*.

Xianglong Yan, Zhiteng Li, Tianao Zhang, Linghe Kong, Yulun Zhang, and Xiaokang Yang. 2025. Recalkv: Low-rank kv cache compression via head reordering and offline calibration. *arXiv preprint arXiv:2505.24357*.

Yibo Yang, Xiaojie Li, Zhongzhu Zhou, Shuaiwen Song, Jianlong Wu, Liqiang Nie, and Bernard Ghanem. 2024. Corda: Context-oriented decomposition adaptation of large language models for task-aware parameter-efficient fine-tuning. *Advances in Neural Information Processing Systems*, 37:71768–71791.

Hong Yankun, Li Xing, Zhen Hui-Ling, Yu Xianzhi, Liu Wulong, and Yuan Mingxuan. 2025. Svdq: 1.25-bit and 410x key cache compression for llm attention. *arXiv preprint arXiv:2502.15304*.

Zhihang Yuan, Yuzhang Shang, Yue Song, Qiang Wu, Yan Yan, and Guangyu Sun. 2023. Asvd: Activation-aware singular value decomposition for compressing large language models. *arXiv preprint arXiv:2312.05821*.

Yuxuan Yue, Zhihang Yuan, Haojie Duanmu, Sifan Zhou, Jianlong Wu, and Liqiang Nie. 2024. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*.

Rongzhi Zhang, Kuang Wang, Liyuan Liu, Shuohang Wang, Hao Cheng, Chao Zhang, and Yelong Shen. 2024. Lorc: Low-rank compression for llms kv cache with a progressive compression strategy. *arXiv preprint arXiv:2410.03111*.

Appendix

A Preliminaries

A.1 Singular Value Decomposition

SVD is a fundamental tool for low-rank approximation, where any matrix $A \in \mathbb{R}^{m\times n}$ can be decomposed as $A = U\Sigma V^T$ with $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ orthonormal, and $\Sigma = \mathrm{diag}(s) \in \mathbb{R}^{m\times n}$ containing singular values in descending order. If the top r singular values dominate, A can be approximated by $A_r = U_{[:,1:r]}\Sigma_{1:r,1:r}V^T_{[:,1:r]}$, with the error given by $\|A - A_r\|_F^2 = \sum_{i=r+1}^{\min(m,n)} s_i^2$.

In many recent studies, the idea of truncated SVD has been widely adopted, requiring only the top r singular components. However, applying the exact SVD incurs a computational cost of $O(mn^2)$, as the entire decomposition must be computed before extracting the relevant components.

A.2 Hadamard Transformation

Many recent studies (Xi et al., 2023; Lin et al., 2024b; Tseng et al., 2024; Kim et al., 2025; Ashkboos et al., 2024; Liu et al., 2024a; Ashkboos et al., 2025; Xiang and Zhang, 2024) empirically demonstrated that HT is highly beneficial for mitigating the influence of activation outliers by transforming the data into the frequency domain. From a similar perspective, our method also adopts the Hadamard Transform (HT; (Sylvester, 1867)) as a core building block of HaPPI.

HT is a discrete Fourier-like transform that linearly projects a vector into the frequency domain. Given a vector $v \in \mathbb{R}^n$ with $n = 2^d$, its frequency-domain representation $\tilde{v} \in \mathbb{R}^n$ is obtained by multiplying it with the Walsh–Hadamard matrix H_d as $\tilde{v} = H_d \cdot v$. Here, the Walsh–Hadamard matrix H_d is defined recursively as:

$$H_1 = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}, \quad H_n = H_1 \otimes H_{n-1}, \quad (4)$$

where $\otimes$ denotes the Kronecker product, and H_d is a $2^d \times 2^d$ orthogonal matrix satisfying $H_d H_d^T = H_d^T H_d = I$. This orthogonality guarantees the lossless reconstruction of the transformed vector.

B Detailed Theoretical Analysis

B.1 Proof of Proposition 1

Proof. Consider a row a_i whose energy is concentrated in at most k entries. Let the dominant entries have indices $\mathcal{S}$ with $|\mathcal{S}| = k$, so that $\sum_{p\in\mathcal{S}} a_{ip}^2$ accounts for the dominant fraction of $\|a_i\|^2$.

The elements of the transformed row in A_H are given by $(A_H)_{ij} = \frac{1}{\sqrt{n}}\sum_p a_{ip}h_{pj}$. Since the energy is concentrated in k entries, the sum is dominated by the k terms in $\mathcal{S}$: $|(A_H)_{ij}| \le \frac{1}{\sqrt{n}}\sum_{p\in\mathcal{S}}|a_{ip}| \le \frac{\sqrt{k}}{\sqrt{n}}\sqrt{\sum_{p\in\mathcal{S}} a_{ip}^2}$, where the last step follows from Cauchy–Schwarz. Bounding each entry by the matrix-wise maximum, $|a_{ip}| \le \|A\|_{\max}$, gives $\sqrt{\sum_{p\in\mathcal{S}} a_{ip}^2} \le \sqrt{k}\,\|A\|_{\max}$, and therefore $\|A_H\|_{\max} \le \frac{k}{\sqrt{n}}\|A\|_{\max}$.

Applying this to the definition of γ and using $\|A_H\|_F = \|A\|_F$ (orthogonality of H):

$$\gamma(A_H) = \frac{mn\|A_H\|_{\max}^2}{\|A_H\|_F^2} \le \frac{k^2}{n}\cdot\frac{mn\|A\|_{\max}^2}{\|A\|_F^2} = \frac{k^2}{n}\gamma(A) \quad (5)$$

In the extreme case where each row is dominated by a single outlier ($k = 1$), this reduces to $\gamma(A_H) \approx \frac{1}{n}\gamma(A)$, confirming that the Hadamard transform effectively suppresses outlier-induced instability. □

B.2 Justification of κ_{eff}

We justify the definition $\kappa_{\text{eff}}(A) = \kappa(A)\cdot\gamma(A)$. In finite-precision arithmetic, a single matrix-vector product $y = Ap$ incurs a per-element rounding error bounded by $|\delta y_i| \le \epsilon_{\text{mach}}\cdot n\cdot\|A\|_{\max}\cdot\|p\|_\infty$. For matrices with large outliers, $\|A\|_{\max}$ can be much larger than the typical element magnitude $\|A\|_F/\sqrt{mn}$. The ratio between these quantities is precisely $\sqrt{\gamma(A)}$.

Over L power iterations, each iteration amplifies errors from the previous step. The standard condition number $\kappa(A) = \sigma_1/\sigma_r$ governs this amplification: small singular values are relatively more affected by absolute errors. Combining both effects, the total accumulated error after L iterations scales as:

$$\|\delta\|_{\text{total}} = O(\epsilon_{\text{mach}}\cdot L\cdot\kappa(A)\cdot\gamma(A)) = O(\epsilon_{\text{mach}}\cdot L\cdot\kappa_{\text{eff}}(A)) \quad (6)$$

This motivates the product form and explains why matrices with both high condition numbers and large outliers (common in KV caches) suffer the most from numerical errors in standard power iteration.

B.3 PCA Initialization Convergence Analysis

Standard power iteration with random initialization requires the iterates to "discover" the principal subspace from scratch. For standard power iteration

on A with random initialization, after L iterations the error angle satisfies:

$$\sin(\theta_L^{\text{PI}}) = O\left(\left(\frac{\sigma_2}{\sigma_1}\right)^L\right) \tag{7}$$

HaPPI initializes using approximate principal components from the covariance matrix $C_H = A_H^T A_H$. Since C_H has eigenvalues σ_i^2, each power iteration on C_H converges at rate $(\sigma_2/\sigma_1)^2$ rather than σ_2/σ_1. After 2 iterations on C_H, the initial error angle satisfies:

$$\sin(\theta_0^{\text{HaPPI}}) = O\left(\left(\frac{\sigma_2}{\sigma_1}\right)^4\right) \tag{8}$$

After L subsequent iterations on A_H, HaPPI achieves:

$$\sin(\theta_L^{\text{HaPPI}}) = O\left(\left(\frac{\sigma_2}{\sigma_1}\right)^{L+4}\right) \tag{9}$$

This shows that HaPPI's PCA initialization provides an effective "head start" of 4 iterations worth of convergence compared to random initialization. The analysis above is presented for the top-1 case for clarity; for rank-r subspace iteration, the convergence rate is governed by the spectral gap σ_{r+1}/σ_r between the retained and discarded singular values, so the same argument generalizes by replacing σ_2/σ_1 with σ_{r+1}/σ_r, yielding the $O\big((\sigma_{r+1}/\sigma_r)^{L+4}\big)$ convergence term used in the main-text error bound.

B.4 Standard Power Iteration Error Bound

For comparison, the standard power iteration error bound is:

$$\begin{aligned}\|A - \hat{A}_r^{\text{PI}}\|_F \le \|A - A_r\|_F \\ + O\left(\left(\frac{\sigma_{r+1}}{\sigma_r}\right)^L\right) \\ + O(\epsilon_{\text{mach}} \cdot L \cdot \kappa(A) \cdot \gamma(A))\end{aligned} \tag{10}$$

Compared to HaPPI (Eq. in the main text), standard power iteration has: (1) a weaker convergence exponent of L instead of $L+4$, and (2) a numerical error term that is up to n/k^2 times larger due to the absence of Hadamard-based outlier suppression.

C Overhead Analysis

We analyze the computational overhead of HaPPI in terms of FLOPs, with the results summarized in Fig. 2. Experiments were conducted using a tensor of size (1, 8, 2048, 128), simulating the scenario of processing a sequence of 2048 tokens in Llama3-8B. The number of iterations for all methods except SVD was fixed at two. As shown in the figure, HaPPI requires 0.35 GFLOPs, which is slightly higher than GEAR's power iteration (0.12 GFLOPs) but significantly lower than SVD low-rank (0.66 GFLOPs) and full SVD (1.24 GFLOPs).

This additional cost primarily arises from two sources: HT and the covariance matrix computation. HT introduces minimal overhead using the Fast Walsh–Hadamard Transform, while the primary computational cost comes from computing the covariance matrix $C_H = A_H^T A_H$, which accounts for most of the additional 0.23 GFLOPs. However, as shown by the latency measurements in Fig. 6, despite the FLOPs difference, HaPPI's overall compression time is lower than that of torch.svd_lowrank and close to the latency of power iteration.

D Latency Comparison

To evaluate computational efficiency alongside memory savings, we measured compression latency for the Qwen2-7B-Instruct attention mechanism, as shown in Fig. 6 (right).

HaPPI-KV (r=4) exhibited latency comparable to existing power-iteration-based approaches. At 2,048 tokens, HaPPI-KV recorded 11.83 ms, nearly identical to GEAR (10.83 ms). At 32,768 and 65,536 tokens, HaPPI-KV recorded 81.43 ms and 148.26 ms, respectively, only about 1% slower than GEAR (80.96 ms and 142.8 ms).

To provide a comprehensive view of practical overhead, we also measured end-to-end inference latency of Llama3-8B on GSM8K (sequence length $\approx$ 4096, rank=4) on an A100 GPU. The results show that HaPPI-KV incurs minimal overhead: FP16 baseline takes 7.31s for generation, while HaPPI-KV adds only 1.82s for compression (total 9.13s), comparable to GEAR's 1.78s compression overhead (total 9.09s). This demonstrates that HaPPI-KV's accuracy gains come at negligible additional cost in real-world deployment scenarios.

Overall, HaPPI-KV achieves nearly equivalent computational efficiency to power-iteration-based methods while delivering significant improvements in both accuracy and memory efficiency. The additional overhead from the Hadamard transform and covariance matrix computation constitutes only a minor fraction of the total cost, demonstrating HaPPI-KV as a practical and scalable solution.

Table 2: Hyperparameters of Methods.

Parameters	KCVT-nbit	KIVI-nbit	PALU-50%	GEAR-r=m,Q=nbit	HaPPI-r=m,Q=nbit
Quant. Bit	n	n	N/A	n	n
Group Size	64	64	N/A	64	64
Key Rank	N/A	N/A	96	m	m
Value Rank	N/A	N/A	96	m	m
Loop	N/A	N/A	3	3	3
Left	N/A	N/A	N/A	0.01	N/A
Streaming	True	True	True	True	True
Streaming Gap	64	64	64	64	64

Table 3: Description of Hyperparameters.

Parameter	Description
Quant. Bit	Quantize bit.
Group Size	Quantize group size.
Key Rank	Rank compared with smaller dimension set to K cache.
Value Rank	Rank compared with smaller dimension set to V cache.
Loop	Number of SVD solver iterations (Power-iteration).
Left	Outlier extraction part compared with total matrix.
Streaming	Use streaming mode.
Streaming Gap	Iteration length for re-compression.

E Experiment Hyperparameters

Tab. 2 and 3 present the hyperparameters of the methods considered in our experiments. In Tab. 2, n denotes the quantization bit and m denotes the low-rank rank applied to the key and value tensors. In our LLM inference experiments, we set $n \in \{2, 3\}$ and $m \in \{2, 4, 8\}$, whereas the LoRA initialization experiment in Tab. 5 uses $m = 16$. PALU-50% does not employ quantization and therefore does not define either a quantization bit or a group size. Conversely, KCVT-nbit and KIVI-nbit are quantization-only methods without low-rank approximation, and thus do not specify key rank, value rank, or loop parameters. The Left parameter denotes the fraction of the matrix reserved for outlier extraction; for instance, a value of 0.01 indicates that 1% of the matrix is allocated for this purpose. The Streaming parameter governs the compression of newly generated KV caches. Specifically, the cache is retained in full precision for the duration of the Streaming Gap, after which compression is applied using the designated method. If Streaming is disabled, no compression is applied to the newly generated KV caches. For xKV, we use rank 128 with group size 64.

F Vision Language Model Captioning Task

Fig. 7 presents a qualitative evaluation of caption generation on a sample image using the LLaVA-1.5-7B model (Liu et al., 2023) with two KV cache compression methods: GEAR (r=2, Q=2bit) and HaPPI-KV (r=2, Q=2bit). At the beginning of the generation process, both GEAR and HaPPI-KV produce nearly identical captions, suggesting that the information is well preserved under compression. However, as the generation proceeds, GEAR tends to induce hallucinations, as illustrated in the figure, whereas HaPPI-KV consistently generates accurate and faithful captions.

G Ablation Study

To systematically analyze the impact of each component of HaPPI-KV on overall performance, we conducted an ablation study using the Mistral-7B model and GSM8K-CoT benchmark.

As shown in Table 4, the baseline GEAR method showed a 2.8% performance degradation compared to the FP baseline. Interestingly, when the outlier-preserving strategy was removed from GEAR, performance further declined by 1.28% to 51.87%, suggesting that outlier handling plays a crucial role in KV cache compression.

In the experiment where power iteration was replaced with HaPPI, we achieved 53.1% accuracy, confirming a 1.23% performance improvement over the baseline power iteration. This demonstrates that HaPPI's improved low-rank approximation capability is effective in actual downstream tasks. Furthermore, analyzing the impact of component order in the compression pipeline, we found

Table 4: Ablation study on Mistral-7B with GSM8K-CoT benchmark in r=2, q=2 environment.

Method	Original	Residual	Additional	Accuracy
FP	-	-	-	55.95
GEAR	Quant.	Power Iteration	Outlier preserving	53.15
	Quant.	Power Iteration	-	51.87
	Quant.	HaPPI	-	53.10
	HaPPI	Quant.	-	54.13
HaPPI-KV	HaPPI	Quant.	Key whitening	55.63

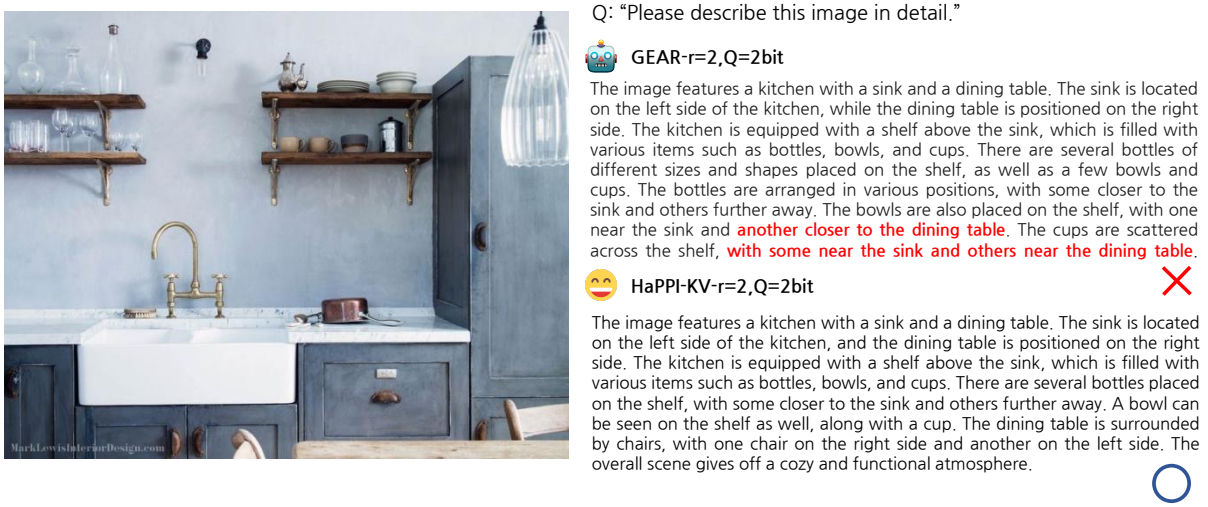

Figure 7: Image captioning results on LLaVA-1.5-7B with GEAR and HaPPI compression methods.

that the strategy of applying HaPPI first followed by quantization achieved 54.13% accuracy, showing superior results compared to the reverse order. This indicates that performing high-quality low-rank approximation first and then applying quantization to the residuals is more effective.

Finally, the complete HaPPI-KV pipeline with key whitening achieved a final performance of 55.63%, obtaining an additional 1.5% performance improvement from key whitening alone. This confirms that the preprocessing step of normalizing the outlier distribution of the key tensor makes a substantial contribution to overall compression quality. Overall, each component cumulatively contributes to performance improvement, ultimately achieving 55.63%, which is nearly equivalent to FP performance, demonstrating the efficacy of the proposed methodology.

H LoRA Initialization Task

To validate HaPPI's accurate low-rank approximation capability, we evaluated its performance on LoRA adapter initialization tasks. Recent studies have reported that LoRA initialization using truncated SVD significantly improves fine-tuning performance, and particularly in methods such as PiSSA, performance differences exist between exact SVD and approximate SVD.

The experimental results shown in Table 5 represent accuracy across various tasks performed on the Llama3-8B-Instruct model with rank-16 settings. The baseline LoRA methodology recorded an average accuracy of 44.19%, while PiSSA with SVD-based initialization showed a 1.7% performance improvement to 45.89%.

A notable observation is that when PiSSA used approximate SVD instead of exact SVD, performance declined by 0.81% to 45.08%. This demonstrates that SVD approximation errors directly impact actual downstream task performance. In contrast, when HaPPI was applied, we achieved an average accuracy of 46.79%, surpassing all baseline methods. In particular, HaPPI achieved 70.8% on GSM8K, 13.01% on MATH, 51.93% on HumanEval, and 53.02% on IFeval, achieving the highest performance on most individual tasks.

These results demonstrate that HaPPI provides high-quality low-rank approximation that leads to downstream performance improvements beyond simply reducing numerical approximation errors. In particular, the 0.9% performance improvement over PiSSA clearly shows the importance of accurate SVD approximation and the practical value of the HaPPI algorithm.

Table 5: Experimental accuracy (%) results of LoRA finetuning on Llama3-8B-Instruct. Note that rank is 16 and loop count for PiSSA (SVD approx.) is 2.

Method	GSM8K	MATH	HumanEval	MBPP	IFeval	Average
LoRA	69.07	12.58	46.96	43.16	49.18	44.19
PiSSA	69.67	**13.20**	49.75	**46.45**	50.36	45.89
PiSSA (SVD low-rank)	69.21	12.58	49.40	45.21	49.02	45.08
HaPPI	**70.80**	13.01	**51.93**	45.21	**53.02**	**46.79**

I Licenses for Models and Datasets

We list the licenses of all models and datasets used in our experiments.

Models.

- **Llama3-8B-Instruct** (Grattafiori et al., 2024): Released under the Meta Llama 3 Community License Agreement. Commercial use is permitted; services exceeding 700 million monthly active users require a separate license from Meta.
- **Mistral-7B** (Jiang et al., 2024): Released under the Apache License 2.0.
- **Qwen2-7B** (Team, 2024): Released under the Apache License 2.0.

Datasets and Benchmarks.

- **GSM8K** (Cobbe et al., 2021): Released under the MIT License.
- **AQuA-RAT** (Ling et al., 2017): Released under the Apache License 2.0.
- **BigBench Hard (BBH)** (Suzgun et al., 2022): Released under the MIT License.
- **LongBench** (Bai et al., 2023): No explicit license is stated in the repository or dataset card. We use this benchmark solely for non-commercial academic research purposes, consistent with the authors' intended use.
- **RULER** (Hsieh et al., 2024): Released under the Apache License 2.0 (Copyright © 2024 NVIDIA Corporation).