ContrastKV: Robust KV Cache Eviction via Contrastive Signal Fusion for Multi-Query Generalization

Xingchi Chen[1,†], Peiyuan Zong[1,†], Ziqiang Gao[2,§], Qing Li[1,*], Yong Jiang[3], Fa Zhu[4], Hui Li[2]
[1]Pengcheng Laboratory, [2]Dalian Maritime University,
[3]Tsinghua Shenzhen International Graduate School, Tsinghua University, [4]Nanjing Forestry University
{chenxch01, zongpy, liq}@pcl.ac.cn

Abstract

Large Language Models (LLMs) face significant memory and latency overheads during inference due to KV cache grows with the context length. This issue is especially pronouced in Knowledge Base Question Answering (KBQA) settings that require support for multiple downstream queries. Query-aware eviction methods do not generalize across queries, while existing query-agnostic approaches rely on a single proxy query, leading to fragile eviction decisions under high eviction ratios. We propose **ContrastKV**, a robust query-agnostic KV cache eviction algorithm for *multi-query generalization*. ContrastKV introduces a *contrastive signal fusion* mechanism that jointly exploits complementary semantic and structural signals. By contrasting semantic consistency with structural robustness, the method constructs a more reliable eviction criterion that alleviates the blind spots of single-query proxies. The framework integrates efficient signal generation, parallel importance scoring, and multi-level fusion across heads and layers. Experiments show that ContrastKV outperforms state-of-the-art methods, retaining up to 92% accuracy with only 20% of the KV cache budget, while reducing decoding latency by approximately 50% and significantly lowering GPU memory usage.

1 Introduction

Large language models (LLMs) (Achiam et al., 2023; Grattafiori et al., 2024; Yang et al., 2025a; Kong et al., 2025) have demonstrated remarkable performance across a wide range of applications, including machine translation (Feng et al., 2025b), content generation (Venkatraman et al., 2025), and complex reasoning tasks such as coding (Coignion et al., 2024; Zhang et al., 2025b) and mathematics (Setlur et al., 2024; Gao et al., 2025). Despite these advances, the practical deployment of LLMs remains constrained by the substantial memory overhead incurred during long-context inference, which is dominated by the storage of key–value (KV) caches (Bai et al., 2024). For example, processing a 120K-token input with the Qwen2.5-14B model requires approximately 33 GB of KV cache memory, exceeding its 28 GB parameter footprint. This imbalance motivates a growing line of research on KV cache eviction (Zhang et al., 2023; Xu et al., 2025; Li et al., 2024a; Kim et al., 2025), which aims to reduce memory consumption and decoding latency, thereby enabling more cost-effective and responsive inference.

Among long-context applications, knowledge base question answering (KBQA) represents a particularly challenging and representative scenario. In KBQA, LLMs first ingest a large knowledge base to acquire domain-specific or up-to-date information, after which users issue queries grounded in this context. In this work, we use the term *knowledge base* broadly to refer not only structured knowledge (Lan et al., 2022; Li et al., 2024b), but also unstructured corpora such as code repositories, document collections, and historical dialogue logs. A common characteristic of these settings is that the knowledge base is orders of magnitude longer than the subsequent queries and answers, leading to large KV caches whose contents must be reused across multiple downstream queries. However, most existing KV cache eviction methods are query-aware, making eviction decisions conditioned on a specific query and thus unsuitable for KBQA. Although some query-agnostic methods have been proposed, they typically suffer from severe performance degradation under high eviction ratios, limiting their practicality.

To better understand this limitation, we re-examine existing query-agnostic eviction strategies and identify a fundamental blind spot: their reliance on a single, generic proxy query to approximate an ideal eviction criterion. Such an approxi-

[†]The first two authors have equal contribution.
*Corresponding author: Qing Li. [†]This work was done during an internship at Pengcheng Laboratory.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9216–9229
July 2-7, 2026

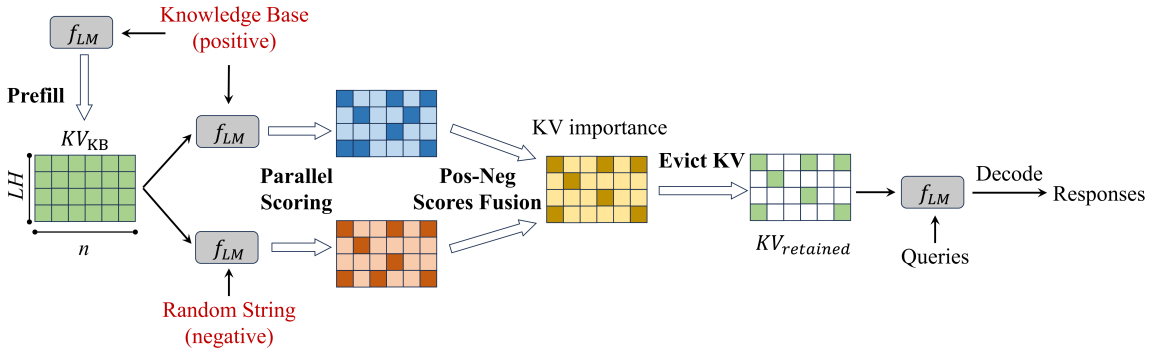

Figure 1: Overview of ContrastKV. After the prefill stage, ContrastKV proceeds in three steps: (1) *Contrastive Signal Generation*, which constructs a positive signal from the original knowledge base and a negative signal from random strings; (2) *Parallel Scoring*, which independently computes attention-based importance scores for both signals; and (3) *Positive–Negative Fusion*, which combines the scores via a multi-level strategy to identify KV pairs with strong generalization potential.

mation fails to capture the diversity of downstream queries, resulting in brittle decision boundaries and poor generalization. Motivated by principles from ensemble learning (Domingos, 2000; Cao et al., 2019; Ganaie et al., 2022; Fu et al., 2025a), particularly the role of diversity in reducing generalization error (KROGH, 1995), we propose to introduce *contrastive signals* into KV cache eviction. By explicitly contrasting semantic and structural signals, we aim to maximize the discriminability across diverse query behaviors, thereby reducing overall generalization error and enabling the identification of KV pairs that are robust to query variation.

Building on this insight, we propose **ContrastKV**, a query-agnostic KV cache eviction algorithm tailored for KBQA scenarios. As illustrated in Figure 1, ContrastKV is centered around a contrastive evaluation framework that jointly considers positive and negative signals. The algorithm operates in two fusion stages. At the head level, ContrastKV identifies strong signal positions for each attention head and dynamically allocates retention budgets to prevent the loss of head-specific critical information. At the layer level, it transforms the ranking distribution induced by negative signals into gain coefficients, which are used to refine the positive scores and preserve potentially valuable KV pairs. Through this dual-level contrastive fusion, ContrastKV constructs a more robust and generalizable eviction criterion, enabling effective KV reuse across diverse queries arising from multi-turn or multi-user KBQA interactions.

We evaluate ContrastKV on multiple KBQA benchmarks using three representative LLMs, i.e., Qwen2.5-7B, Llama3.1-8B and Qwen2.5-14B. Experimental results show that ContrastKV achieves substantial KV cache compression while maintaining strong task performance. In particular, when retaining only 20% of the KV cache budget, ContrastKV preserves 92% of the original accuracy, outperforming state-of-the-art query-agnostic methods by 22%. Moreover, ContrastKV reduces decoding latency by approximately 50% under the same budget and yields near-linear GPU memory savings as the budget decreases.

Overall, this work makes three key contributions: (1) we identify and analyze the generalization bottleneck of existing query-agnostic KV cache eviction methods in KBQA, (2) we propose ContrastKV, a novel contrastive signal fusion framework that enables robust KV cache eviction across diverse queries, and (3) we demonstrate through extensive experiments that ContrastKV significantly improves the accuracy–efficiency trade-off, facilitating practical long-context inference in resource-constrained environments.

2 Related Works

For large language models (LLMs) based on the Transformer (Vaswani et al., 2017; Dao et al., 2022) architecture, the KV cache is a core component of autoregressive generation. However, as model size and sequence length increase, the memory footprint of the KV cache gradually becomes a performance bottleneck (Xiao et al., 2024; Liu et al., 2025). A large body of research focuses on evicting the KV cache to improve the inference efficiency of LLMs.

Early studies on KV cache eviction mainly examine the feasibility of eviction the KV cache (Zhang et al., 2023; Liu et al., 2023). After that, most pro-

posed algorithms are query-aware. Query-aware eviction methods evict the KV cache based on the semantic information of the current input query and prioritize tokens that are highly relevant to the query. For example, cache eviction can be guided by an observation window (Li et al., 2024a; Zhang et al., 2025a). Building on this idea, researchers further propose improved query-dependent methods, such as pyramid-style hierarchical eviction (Cai et al., 2024; Yang et al., 2024; Tang et al., 2025) and head-level dynamic eviction (Fu et al., 2025b; Feng et al., 2025a). However, although query-dependent methods preserve important information accurately, they rely heavily on the input query and are difficult to generalize across different tasks or contextual settings (Li et al., 2025).

Recently, some studies explore query-agnostic eviction methods (Corallo et al., 2025; Yang et al., 2025b; Chari and Van Durme, 2025). These methods generate reusable evicted caches in a single pass and address the performance degradation of query-aware methods caused by overfitting to the initial query in multi-query scenarios. Among existing methods, KVzip (Kim et al., 2025) represents the current state of the art. The core contribution of KVzip lies in validating the feasibility and effectiveness of the importance evaluation mechanism based on knowledge base reconstruction.

3 Problem Formulation

In the prefill stage of LLMs, the KV cache generated from a knowledge base of length n can be represented as $\mathrm{KV}_{KB} \in \mathbb{R}^{n \times d}$. Since queries are unavailable during this stage, query-agnostic KV cache eviction requires reconstructing important context information without task-specific guidance. During the decoding stage, the model must process diverse queries $q \in \mathcal{Q}$ that were unknown during prefill. To reduce memory and computational overhead, KV cache eviction algorithms aim to retain only a subset $\mathrm{KV}_{\text{retained}} \subset \mathrm{KV}_{KB}$ of size $k \ll n$, such that:

$$f_{\text{LM}}(q \mid \mathrm{KV}_{\text{retained}}) \approx f_{\text{LM}}(q \mid \mathrm{KV}_{KB}), \quad \forall q \in \mathcal{Q} \tag{1}$$

where f_{LM} represents the model's generation quality. The fundamental challenge is that the optimal subset $\mathrm{KV}^{*}_{\text{retained}}$ depends on the unknown query distribution $\mathcal{Q}$.

Existing query-agnostic approaches approximate the ideal by identifying important KV pairs using a single proxy query $\hat{q}$ that aims to represent $\mathcal{Q}$. Let $\mathcal{K}_{\hat{q}} \subset \mathrm{KV}_{KB}$ denote the KV cache subset identified as important by $\hat{q}$. The objective becomes:

$$\mathcal{K}_{\hat{q}} = \mathrm{Top}_k\left(\mathrm{Importance}(\mathrm{KV}_{KB}, \hat{q})\right) \tag{2}$$

where $\mathrm{Importance}(\cdot)$ computes a scoring function and Top_k selects the top-k highest scores. However, different proxy queries (e.g., chat templates q_{temp} (Xiao et al., 2024), task descriptions q_{task} (Corallo et al., 2025), or reconstruction prompts q_{rec}) (Kim et al., 2025) capture different aspects of importance, leading to:

$$\exists q \in \mathcal{Q} : \mathcal{K}_{\hat{q}} \not\supseteq \mathcal{K}_q \tag{3}$$

where $\mathcal{K}_q$ represents the truly important KV pairs for query q. This **blind spot problem** arises because a single query $\hat{q}$ cannot adequately approximate the diverse information needs across $\mathcal{Q}$.

Current state-of-the-art query-agnostic methods like **KVzip** (Kim et al., 2025) effectively compress KV caches across multi-task scenarios by employing knowledge base reconstruction as a universal query. However, empirical studies reveal significant performance degradation when compression becomes aggressive. Specifically, when the KV cache retention ratio drops below 30% (see Figure 2), KVzip fails to maintain generalization stability. The retained cache $\mathrm{KV}_{\text{retained}}$ exhibits substantial variance in its ability to support diverse downstream queries, violating the consistency requirement in Eq. 1.

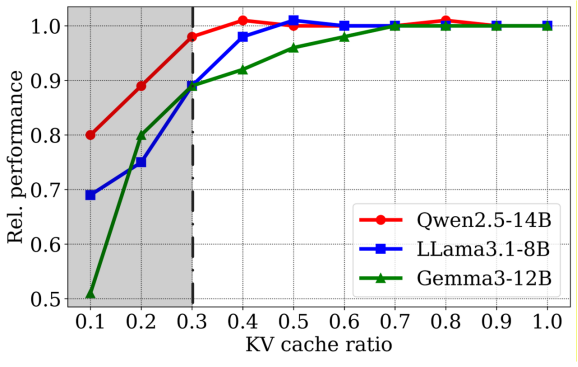

Figure 2: The overall performance of KVzip.

This limitation highlights a critical gap in current eviction algorithms. Notably, theoretical analysis in **H2O** (Zhang et al., 2023) demonstrates that KV caches can theoretically be compressed to as low as 5% of their original size while preserving model performance. This substantial discrepancy

between theoretical potential and practical achievement (e.g., from 5% to 30%) indicates considerable room for improvement in designing more robust and generalizable eviction algorithms.

From a set-theoretic viewpoint, the ideal universal query can be conceptualized as covering the union of all possible query-specific importance sets:

$$\mathcal{K}_{\text{ideal}} = \bigcup_{q \in \mathcal{Q}} \mathcal{K}_q \tag{4}$$

No single proxy query can adequately approximate $\mathcal{K}_{\text{ideal}}$. However, by introducing complementary signals, we can construct a more comprehensive approximation. We propose that the generalization capability can be enhanced by considering both positive signals (semantic relevance) and negative signals (their complement). Let $\mathcal{K}_{\text{pos}}$ represent KV pairs that are salient under semantic reconstruction, and $\mathcal{K}_{\text{neg}}$ represent KV pairs that receive high activation under a maximum-entropy noise signal. Note that $\mathcal{K}_{\text{pos}}$ and $\mathcal{K}_{\text{neg}}$ are not strict complementary sets but soft importance regions induced by two different scoring distributions. The overlap between them refers to tokens that remain highly salient under both distributions, not a literal set-theoretic intersection, capturing tokens that combine semantic informativeness with structural stability and making them more robust across diverse downstream queries.

To address these limitations, we formulate the KV cache eviction problem as a **contrastive selection task**. Instead of relying on a single proxy query, we propose to use dual signals: a positive (semantic) signal q_{pos} that approximates expected query patterns, and a negative (structural) signal q_{neg} that provides structural contrast through maximum entropy noise.

Let $\mathbf{S}_{\text{pos}} = \text{Importance}(\text{KV}_{KB}, q_{\text{pos}})$ and $\mathbf{S}_{\text{neg}} = \text{Importance}(\text{KV}_{KB}, q_{\text{neg}})$ be the importance score vectors computed in parallel. Our goal is to find a fusion function $\mathcal{F}$ that combines these signals to produce a more robust importance estimation:

$$\mathbf{S}_{\text{fused}} = \mathcal{F}\left(\mathbf{S}_{\text{pos}}, \mathbf{S}_{\text{neg}}; \Theta\right) \tag{5}$$

where Θ represents algorithmic parameters including normalization schemes and fusion rules. The final retained cache is then selected as:

$$\text{KV}_{\text{retained}} = \text{Top}_k\left(\mathbf{S}_{\text{fused}}\right) \tag{6}$$

The core optimization challenge is to design $\mathcal{F}$ such that it simultaneously addresses three key issues:

1. **Efficiency**: $\mathcal{F}$ must not significantly increase computational overhead compared to single-query methods, especially given the already expensive context reconstruction process.

2. **Balance**: $\mathcal{F}$ must normalize and balance the different numerical scales of $\mathbf{S}_{\text{pos}}$ and $\mathbf{S}_{\text{neg}}$, addressing the inherent score disparity between semantic and structural signals.

3. **Generalization**: $\mathcal{F}$ must prioritize semantic importance while leveraging structural contrast to identify KV pairs with broader utility across $\mathcal{Q}$, particularly crucial for high compression ratios (e.g., below 30% retention).

Formally, we aim to maximize the expected coverage of important KV pairs across the query distribution while pushing the eviction boundary toward theoretical limits:

$$\begin{aligned} &\max_{\mathcal{F}} \mathbb{E}_{q \sim \mathcal{Q}} \left[\frac{|\mathcal{K}_q \cap \text{KV}_{\text{retained}}|}{|\mathcal{K}_q|} \right] \\ &\text{s.t.} \quad |\text{KV}_{\text{retained}}| \leq \alpha \cdot n, \ \alpha < 0.3 \end{aligned} \tag{7}$$

where $\mathcal{K}_q$ represents the ground-truth important KV pairs for query q, and α denotes the retention ratio. This formulation explicitly targets the challenging regime where existing methods fail, while incorporating the set-theoretic insight that contrastive signals can better approximate the universal query domain.

4 Methodology

To address the challenge of KV cache eviction during the prefill stage, we propose a novel algorithm named **ContrastKV**. The core limitation of existing query-agnostic methods lies in their reliance on a single proxy query to approximate an ideal universal query q_{univ}, which inevitably introduces blind spots for diverse downstream queries. Our key insight is that the robustness of the eviction decision can be enhanced by contrasting semantic and structural signals, inspired by the diversity principle in ensemble learning.

The ContrastKV algorithm operates in three consecutive stages to identify a more generalizable set of KV pairs:

1. **Contrastive Signal Generation**: Efficiently produces a pair of complementary signals, including a positive (semantic) sample and a negative (structural) sample.

2. **Parallel Importance Scoring**: Independently computes normalized importance scores for both signals to balance their influence and reduce computational overhead.

3. **Contrastive Fusion**: Fuses the two signals through a two-level mechanism (Head-level and Layer-level) to refine the final eviction decision, ensuring both global coverage and local precision.

By contrasting semantic consistency with structural robustness, ContrastKV constructs a more reliable decision boundary for multi-query scenarios.

4.1 Contrastive Signal Generation

The goal of this stage is to generate a concise yet discriminative pair of signals: one to capture semantically salient token distributions, and the other to establish a stable structural baseline.

Let the knowledge base have length n. For the **positive signal**, we follow the KVzip approach and adopt the knowledge base reconstruction prompt as q_{pos}, setting its length to $t_{\text{pos}} = n$. For the **negative signal**, we require an input that is semantically orthogonal to the knowledge base in order to maximize contrastive ambiguity. Rather than using a lengthy, unrelated document, we generate a **Maximum Entropy Noise Sequence** of length t_{neg} ($t_{\text{neg}} \ll n$), where each token is independently sampled from a uniform distribution over the vocabulary. This compact noise sequence serves as q_{neg} and provides a pure structural baseline. It is designed to highlight "attention sink" tokens—tokens that attract high attention scores irrespective of semantic content. Appendix A.1 provides a detailed analysis of the negative signal selection.

Both q_{pos} and q_{neg} are forwarded through the LLM together with the full KV cache to obtain their respective attention patterns. These patterns then serve as the foundation for the subsequent scoring distributions.

4.2 Parallel Importance Scoring

To avoid the latency increase from processing two extended sequences, we evaluate the positive and negative signals in parallel. For each layer l and KV head h, given the query features $\mathbf{Q}_{l,h}$ and key features $\mathbf{K}_{l,h}$, we compute the attention matrix $\mathbf{A}_{l,h} = \text{Softmax}(\mathbf{Q}_{l,h}\mathbf{K}_{l,h}^T)$ and slice it to obtain $\bar{\mathbf{A}}_{l,h}$ corresponding to the cached keys.

The importance score vector for each KV head is derived by taking the maximum over the group and token dimensions:

$$S_{l,h}^{(\text{pos/neg})} = \max_{g=1,\ldots,G;\, i=1,\ldots,t_{\text{pos/neg}}} \bar{\mathbf{A}}_{l,h}[g,i] \in \mathbb{R}^n \tag{8}$$

We designate the aggregated scores S from all KV heads as the maximum cross-attention score. The visualized results of these scores are shown in the Appendix A.2.

4.3 Contrastive Fusion Mechanism

The normalized scores are fused through a two-stage process, first at the head-level to aggregate attention patterns across heads, and then at the layer-level to consolidate the signal across layers. This design preserves globally important tokens while refining local decisions.

Head-level Fusion. As different attention heads exhibit specialized focus, some may be critical for certain query types even if their overall attention mass is low. To prevent evicting all KV pairs from such heads during dynamic budget allocation, we perform head-level gating. We aim to dynamically reserve a portion of relatively important KV pairs for each head through head-level fusion based on positive and negative sample scores, see Figure 3.

Let $\mathcal{G}_\beta(\cdot)$ denote the β-th quantile of a score vector. We set the lower and upper thresholds at β and $1-\beta$. For each position i, the gated score $\tilde{S}_i$ is defined as:

$$\tilde{S}_i = \begin{cases} 1.0, & \text{if } S_i^{pos} \geq \mathcal{G}_{1-\beta}(S^{pos}) \\ & \text{and } S_i^{neg} \geq \mathcal{G}_{1-\beta}(S^{neg}), \\ 0.0, & \text{if } S_i^{pos} \leq \mathcal{G}_{\beta}(S^{pos}) \\ & \text{and } S_i^{neg} \leq \mathcal{G}_{\beta}(S^{neg}), \\ S_i^{pos}, & \text{otherwise.} \end{cases} \tag{9}$$

This rule promotes tokens that are salient in both signals (likely generalizable) and demotes tokens that are negligible in both (likely redundant), while retaining the original positive score for ambiguous cases. It ensures each head retains a minimal set of high-potential KV pairs.

Layer-level Fusion. To further enhance sensitivity to general query patterns, we perform a score boost at each layer using the negative signal as a reference, as illustrated in Figure 4. The boost is

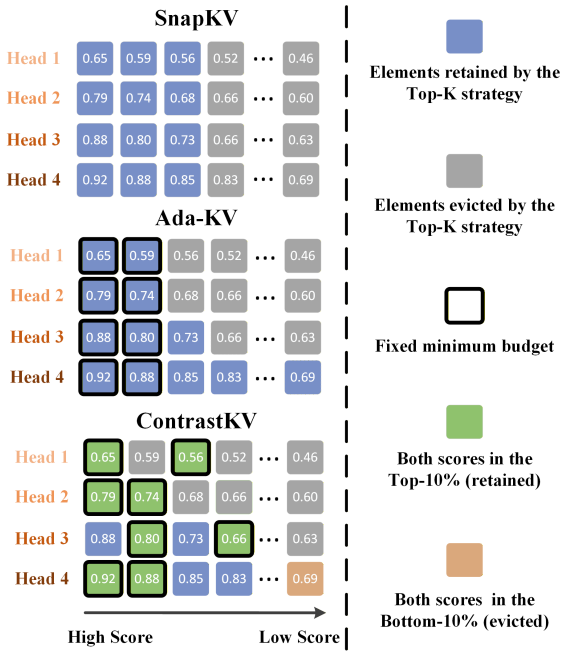

Figure 3: Head-level fusion strategy overview. The scores here are only for the purpose of illustrating the algorithm's principle and do not represent the actual attention scores. The color of the Head represents its importance, from light to dark.

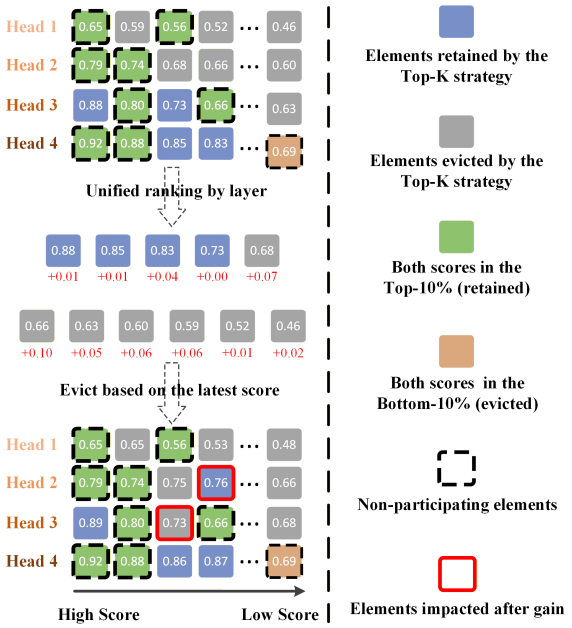

Figure 4: Layer-level fusion strategy overview.

applied only to positions not already decided by Eq. 9 (i.e., those with score S_i^{pos}). The boost magnitude is proportional to the min-max normalized score $\hat{S}_i^{neg}$ of the negative score:

$$\hat{S}_i^{neg} = \frac{S_i^{neg} - \min(S_i^{neg})}{\max(S_i^{neg}) - \min(S_i^{neg})}$$
$$\hat{S}_i = \min\left(S_i^{pos} + \gamma \cdot \hat{S}_i^{neg},\ \max(S_i^{pos})\right) \quad (10)$$

where γ is a hyperparameter controlling the gain intensity. This step gently elevates tokens that, while not top in positive signal, show relative prominence in the negative signal, refining the decision boundary without overriding the primary semantic signal.

The fusion function of the head-level fusion and the layer-level fusion is fixed rather than learned because we intentionally adopt a simple linear formulation to preserve interpretability and robustness across different models and tasks.

In conclusion, ContrastKV introduces a contrastive eviction framework that mitigates the blind spots of single-query approximation by jointly leveraging semantic (positive) and structural (negative) signals. Through parallel scoring with normalization and a two-level fusion strategy, it achieves a more robust and generalizable KV cache selection. Experimental results (Sec. 5) demonstrate that ContrastKV consistently outperforms prior methods across diverse query benchmarks. Additionally, we provide experimental results to explain the recommended values for hyperparameters such as t_{neg}, β, and γ in Appendix A.3.

5 Experiment

All experiments in this section were conducted in an environment with PyTorch 2.1, Python 3.10, and CUDA 12.1. Unless otherwise specified, experiments using Qwen2.5-7B-Instruct-1M and Llama3.1-8B-Instruct are conducted on an RTX 4080 SUPER with 32 GB of memory and an RTX 4090 with 48 GB of memory, respectively.

5.1 Experimental Setup

Baselines. We compare ContrastKV with state-of-the-art KV cache eviction algorithms from two categories: query-aware methods (SnapKV (Li et al., 2024a) and Ada-KV (Feng et al., 2025a)) and query-agnostic approaches (KVzip (Kim et al., 2025)). For SnapKV and its improved variant Ada-SnapKV, we apply max-pooling with a kernel size of 7 and an observation window of 32, following the original configurations. While SnapKV uses uniform attention-head budget allocation, Ada-KV adopts an adaptive allocation strategy. For KVzip, we adhere to the official GitHub implementation with a fixed chunk size of $m = 2$K across all contexts, models, and tasks; it also employs adaptive allocation similar to Ada-KV.

Table 1: Performance comparison of different KV eviction methods across various models and retrieval tasks. En.Sum uses **ROUGE**, Code.RepoQA uses **Pass@1**, and the remaining datasets use **Accuracy** as the evaluation metric.

Model	Method	Multi-tasking		Global Information		Semantic Retrieval		String Retrieval	
		Mix.Sum +NIAH	Mix.RepoQA +KV	ICL. ManyShot	En.Sum	En.MultiChoice	Code. RepoQA	Retr.KV	Retr. Prefix-Suffix
Qwen2.5-7B-Instruct-1M	Full KV	68.27	80.11	37.41	36.62	79.17	59.32	67.80	51.40
	Budget = 20%								
	SnapKV	32.10	5.68	**35.93**	31.05	55.09	6.59	6.00	5.80
	Ada-KV	26.68	3.69	35.19	29.20	53.70	4.09	3.80	1.40
	KVzip	65.38	73.72	32.96	35.43	72.22	50.91	33.20	2.40
	ContrastKV	**67.86**	**77.70**	35.56	**36.21**	**77.78**	**55.68**	**62.00**	**27.2**
	Budget = 10%								
	SnapKV	24.76	2.27	**35.19**	30.12	54.63	2.73	2.80	1.40
	Ada-KV	18.69	1.56	**35.19**	28.52	47.67	1.36	0.60	0.20
	KVzip	52.13	53.69	32.59	33.66	68.06	27.05	7.80	0.00
	ContrastKV	**63.79**	**66.05**	33.70	**33.73**	**69.44**	**42.73**	**39.60**	**7.00**
Llama3.1-8B-Instruct	Full KV	65.17	23.72	34.07	39.15	75.00	42.73	61.40	27.40
	Budget = 20%								
	SnapKV	31.71	3.41	**39.63**	30.44	59.72	3.41	7.20	1.20
	Ada-KV	31.35	3.98	**39.63**	30.69	**66.20**	4.77	7.20	0.60
	KVzip	59.85	12.36	35.93	37.76	59.72	21.14	0.20	0.40
	ContrastKV	**66.62**	**20.03**	36.67	**39.27**	60.19	**35.91**	**47.20**	**31.40**
	Budget = 10%								
	SnapKV	23.61	1.56	40.74	28.94	47.69	0.68	2.20	0.60
	Ada-KV	22.43	1.56	**41.85**	28.90	50.46	1.14	3.40	0.60
	KVzip	47.01	9.66	32.22	34.38	63.89	14.09	0.00	0.00
	ContrastKV	**60.92**	**14.63**	34.44	**37.62**	**64.81**	**24.77**	**15.80**	**17.20**
Qwen2.5-14B-Instruct-1M	Full KV	69.71	84.80	45.56	38.63	81.94	68.18	94.80	81.40
	Budget = 20%								
	KVzip	67.55	82.67	46.30	36.86	79.17	65.00	33.40	12.80
	ContrastKV	**69.58**	**83.95**	**46.67**	**38.51**	**81.94**	**67.05**	**85.20**	**60.80**
	Budget = 10%								
	KVzip	63.84	77.13	45.93	31.38	71.76	55.45	3.00	3.40
	ContrastKV	**68.39**	**81.25**	**45.93**	**37.12**	**81.94**	**63.64**	**62.40**	**38.40**

Datasets. All methods are evaluated on the SCBench benchmark (Li et al., 2025), which provides a comprehensive multi-query evaluation suite comprising 11 datasets spanning four long-text capabilities: string retrieval, semantic retrieval, global information, and multi-tasking. In the parameter study (Section A.3), we select one representative dataset from each of the first three categories to establish broad yet balanced hyperparameters. For the main experiments (Section 5.2) and ablation study (Section 5.3), we use the remaining eight datasets, with two per capability category.

Models. We evaluate three instruction-tuned LLMs: Qwen2.5-7B-Instruct-1M, Llama3.1-8B-Instruct, and Qwen-14B-Instruct-1M, covering mainstream mid-scale and large model sizes to verify the model-agnostic property. All experiments are conducted in bfloat16 precision. Due to the 128K-token context-length limit of Llama3.1-8B-Instruct, we tokenize inputs with the Llama-3.1 tokenizer and exclude samples from En.QA and En.MultiChoice whose contexts exceed 125K tokens.

5.2 Performance Comparison

As shown in Table 1, we evaluate multi-query performance of Qwen2.5-7B-Instruct-1M and Llama3.1-8B-Instruct across eight benchmarks spanning four categories. Query-aware eviction methods (e.g., SnapKV, AdaKV) suffer severe performance degradation in multi-query settings, demonstrating limited generalization. The baseline KVzip also exhibits significant drops, retaining only 70% accuracy at 20% cache budget and 56% at 10% budget. In contrast, our proposed ContrastKV maintains 92% of the original accuracy

under a 20% cache budget and 74% even at an extreme 10% budget, indicating robust compression with minimal accuracy loss. Notably, all methods show higher tolerance in global-information tasks, where accuracy occasionally improves. This effect likely stems from reduced attention dispersion after KV eviction (Ye et al., 2025). These results highlight that ContrastKV not only consistently outperforms existing baselines but also sustains strong generalization in tasks beyond query-specific retrieval, validating the effectiveness of our contrastive fusion design for practical multi-query inference.

To further validate the model-agnostic behaviour of ContrastKV, we conduct additional experiments on Qwen2.5-14B-Instruct-1M. The results are consistent with those observed on the 7B and 8B models: ContrastKV substantially outperforms KVzip under both 20% and 10% budget settings, particularly on string retrieval tasks, where the performance gap is most pronounced.

We also evaluated the performance of KVzip and ContrastKV on the Qwen2.5-7B-Instruct-1M dataset at different retention rates (Table 2). When the budget is sufficient ($\geq$50%), ContrastKV slightly lags behind KVzip. However, when the budget drops to 40% or lower, ContrastKV begins to show a significant advantage. This is precisely the regime where KV eviction becomes practically meaningful, as high-budget settings yield limited acceleration gains.

Table 2: Normalized performance of KVzip and ContrastKV under different KV cache budget ratios on Qwen2.5-7B-Instruct-1M.

Ratio	KVzip	ContrastKV
1.00	100.00%	100.00%
0.90	99.44%	98.81%
0.80	99.36%	98.92%
0.70	99.14%	98.48%
0.60	98.46%	97.85%
0.50	97.08%	96.68%
0.40	94.50%	95.01%
0.30	92.51%	93.62%
0.20	76.28%	91.68%
0.10	57.28%	74.19%

5.3 Ablation Study

We conduct an ablation study on ContrastKV's fusion components using Qwen2.5-7B-Instruct-1M under a 20% KV cache budget. Results in Table 3 show that head-level fusion provides the primary performance gain, most notably improving string retrieval by 42.22%. This improvement stems from its ability to preserve salient common features across positive and negative samples, enhancing robustness to diverse queries. Layer-level fusion contributes more modest gains but further refines performance when combined with head-level fusion (45.81% improvement over the baseline). These findings confirm that while head-level fusion drives the core capability of query-agnostic generalization, layer-level fusion effectively refines the eviction policy. Their combination in ContrastKV ensures a balanced and highly effective compression mechanism, validating the design of our multi-level contrastive fusion approach.

5.4 Efficiency

To evaluate the efficiency of ContrastKV, we conduct experiments and report the results in Table 4. We measure decoding speed and memory consumption with and without our method under constrained KV cache budgets. For a realistic and reproducible setup, we adopt the publicly released code from Ku et al. (2025) as the knowledge base, which postdates the release of Qwen2.5-7B-Instruct-1M. Based on this knowledge base, we construct ten relevant questions to assess decoding latency and GPU memory usage. The full set of questions and response details are provided in Appendix A.4. As shown in Table 4, ContrastKV achieves significantly lower decoding latency compared to the full-cache baseline. This improvement is particularly evident in long-context question-answering tasks. We further examine performance under two KV cache budget levels (e.g., 20% and 10%). The reported results are averaged over ten knowledge-base queries. Our analysis shows that GPU memory usage scales nearly linearly with the cache budget, while decoding latency drops substantially. Using only 20% of the KV cache reduces latency by approximately 50% relative to the full-cache setting. These findings confirm that ContrastKV delivers considerable inference acceleration alongside notable memory savings, underscoring its practical efficiency for deployment in resource-aware scenarios.

5.5 Application Potential in Edge Networks

The high eviction ratio achieved by our algorithm, together with the reusability of evicted KV caches, enables its integration with cloud-edge collaboration frameworks (Eshratifar et al., 2019; Zhou et al., 2021; Yao et al., 2025). Such integration can fully

Table 3: The contribution of using Head-level fusion and Layer-level fusion individually. We assume the model achieves 100% accuracy with the full KV cache and normalize the performance of each task relative to the full-cache performance.

Method	Multi-tasking	Global Information	Semantic Retrieval	String Retrieval
KB.Rec (Kim et al., 2025)	93.75%	92.38%	88.91%	30.42%
KB.Rec w. Head-level fusion	97.88%(+4.13%)	96.07%(+3.69%)	96.07%(+7.16%)	72.64%(+42.22%)
KB.Rec w. Layer-level fusion	97.66%(+3.91%)	95.25%(+2.87%)	94.53%(+5.62%)	48.03%(+17.61%)
ContrastKV	98.10%(+4.35%)	96.80%(+4.42%)	96.37%(+7.46%)	76.23%(+45.81%)

Table 4: Model Speed and KV Cache GPU Memory Usage

Method	Budget	Decoding Latency	GPU Memory
Full KV	100%	110.81 ms/token	7.8 GB
ContrastKV	20%	55.28 ms/token	1.5 GB
ContrastKV	10%	44.26 ms/token	0.7 GB

leverage the efficiency gains of ContrastKV, enhance user-side responsiveness, and reduce the frequent query load on cloud servers. In Appendix A.5, we outline a preliminary cloud-edge acceleration framework built upon our eviction algorithm and report initial feasibility experiments that support its practical deployment.

6 Conclusion

This paper tackles the KV cache explosion problem in long-context, multi-query KBQA through ContrastKV, a query-agnostic eviction framework based on contrastive signal fusion. Unlike prior methods that depend on a single proxy query, the proposed approach jointly assesses semantic consistency and structural robustness, enabling consistent performance across varied downstream tasks. Extensive experiments show that ContrastKV maintains 92% of the original accuracy with only 20% of the KV cache, while significantly lowering inference latency and memory usage. The combination of head-level and layer-level fusion further improves generalization capability, and the efficiency benefits together with cloud-edge compatibility underscore its practical relevance. Future work will explore extensions to multimodal and dynamic retrieval scenarios, as well as optimized edge-side deployment.

Limitations

Although ContrastKV has clear advantages over existing methods, it still has two main limitations: (1) If the model experiences memory overflow during the prefill phase, ContrastKV cannot alleviate this issue. This problem also exists in most KV cache eviction methods. (2) Because the positive sample has the same length as the knowledge base, the attention computation results in a high time complexity. As a result, the eviction algorithm phase consumes a significant amount of time. However, since the evicted KV cache is reusable across multiple queries, the eviction cost is amortized in practice. In deployment, the system can serve initial queries using the full KV cache and switch to the evicted cache once eviction is complete, ensuring that online interaction latency is not affected.

Additionally, due to device limitations and research gaps, we have only proposed a preliminary concept of the cloud-edge collaborative acceleration framework based on KV cache eviction algorithms and conducted simple feasibility experiments. We plan to further explore the feasibility of this concept in the future.

Acknowledgments

This work was supported by Pengcheng Laboratory under grant No. PCL2025A09 and No. 2025QYB020.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, and 1 others. 2024. Longbench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 3119–3137.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and 1 others. 2024. Pyramidkv: Dynamic kv cache compression based on

pyramidal information funneling. *arXiv preprint arXiv:2406.02069*.

Zhaojing Cao, Can Wan, Zijun Zhang, Furong Li, and Yonghua Song. 2019. Hybrid ensemble deep learning for deterministic and probabilistic low-voltage load forecasting. *IEEE Transactions on Power Systems*, 35(3):1881–1897.

Vivek Chari and Benjamin Van Durme. 2025. Compactor: Calibrated query-agnostic kv cache compression with approximate leverage scores. *arXiv preprint arXiv:2507.08143*.

Tristan Coignion, Clément Quinton, and Romain Rouvoy. 2024. A performance study of llm-generated code on leetcode. In *Proceedings of the 28th international conference on evaluation and assessment in software engineering*, pages 79–89.

Giulio Corallo, Orion Weller, Fabio Petroni, and Paolo Papotti. 2025. Beyond rag: Task-aware kv cache compression for comprehensive knowledge reasoning. *arXiv preprint arXiv:2503.04973*.

Tri Dao, Daniel Y Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *The Thirty-Sixth Annual Conference on Neural Information Processing Systems*, pages 16344–16359.

Pedro Domingos. 2000. A unified bias-variance decomposition. In *Proceedings of 17th international conference on machine learning*, pages 231–238. Morgan Kaufmann Stanford.

Amir Erfan Eshratifar, Mohammad Saeed Abrishami, and Massoud Pedram. 2019. Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services. *IEEE transactions on mobile computing*, 20(2):565–576.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. 2025a. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*.

Zhaopeng Feng, Yan Zhang, Hao Li, Bei Wu, Jiayu Liao, Wenqiang Liu, Jun Lang, Yang Feng, Jian Wu, and Zuozhu Liu. 2025b. Tear: Improving llm-based machine translation with systematic self-refinement. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3922–3938.

Xianlei Fu, Robert Lee Kong Tiong, and Limao Zhang. 2025a. Emnet: An ensemble deep learning approach for geological condition detection in tunnel excavation. *Expert Systems with Applications*, 261:125484.

Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. 2025b. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. In *The Thirteenth International Conference on Learning Representations*, pages 35441–35462.

Mudasir A Ganaie, Minghui Hu, Ashwani Kumar Malik, Muhammad Tanveer, and Ponnuthurai N Suganthan. 2022. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence*, 115:105151.

Bofei Gao, Zefan Cai, Runxin Xu, Peiyi Wang, Ce Zheng, Runji Lin, Keming Lu, Dayiheng Liu, Chang Zhou, Wen Xiao, and 1 others. 2025. Llm critics help catch bugs in mathematics: Towards a better mathematical verifier with natural language feedback. In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 14588–14604.

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*.

Jang-Hyun Kim, Jinuk Kim, Sangwoo Kwon, Jae W Lee, Sangdoo Yun, and Hyun Oh Song. 2025. Kvzip: Query-agnostic kv cache compression with context reconstruction. In *The Thirty-Ninth Annual Conference on Neural Information Processing Systems*.

Zixiao Kong, Xianquan Wang, Shuanghong Shen, Keyu Zhu, Huibo Xu, and Yu Su. 2025. Scholargec: Enhancing controllability of large language model for chinese academic grammatical error correction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 24339–24347.

A KROGH. 1995. Neural net-work ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, 7:231–238.

Max Ku, Cheuk Hei Chong, Jonathan Leung, Krish Shah, Alvin Yu, and Wenhu Chen. 2025. Theoremexplainagent: Towards video-based multimodal explanations for llm theorem understanding. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6663–6684.

Yunshi Lan, Gaole He, Jinhao Jiang, Jing Jiang, Wayne Xin Zhao, and Ji-Rong Wen. 2022. Complex knowledge base question answering: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11196–11215.

Yucheng Li, Huiqiang Jiang, Qianhui Wu, Xufang Luo, Surin Ahn, Chengruidong Zhang, Amir H Abdi, Dongsheng Li, Jianfeng Gao, Yuqing Yang, and 1 others. 2025. Scbench: A kv cache-centric analysis of long-context methods. In *The Thirteenth International Conference on Learning Representations*, pages 39209–39239.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. 2024a. Snapkv: Llm knows what you are looking for before generation. In *The Thirty-Eighth Annual Conference on Neural Information Processing Systems*, pages 22947–22970.

Zhenyu Li, Sunqi Fan, Yu Gu, Xiuxing Li, Zhichao Duan, Bowen Dong, Ning Liu, and Jianyong Wang. 2024b. Flexkbqa: A flexible llm-powered framework for few-shot knowledge base question answering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pages 18608–18616.

Qianli Liu, Zicong Hong, Peng Li, Fahao Chen, and Song Guo. 2025. Mell: Memory-efficient large language model serving via multi-gpu kv cache management. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pages 1–10. IEEE.

Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. 2023. Scissorhands: exploiting the persistence of importance hypothesis for llm kv cache compression at test time. In *The Thirty-Seventh Annual Conference on Neural Information Processing Systems*, pages 52342–52364.

Amrith Setlur, Saurabh Garg, Xinyang Geng, Naman Garg, Virginia Smith, and Aviral Kumar. 2024. Rl on incorrect synthetic data scales the efficiency of llm math reasoning by eight-fold. *Advances in Neural Information Processing Systems*, 37:43000–43031.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *The Thirty-First Annual Conference on Neural Information Processing Systems*.

Saranya Venkatraman, Nafis Irtiza Tripto, and Dongwon Lee. 2025. Collabstory: Multi-llm collaborative story generation and authorship analysis. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 3665–3679.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. 2024. Efficient streaming language models with attention sinks. In *The Twelfth International Conference on Learning Representations*, pages 16082–16102.

Fangyuan Xu, Tanya Goyal, and Eunsol Choi. 2025. Refreshkv: Updating small kv cache during long-form generation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 24878–24893.

An Yang, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoyan Huang, Jiandong Jiang, Jianhong Tu, Jianwei Zhang, Jingren Zhou, and 1 others. 2025a. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*.

Dongjie Yang, Xiaodong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. 2024. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 3258–3270.

Yanlai Yang, Zhuokai Zhao, Satya Narayan Shukla, Aashu Singh, Shlok Kumar Mishra, Lizhu Zhang, and Mengye Ren. 2025b. Streammem: Query-agnostic kv cache memory for streaming video understanding. *arXiv preprint arXiv:2508.15717*.

Zhi Yao, Zhiqing Tang, Wenmian Yang, and Weijia Jia. 2025. Enhancing llm qos through cloud-edge collaboration: A diffusion-based multi-agent reinforcement learning approach. *IEEE Transactions on Services Computing*, 18(3):1412–1427.

Tianzhu Ye, Li Dong, Yuqing Xia, Yutao Sun, Yi Zhu, Gao Huang, and Furu Wei. 2025. Differential transformer. In *The Thirteenth International Conference on Learning Representations*, pages 65910–65930.

Minwei Zhang, Haifeng Sun, Jingyu Wang, Shaolong Li, Wanyi Ning, Qi Qi, Zirui Zhuang, and Jianxin Liao. 2025a. Clusterattn: Kv cache compression under intrinsic attention clustering. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14451–14473.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, and 1 others. 2023. H2o: heavy-hitter oracle for efficient generative inference of large language models. In *The Thirty-Seventh Annual Conference on Neural Information Processing Systems*, pages 34661–34710.

Ziyao Zhang, Chong Wang, Yanlin Wang, Ensheng Shi, Yuchi Ma, Wanjun Zhong, Jiachi Chen, Mingzhi Mao, and Zibin Zheng. 2025b. Llm hallucinations in practical code generation: Phenomena, mechanism, and mitigation. *Proceedings of the ACM on Software Engineering*, 2(ISSTA):481–503.

Hongbo Zhou, Weiwei Zhang, Chengwei Wang, Xin Ma, and Haoran Yu. 2021. Bbnet: A novel convolutional neural network structure in edge-cloud collaborative inference. *Sensors*, 21(13):4494.

A Appendix

A.1 Ablation Experiment on the Contrast Sample

To validate the rationale for selecting random strings, which are completely unrelated to the knowledge base, as the contrast sample, we conducted an ablation experiment. For the contrast sample corresponding to the random strings, we selected the first 10% of the content from the knowledge base, while keeping other operations

unchanged. As shown in Table 5, with a 20% budget, the accuracy of the two different samples is similar. However, at a higher eviction ratio, specifically with a 10% budget, using random strings as the contrast sample demonstrates a clear advantage. This result not only verifies the rationale for introducing contrast samples but also proves the necessity of selecting data unrelated to the knowledge base for comparison.

A.2 Attention Score Visualization

We refer to the aggregated scores S of all KV heads as the maximum cross-attention score. Figure 5 illustrates the visualization of these scores. It can be seen that our algorithm identifies and retains many KV pairs with lower attention scores in single knowledge base reconstruction, especially in the shallower layers.

A.3 Algorithm Parameter Settings

In Section 4, we provide a detailed description of the ContrastKV algorithm's procedure. This algorithm includes three critical parameters: β, t_{neg}, and γ. Currently, these parameters are manually set. To explore the optimal parameter values, we conduct comparative experiments using the Qwen2.5-7B-Instruct-1M model in this section, with a fixed KV cache budget of 20%.

As shown in Table 6, when the β ranges from 5% to 10%, the accuracy difference remains within 0.5%. However, performance significantly declines when the β exceeds 15%. Therefore, we recommend setting the β to any value between 5% and 10%.

Table 7 shows that under various settings for t_{neg}, the accuracy differences stay within 1%. We infer that the specific length of negative samples has minimal impact on our algorithm's accuracy, which is more closely related to the original context's irrelevance. Considering time cost, we suggest limiting the negative sample length to within 128 tokens.

Similarly, as indicated in Table 8, the effects of different boost scales are analogous to those of negative sample lengths, with accuracy differences remaining within 1% across different settings. We recommend setting the boost scale γ to the value corresponding to the highest achieved accuracy, which is 0.12.

A.4 Question Settings and Response Details for Efficiency Analysis

Since the code of (Ku et al., 2025) is released in March 2025, which is later than the release date of Qwen2.5-7B-Instruct-1M in January 2025, the model cannot have seen this data during pretraining or instruction tuning. This setting helps verify whether the model's KV cache eviction algorithm can still support accurate retrieval and reasoning when facing entirely unfamiliar long-text inputs, rather than relying on memorized knowledge encoded in its parameters.

We design 10 related questions based on this knowledge base to evaluate decoding latency and GPU memory usage. These questions are distributed across three core code files. The questions and their corresponding answers are generated by the Gemini 3 Pro model and manually verified for accuracy. Detailed information is shown in Table 9. Although some details differ, the KV cache with both 20% and 10% budgets is able to answer all questions correctly.

A.5 Cloud-Edge Collaborative Acceleration Framework Based on KV Eviction Algorithm

We propose to deploy the designed KV cache eviction algorithm through cloud-edge collaboration. The framework consists of three layers: the user side, edge servers, and cloud servers. The workflow is as follows: the user uploads the knowledge base to the cloud server, where a prefill operation is performed to generate a large KV cache. Next, the cloud server executes the KV cache eviction algorithm while processing user requests. Finally, the evicted KV cache is transmitted to the edge server, where it is loaded and used for subsequent inference tasks. This cloud-edge collaborative processing approach avoids the computational burden and resource limitations of handling large KV caches directly on the edge server. Additionally, by offloading some tasks to the edge server, it alleviates the access pressure on the cloud server.

We conducted a preliminary feasibility experiment using a workstation equipped with an H800 80GB to simulate a cloud server, and a desktop with an RTX 4090 24GB to simulate an edge server. The tested model was Qwen2.5-7B-Instruct-1M. We selected the code of exllamav3 as the knowledge base, with a length of 480k tokens. The results are shown in Table 10. Using the evicted KV cache, the in-

Table 5: The impact of using contrast samples related and unrelated to the knowledge base.

Budget	Method	Multi-tasking	Global Information	Semantic Retrieval	String Retrieval
20%	ContrastKV w. Random String	98.10%	96.80%	96.37%	76.23%
	ContrastKV w. First 10% of KB	97.18%(-0.92%)	95.31%(-1.49%)	97.68%(+1.31%)	77.85%(+1.62%)
10%	ContrastKV w. Random String	87.53%	91.08%	81.00%	39.09%
	ContrastKV w. First 10% of KB	75.58%(-11.95%)	88.06%(-3.02%)	77.07%(-3.93%)	22.32%(-16.77%)

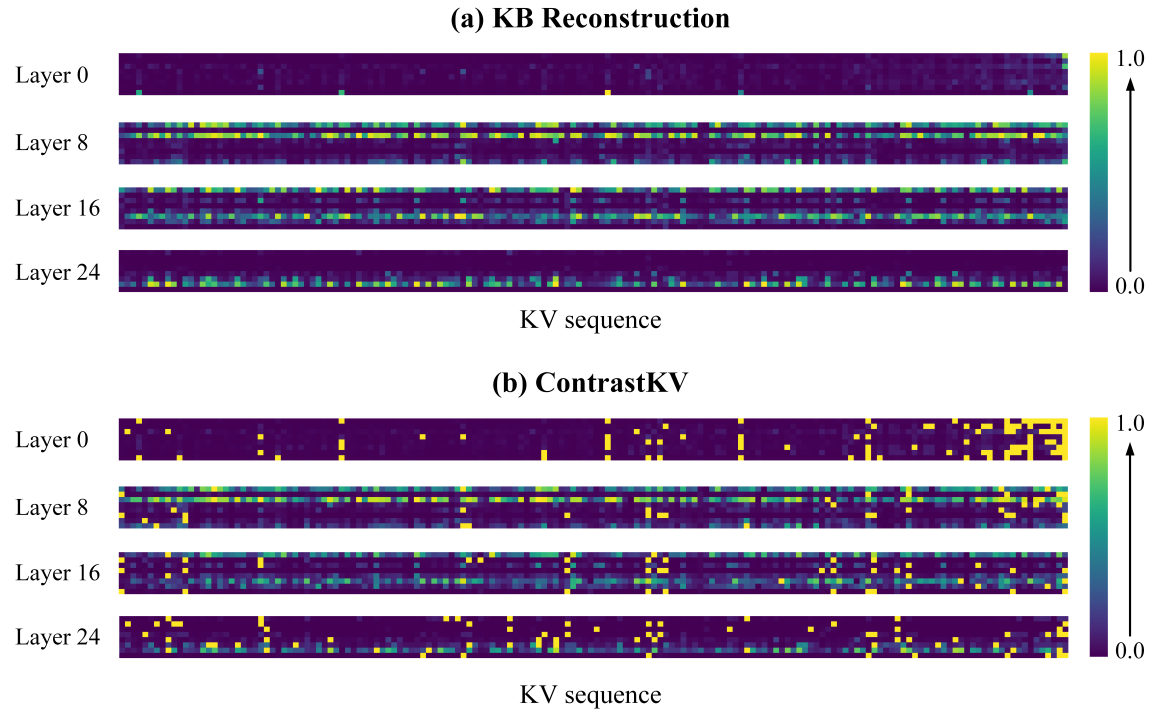

Figure 5: Visualization of maximum attention scores. Each heatmap visualizes the maximum attention scores received by KV pairs in KV_c for a knowledge base example, computed using Llama3.1-8B-Instruct.

Table 6: Performance Under Different Values of β

β	Math.Find	En.QA	Retr.MultiHop	Average
5%	**34.33**	42.16	39.29	38.59
10%	34.17	**42.31**	**39.69**	**38.72**
15%	33.67	40.45	39.29	37.80
20%	33.83	36.95	38.78	36.52

Table 7: Performance Under Different Values of t_{neg}

t_{neg}	Math.Find	En.QA	Retr.MultiHop	Average
16	33.67	41.05	**40.62**	38.45
32	33.83	40.46	40.58	38.29
64	**34.17**	**42.31**	39.69	**38.72**
128	33.33	40.22	39.91	37.82

ference speed was significantly higher than when using the full KV cache, even though the performance of the edge server was lower than that of the cloud server. Additionally, because our algorithm supports higher eviction ratios, it allows for broader

Table 8: Performance Under Different Values of γ

γ	Math.Find	En.QA	Retr.MultiHop	Average
0.06	34.00	41.23	**40.00**	38.41
0.12	**34.17**	42.31	39.69	**38.72**
0.24	33.83	40.57	39.56	37.99
0.48	33.17	**42.81**	39.69	38.56

memory limits while maintaining accuracy.

Table 9: Question Settings and Response Details for Efficiency Analysis

Questions	Ground Truth	Responses of Full KV Cache	Responses of 20% KV Cache	Responses of 10% KV Cache	Answer Path
What is the default AI model used if the –model argument is not provided?	gemini/ gemini-1.5-pro-002	The default AI model used if the –model argument is not provided is "gemini/gemini-1.5-pro-002".	default AI model used if the –model argument is not provided: gemini/gemini-1.5-pro-002	The default AI model used if the –model argument is not provided is gemini/gemini-1.5-pro-002.	TheoremExplainAgent -main/generate_video.py
What is the default value for the maximum number of retries for code generation?	5	5	5	5	TheoremExplainAgent -main/generate_video.py
Which two embedding models are allowed according to the argument parser?	azure/text-embedding-3-large and vertex_ai/text-embedding-005	azure/text-embedding-3-large, vertex_ai/text-embedding-005	azure/text-embedding-3-large, vertex_ai/text-embedding-005	azure/text-embedding-3-large, vertex_ai/text-embedding-005	TheoremExplainAgent -main/generate_video.py
What is the default AI model used for text evaluation?	azure/gpt-4o	azure/gpt-4o	azure/gpt-4o	azure/gpt-4o	TheoremExplainAgent -main/evaluate.py
Into how many chunks is a video file divided during evaluation?	10	10	10	10	TheoremExplainAgent -main/evaluate.py
Which mathematical method is used to calculate the overall scores?	Geometric mean	Geometric mean	Geometric Mean	The geometric mean is used to calculate the overall scores.	TheoremExplainAgent -main/evaluate.py
What are the supported file extensions for video evaluation?	.mp4 and .mkv	.mp4, .mkv	.mp4, .mkv	.mp4, .mkv	TheoremExplainAgent -main/evaluate.py
What is the version of the numpy library specified?	2.2.2	numpy∼=2.2.2	numpy∼=2.2.2	numpy∼=2.2.2	TheoremExplainAgent -main/requirements.txt
Which package variant should be installed for kokoro-onnx if a GPU is available?	kokoro-onnx[gpu]	kokoro-onnx[gpu]	kokoro-onnx[gpu]	kokoro-onnx[gpu]	TheoremExplainAgent -main/requirements.txt
What is the specific version of the openai library listed?	1.61.0	openai∼=1.61.0	openai∼=1.61.0	openai∼=1.61.0	TheoremExplainAgent -main/requirements.txt

Table 10: Preliminary verification experiment of cloud-edge collaborative framework. **GPU Memory(KV)** refers to the amount of GPU memory occupied by the KV cache, **GPU Memory(ALL)** refers to the actual peak GPU memory usage during execution, and **Average Accuracy** refers to the average normalized accuracy across the four tasks in Table 3.

Server	Method	Budget	Decoding Latency	GPU Memory(KV)	GPU Memory(ALL)	Average Accuracy
H800	Full KV	100%	91.60 ms/token	27.5 GB	49.6 GB	100%
RTX 4090	KVzip	30%	N/A	7.6 GB	25.8 GB (OOM)	92%
RTX 4090	ContrastKV	20%	40.91 ms/token	5.2 GB	21.6 GB	92%