KVzip: Query-Agnostic KV Cache Compression with Context Reconstruction

Jang-Hyun Kim[1 2], **Jinuk Kim**[1 2], **Sangwoo Kwon**[1], **Jae W. Lee**[1], **Sangdoo Yun**[3], **Hyun Oh Song**[* 1 2]
[1]Seoul National University, [2]Neural Processing Research Center, [3]NAVER AI Lab
{janghyun, hyunoh}@snu.ac.kr
https://github.com/snu-mllab/KVzip

Abstract

Transformer-based large language models (LLMs) cache context as key-value (KV) pairs during inference. As context length grows, KV cache sizes expand, leading to substantial memory overhead and increased attention latency. This paper introduces *KVzip*, a query-agnostic KV cache eviction method enabling effective reuse of compressed KV caches across diverse queries. KVzip quantifies the importance of a KV pair using the underlying LLM to reconstruct original contexts from cached KV pairs, subsequently evicting pairs with lower importance. Extensive empirical evaluations demonstrate that KVzip reduces KV cache size by 3-$4\times$ and FlashAttention decoding latency by approximately $2\times$, with negligible performance loss in question-answering, retrieval, reasoning, and code comprehension tasks. Evaluations include various models such as LLaMA3.1, Qwen2.5, and Gemma3, with context lengths reaching up to 170K tokens. KVzip significantly outperforms existing query-aware KV eviction methods, which suffer from performance degradation even at a 90% cache budget ratio under multi-query scenarios.

1 Introduction

Transformer-based LLMs with long-context capabilities have significantly enhanced real-world applications, including long-document analysis and personalized conversational agents [1, 21, 49]. However, increasing context lengths substantially raises both memory consumption for KV caching and computational costs associated with attention mechanisms [31]. For example, caching 120K tokens in Qwen2.5-14B with FP16 precision requires approximately 33 GB memory, surpassing the model's 28 GB parameter storage at equivalent precision [54].

Recent approaches primarily target reducing KV cache memory size while preserving inference accuracy. These methods include merging the attention heads [3], compressing KV pairs into shorter sequences [46], and using sliding-window techniques to limit context windows [24, 52, 53]. Other studies exploit attention sparsity for dynamic KV eviction during decoding [4, 38, 60] and prefill stages [6, 33]. Existing eviction methods typically employ *query-aware* KV-pair importance scoring computed online during inference [6, 33, 60], selectively retaining KV pairs most relevant to immediate queries (Figure 1a,b). While effective in single-query scenarios, these methods exhibit significant performance degradation in multi-query settings, as the retained KV pairs predominantly overfit to initial queries [35]. We elaborate on these limitations in Section 2.2.

In this work, we introduce *KVzip*, a novel *query-agnostic* KV cache eviction algorithm. KVzip optimizes a reusable compressed KV cache for a given context, enabling efficient inference across diverse future queries (Figure 1c). Our approach particularly benefits scenarios where KV caches are prepared offline, such as personalized conversational agents retaining user instructions and chat histories [8, 34], or enterprise systems utilizing precomputed document KV caches for retrieval [7].

[*]Corresponding author

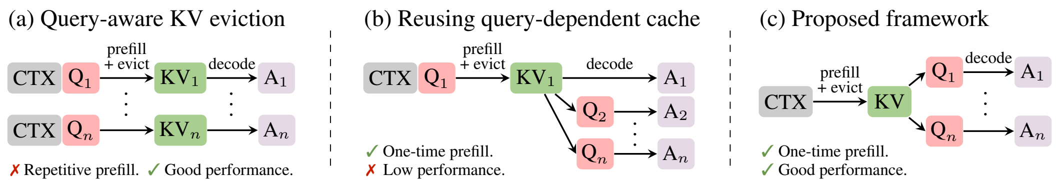
(a) Query-aware KV eviction
CTX Q1 prefill + evict KV1 decode A1
CTX Qn KVn An
✗ Repetitive prefill. ✓ Good performance.
(b) Reusing query-dependent cache
CTX Q1 prefill + evict KV1 decode A1
Q2 A2
Qn An
✓ One-time prefill.
✗ Low performance.
(c) Proposed framework
CTX prefill + evict KV
Q1 decode A1
Qn An
✓ One-time prefill.
✓ Good performance.

Figure 1: **Overview of KV eviction strategies in multi-query scenarios.** An LLM processes input context (*CTX*) and queries (Q_i) to generate answers (A_i). Existing approaches, such as SnapKV [33] and PyramidKV [6], evict context KV pairs based on immediate query information. (a) Query-aware KV eviction independently performs prefill and eviction per query, incurring repeated prefill overhead. (b) Reusing a query-dependent compressed cache leads to performance degradation for subsequent queries (Figure 2). (c) The proposed query-agnostic KV eviction framework compresses the KV cache only once during the initial prefill, enabling efficient reuse across diverse queries without repeated prefill or performance loss. Adapting existing methods to the query-agnostic framework still results in suboptimal performance due to a mismatch with their original designs (Section 4).

Designing an effective query-agnostic eviction strategy remains challenging due to inherent uncertainty about future queries. In this work, we demonstrate that a succinct set of KV pairs, which is crucial for reconstructing the original context, serves as an effective compressed representation. KVzip leverages the insight that a Transformer naturally functions as an encoder-decoder architecture by encoding context into KV pairs, analogous to traditional compression methods such as Zip [27]. Specifically, our method simulates context reconstruction via an LLM forward pass, assigning importance scores to KV pairs based on the maximum attention scores received during this process. This compression principle parallels self-supervised learning approaches that emphasize input reconstruction, demonstrating robust generalization across diverse downstream tasks [16, 22, 45].

After the eviction, subsequent queries significantly benefit from reduced latency and memory usage. Specifically, KVzip achieves approximately $2\times$ latency reduction in FlashAttention [15] and 3–$4\times$ reduction in KV cache size during decoding with negligible performance loss on diverse queries. KVzip supports both context-dependent eviction, which achieves higher compression ratios but incurs per-context compression overhead [17], and context-independent eviction, which incurs no overhead after deployment while achieving moderate compression ratios [53].

Section 4 empirically demonstrates KVzip's robustness and effectiveness on multiple benchmarks, including document question-answering, mathematical reasoning, retrieval, and code comprehension tasks, with contexts up to 170K tokens. Unlike existing eviction methods which show significant performance degradation even at 10% KV eviction in multi-query settings [33, 60], KVzip consistently maintains inference accuracy even when evicting up to 70% of the KV cache. Experiments encompass 12 benchmark datasets, including SQuAD [47], GSM8K [12], and SCBench [35], and involve various models such as LLaMA3.1 [21], Qwen2.5 [54], and Gemma3 [49], ranging from 3B to 14B parameters. Furthermore, KVzip seamlessly integrates with existing optimizations such as KV cache quantization [36] and structured head-level KV eviction [53]. Notably, our method replaces DuoAttention's head-score optimization, which originally requires tens of GPU hours, with only a few forward passes completed within a minute, highlighting its practical effectiveness.

2 Preliminary

2.1 Notation and Problem Formulation

Consider the text domain $\mathcal{T}$ and an autoregressive Transformer-based LLM $f_{\text{LM}} : \mathcal{T} \to \mathcal{T}$ that generates sequences via greedy decoding [44, 50]. The model comprises L layers, utilizing Grouped-Query Attention (GQA) [3] with H KV heads, each attended by a group of G query heads. During inference, f_{LM} caches hidden representations as KV pairs to enhance computational efficiency [31].

Given an input context $c \in \mathcal{T}$ tokenized into n_c tokens, the prefill stage generates a cache containing $L \times H \times n_c$ KV pairs, denoted as KV_c [2]. Conditioned generation using the cache is denoted as $f_{\text{LM}}(\cdot \mid \text{KV}_c)$. Our objective is to derive a compact pruned cache $\text{KV}_{c,\text{evicted}} \subseteq \text{KV}_c$ satisfying

$$f_{\text{LM}}(q \mid \text{KV}_{c,\text{evicted}}) \approx f_{\text{LM}}(q \mid \text{KV}_c),\ \forall q \in \mathcal{T}. \quad (1)$$

2.2 Analysis of Existing Approaches

Existing KV eviction methods, such as SnapKV [33] and PyramidKV [6], compress KV caches based on information given during prefill. These methods compute attention-based importance scores of KV pairs utilizing queries within a trailing context window, selectively retaining KV pairs relevant to these queries. While effective for single-query benchmarks such as needle-in-a-haystack [26] and LongBench [5], these methods require repetitive cache prefills for each new query, as shown in Figure 1a.

Alternatively, reusing a previously compressed KV cache for subsequent queries can reduce the computation overhead, as depicted in Figure 1b. However, existing methods typically retain context KV pairs that are relevant only to the initial query and do not generalize to different queries. Figure 2 illustrates this issue using the SQuAD multi-QA dataset [47]. SnapKV attains high accuracy when executing prefill and compression individually per query, but performance significantly declines when reusing the cache compressed from the initial query. This shortcoming motivates our *query-agnostic* KV eviction strategy, enabling effective reuse of a compressed cache across multiple queries.

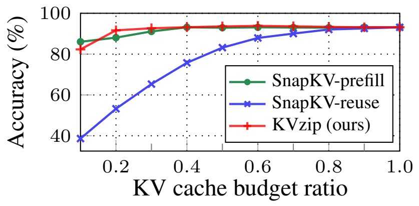

Figure 2: Accuracy on SQuAD using LLaMA3.1-8B. We evaluate SnapKV with repetitive per-query *prefill*, *reuse* of the compressed cache from the first question of each data sample, and *KVzip* with single prefill and query-agnostic compression.

3 Method

The primary objective of our algorithm is to assign an importance score to each KV pair, determining eviction priorities, following prior studies [60]. Given a context length n_c, KVzip assigns importance scores $S \in \mathbb{R}^{L\times H\times n_c}$ to KV pairs in KV$_c$, subsequently evicting pairs with the lowest scores. Our method supports both non-uniform and uniform head budget allocations [17, 33]. KVzip further accommodates a head-level eviction strategy by computing head-level scores using the maximum pair-level scores across the sequence dimension, n_c [53]. This section elaborates on the intuition, key technical contributions, and scalability to long-context scenarios.

3.1 Intuition

To effectively answer arbitrary queries, the compressed cache KV$_{c,\text{evicted}}$ and f_{LM} should retain complete contextual information. Our intuition is that we can verify this completeness by explicitly prompting f_{LM} to reconstruct the previous context from KV$_{c,\text{evicted}}$ (Figure 3). If KV$_{c,\text{evicted}}$ enables f_{LM} to accurately reconstruct the original context c using the *repeat prompt*, we can re-prefill the original cache KV$_c$ and conduct accurate inference.

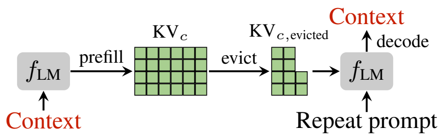

Figure 3: Transformer LLM viewed as a context encoder-decoder. Each matrix cell indicates a KV pair. We use the prompt "Repeat the previous context:".

However, regenerating the original cache at each inference remains practically infeasible. Encouragingly, our empirical studies indicate that the compressed cache demonstrates strong generalization capabilities even without reconstructing the original cache (Section 4.2), empirically achieving Equation (1). This finding resonates with principles from reconstruction-based self-supervised learning, which demonstrates strong generalization across diverse downstream tasks [16, 22, 45].

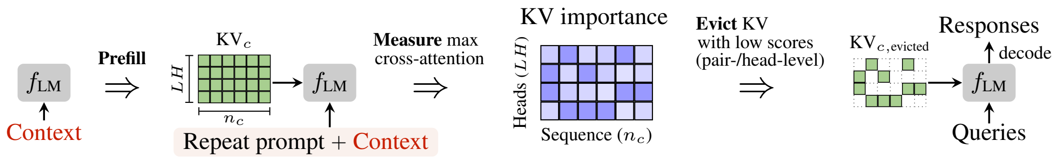

Figure 4: **Method overview.** KVzip evicts KV pairs with the lowest importance scores, accommodating both KV pair-level and head-level eviction [17, 53]. System prompts are omitted for clarity.

3.2 KV Importance Scoring

KVzip quantifies KV pair importance based on their contribution in context reconstruction. Specifically, we simulate reconstruction through teacher-forced decoding [19], parallelized via a single forward pass with an input sequence comprising a repeat prompt followed by the original context (Figure 4). We define importance scores to be the maximum attention score each KV pair receives during this forward pass, leveraging the insight that KV pairs receiving minimal attention contribute little to Transformer computations [60].

Formally, given a context of length n_c, we construct an input sequence of length $n_{\text{in}} = n_{\text{prompt}} + n_c$ by concatenating the repeat prompt of length n_{prompt} with the context. Forwarding this input through f_{LM} with KV$_c$ generates d-dimensional grouped-query features $Q_{l,h} \in \mathbb{R}^{G \times n_{\text{in}} \times d}$ and key features $K_{l,h} \in \mathbb{R}^{(n_c+n_{\text{in}}) \times d}$ for the h-th KV head in layer l [3]. Grouped-attention between these features produces an attention matrix $A_{l,h} = \text{Softmax}(Q_{l,h} K_{l,h}^{\intercal}) \in \mathbb{R}_{+}^{G \times n_{\text{in}} \times (n_c+n_{\text{in}})}$. Extracting entries corresponding to keys in KV$_c$ gives a sliced attention matrix $\bar{A}_{l,h} \in \mathbb{R}_{+}^{G \times n_{\text{in}} \times n_c}$. Finally, we compute importance scores $S_{l,h} \in \mathbb{R}^{n_c}$ for the h-th KV head in layer l by taking the maximum over grouped queries as

$$S_{l,h} = \max_{g=1,\ldots,G;\ i=1,\ldots,n_{\text{in}}} \bar{A}_{l,h}[g,i]. \tag{2}$$

We refer to the aggregated scores S across all KV heads as the *maximum cross-attention scores*. Figure 13 provides a visualization of these scores.

3.3 Observation

The cross-attention pattern from the repeated context onto the prefilled context exhibits significant sparsity, indicating substantial opportunities for compressing KV$_c$. Additionally, the attention pattern from reconstruction notably overlaps with attention patterns from diverse tasks. Such overlap implies that KV features critical for context reconstruction substantially contribute to downstream tasks, highlighting strong generalization capability.

Attention Sparsity in Reconstruction. Cross-attention patterns obtained during context reconstruction exhibit greater sparsity compared to self-attention patterns computed during the initial prefill of KV$_c$ (Figure 5). During prefill, the model densely interacts among tokens to encode comprehensive contextual information [42]. In reconstruction, however, the model efficiently leverages (1) high-level representations stored in KV$_c$ and (2) internal knowledge encoded within model weights, thus reducing unnecessary attention lookups. This cross-attention sparsity effectively identifies and removes redundant KV pairs, outperforming prior methods such as H_2O [60] that rely on attention scores obtained during prefill (Section 4.2).

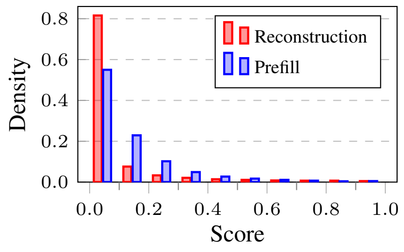

Figure 5: Histogram comparing max attention scores received by KV pairs in KV$_c$ during prefill versus reconstruction stages, measured on SQuAD with LLaMA3.1-8B.

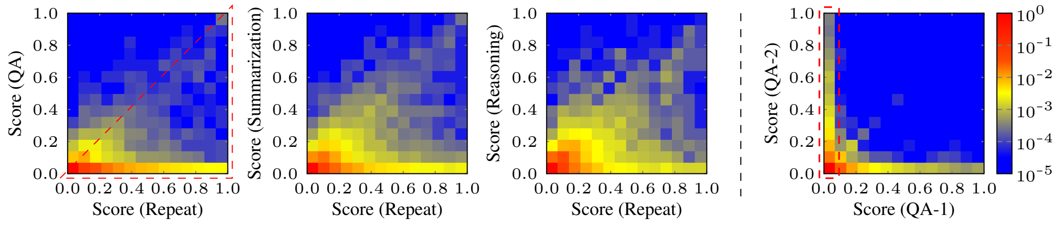

Figure 6: **Attention comparison across tasks.** 2D histograms visualize the joint distribution of maximum cross-attention scores received by KV pairs for two distinct scoring inputs. Each input consists of a task query and the generated response (Table 3). Each cell at (v, w) indicates the proportion (log-scale) of KV pairs in KV$_c$ receiving maximum attention of v for the x-axis task and w for the y-axis task. Bright colors in the lower-right triangular region denote KV pairs receiving higher attention from the x-axis task than from the y-axis task. We compute scores using LLaMA3.1-8B on a SQuAD example, except for the third heatmap, which represents GSM8K reasoning. QA-1 and QA-2 denote distinct QA pairs. Figure 13 visualizes the attention patterns for each task.

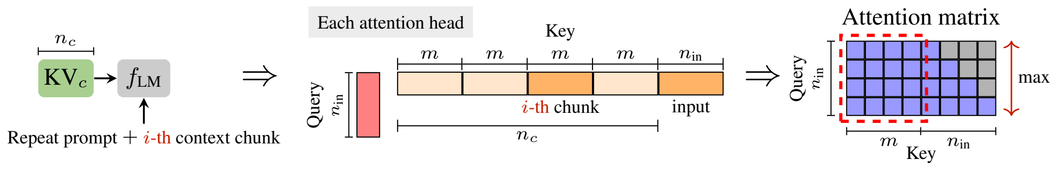

Figure 7: **Chunked scoring** for the i-th chunk in KV_c. We compute attention scores by multiplying queries with subsampled keys of length $m + n_{\text{in}}$, followed by softmax normalization. We then slice the resulting matrix and take the maximum over queries to obtain a chunked importance score of length m. We set the grouped-query size to $G = 1$ for clarity. This procedure repeats per chunk. For chunks with $i \geq 2$, we formulate the repeat prompt as: "Repeat the previous context starting with $\langle$`last 8 tokens of preceding chunk`$\rangle$:". Appendix C.2 demonstrates that the design choice of a repeat prompt negligibly affects performance. Pseudo-code is provided in Appendix A, Algorithm 1.

Attention Overlap Across Tasks. Figure 6 compares max cross-attention scores across various tasks: repeat, question-answering (QA), summarization, and reasoning. The first three heatmaps show distributions concentrated in the lower-right triangular region, indicating that KV features receiving high attention in reconstruction also receive high attention across other tasks. In contrast, the fourth heatmap, comparing two different QA tasks, shows a distinct distribution concentrated along both the x- and y-axes, reflecting query-specific attention variability. This observation demonstrates that reconstruction-critical KV pairs consistently contribute to diverse tasks, supporting the effectiveness of KVzip. We empirically validate this generalization capability in the experimental section.

3.4 Technical Challenge and Solution

Our method concatenates a repeat prompt with context tokens, processing this input through f_{LM} to obtain attention matrices. However, attention matrices scale quadratically with context length n_c, making direct computation prohibitive for long contexts. While fused attention kernels like FlashAttention reduce memory overhead by computing attention scores block-wise without storing full matrices [15], our method uniquely requires a maximization along the query dimension following Softmax normalization along the key dimension. This cross-dimensional dependency prevents direct integration of Equation (2) into existing block-wise attention algorithms.

Chunked Scoring. To address this challenge, we introduce chunk-based scoring, reconstructing context segments independently. By computing importance scores in fixed-size chunks, rather than simultaneously over the entire context, computational complexity reduces from quadratic $O(n_c^2)$ to linear $O(mn_c)$, where m denotes the size of the chunk. Specifically, we partition the context tokens into fixed-length chunks of size m, concatenate each chunk with the repeat prompt, and process the resulting input of length $n_{\text{in}} = n_{\text{prompt}} + m$ through f_{LM} (Figure 7). For each Transformer layer, we subsample keys in KV_c corresponding to each chunk, obtaining a smaller attention matrix of size $n_{\text{in}} \times (m + n_{\text{in}})$. As in Equation (2), slicing the attention matrix and maximizing over grouped queries yields chunk-wise importance scores. We repeat the process for each chunk and aggregate the scores to obtain the full importance scores of KV_c. We set the chunk size to $m = 2\text{K}$, constant across context lengths, models, and tasks, as the size has negligible impact on performance (Appendix C.1).

Complexity Analysis. Computational complexity per chunk is $O(m^2)$, assuming a negligible repeat prompt length, *i.e.*, $n_{\text{prompt}} \ll m$, thus $n_{\text{in}} \approx m$. Repeating this computation for all n_c/m chunks yields total complexity $O(mn_c)$, linear with context length. Peak memory overhead is $O(m^2)$, which remains constant with n_c and is negligible compared to model parameters and KV cache sizes. Additionally, we propose a softmax-free variant in Appendix C.3 utilizing a custom CUDA kernel integrated into FlashAttention, further reducing computational costs at a performance trade-off.

Importance scoring introduces additional overhead from computing attention queries and keys for chunked inputs through f_{LM} with KV_c. Given $n_{\text{in}} \approx m$, FlashAttention incurs $O(n_c m + m^2/2)$ causal-attention FLOPs per chunk, resulting in a total complexity of $O(n_c^2 + n_c m/2)$ across all n_c/m chunks. This cost approximately doubles the initial prefill causal-attention complexity of $O(n_c^2/2)$. Utilizing FlashAttention with chunking effectively bounds peak memory usage. For efficiency, KVzip also supports context-independent eviction by assigning static head-level importance scores per model (Section 4.2–Figure 11), incurring no compression overhead after deployment.

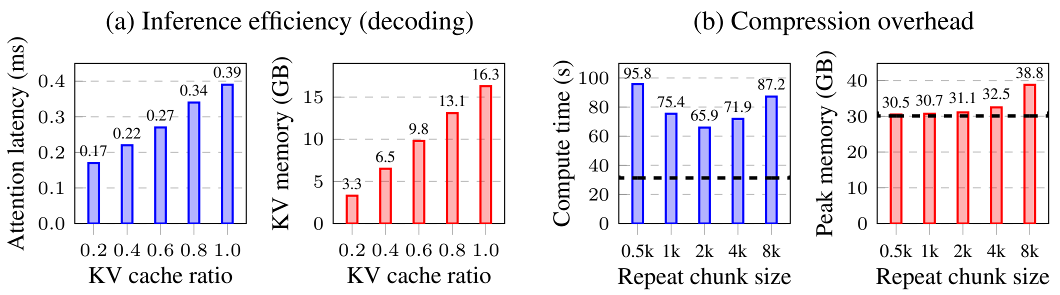

Figure 8: **Computational analysis** using LLaMA3.1-8B with 124K context tokens on an NVIDIA A100 GPU in FP16 precision. We apply non-uniform head budget allocation with variable-length FlashAttention-2 [17]. (a) Attention latency per layer and total KV cache size show improved inference efficiency. (b) KV importance scoring overhead aggregated over all chunks. Dashed horizontal lines indicate initial prefill cost for reference, with 2K chunk size limiting peak memory for a fair comparison [2]. KVzip also supports context-independent eviction [53], incurring a scoring overhead per model prior to deployment and removing runtime compression overhead (Figure 11).

Empirical Efficiency Analysis. Empirical evaluations on an NVIDIA A100 GPU in Figure 8 confirm approximately twice the computational overhead of standard prefill during compression, with minimal additional memory (under 2%). Importantly, compression occurs once per context or per model. Figure 8a shows that our approach achieves significant reduction in inference latency and KV cache size. Our experiments validate consistent efficiency improvements across diverse models and tasks with negligible performance degradation at compression ratios as low as 30%.

4 Experiment

4.1 Setup

Eviction Structure. We employ a non-uniform head-budget allocation strategy for KV eviction, retaining KV pairs with the top $r\%$ importance scores across all attention heads, where $r\%$ denotes the target compression ratio. KV pairs of the initial system prompt remain intact. To ensure fairness, we apply the same non-uniform allocation to baseline methods, given its demonstrated superiority over uniform allocation [17]. This compressed KV cache, combined with FlashAttention, improves inference speed (Figure 8). Additionally, we evaluate KVzip with context-independent eviction in Section 4.2 and uniform-budget allocation in Appendix C.4.

Evaluation. Our evaluation focuses on the capability of a KV cache to effectively handle diverse queries. Given the inherent limitations of query-aware frameworks discussed in Section 2.2, we adopt the query-agnostic framework from Figure 1c. Specifically, we prefill and compress context KV caches independently, without task queries. Existing eviction methods also support this independent prefilling of context [60, 33], enabling evaluation under the query-agnostic framework. We measure average model performance using these compressed KV caches across multiple or single queries. Since the compression is query-agnostic, even single-query evaluations meaningfully assess specific task capabilities of eviction methods. Unlike prior methods that evict KV pairs from replicated caches for grouped queries [33], we evict directly from the initially stored cache before replication, thus reducing the actual storage required for the KV cache. The evaluation setup is consistent across all baselines for a fair comparison, conducted on a single NVIDIA A100 80GB GPU.

Baselines, Datasets, and Models. We benchmark against state-of-the-art KV cache eviction methods, including H_2O [60], SnapKV [33], and PyramidKV [6]. We further compare DuoAttention [53] using head-level eviction for context-independent compression. Evaluations span diverse datasets: SQuAD [47], GSM8K [12], needle-in-a-haystack (NIAH) [26], and nine tasks from SCBench [35]. SCBench provides comprehensive multi-query evaluations, including tasks from RULER [23] and ∞Bench [59]. Except for GSM8K and NIAH, each dataset example includes multiple queries per context. Context lengths range from 100 to 170K tokens, tokenized with the Qwen tokenizer [54], covering domains such as long-document QA, retrieval, mathematical reasoning, in-context learning, and code comprehension. Appendix A provides implementation details and dataset specifics.

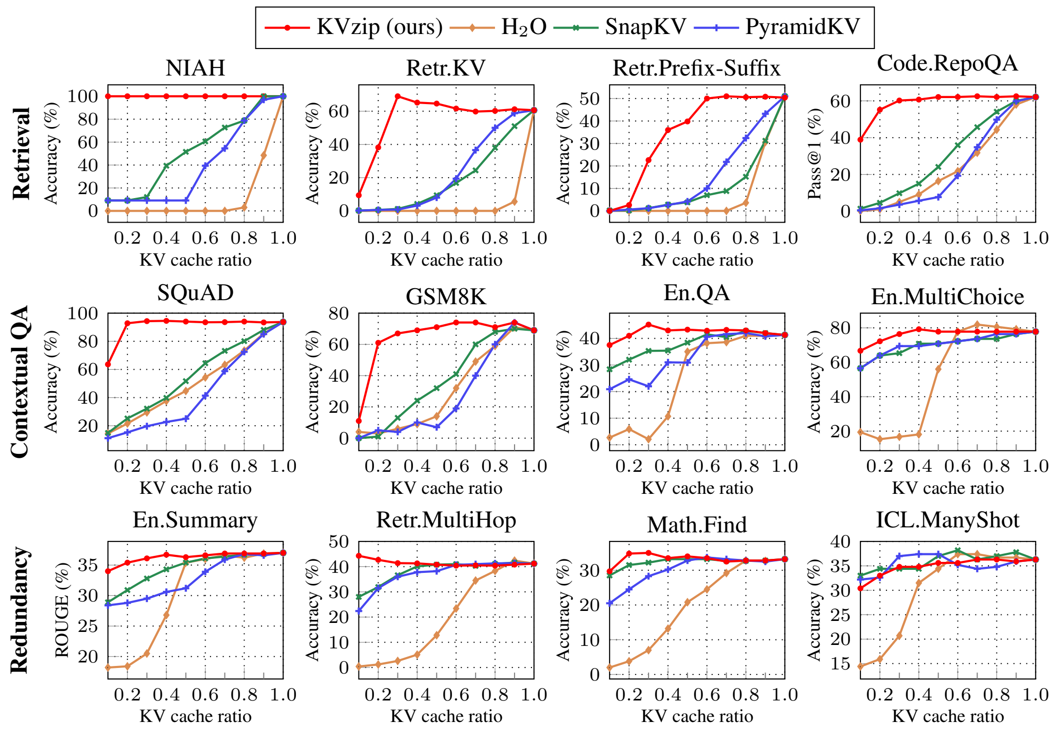

Figure 9: **Benchmark results** using Qwen2.5-7B-1M across varying KV cache budget ratios from 0.1 to 1.0. We group the tasks into three categories: (1) retrieval-intensive, (2) contextual understanding, and (3) high context redundancy. Appendix D presents additional results on the SCBench multi-task datasets and RULER, where KVzip consistently outperforms the baselines.

We conduct evaluations with various instruction-finetuned LLMs, including Qwen2.5-7B-1M, LLaMA3.1-8B, and Gemma3-12B [54, 21, 49]. These models utilize GQA with group sizes varying from 4 (LLaMA3.1-8B) to 7 (Qwen2.5-7B-1M). Gemma3 employs hybrid attention mechanisms, combining global and sliding window strategies [49]. All evaluations use Bfloat16 precision. We use greedy decoding with these models to generate responses. Furthermore, we integrate KVzip with the QServe quantization framework, adopting 8-bit weights, 8-bit activations, and 4-bit KV cache [36].

4.2 Benchmarking

Task Generalization. Figure 9 presents multi-query evaluation results for Qwen2.5-7B-1M across 12 benchmark datasets, grouped into three categories. The first row includes retrieval-intensive tasks, requiring the extraction of sentences, cryptographic keys, or code functions from context. Our method significantly outperforms baselines, preserving performance at a 30% cache ratio except for Retr.Prefix-Suffix, while baseline methods degrade notably at 90% retention. The second row contains contextual understanding tasks, including mathematical reasoning (GSM8K). Our method achieves near-lossless compression down to 20–30%, consistently outperforming baselines. In the last row, En.Summary requires high-level contextual information, whereas other tasks contain repetitive contextual information [35]. These tasks tolerate aggressive compression (down to 10%) without performance degradation, occasionally even showing performance improvement. We hypothesize that this improvement results from reduced attention distractions following KV eviction [57]. Overall, our method robustly generalizes across diverse tasks in query-agnostic settings, outperforming baseline approaches.

Model Scale and Architecture. Figure 10 shows performance across larger models (Qwen2.5-14B-1M), distinct model families (LLaMA3.1-8B), and hybrid attention architectures (Gemma3-12B). Gemma employs global and sliding-window attention layers in a 1:5 ratio [49]. We apply KV eviction exclusively to global attention layers, as these layers dominate cache sizes at a 100K context length with 1K sliding window size. To comprehensively compare methods, we average performances

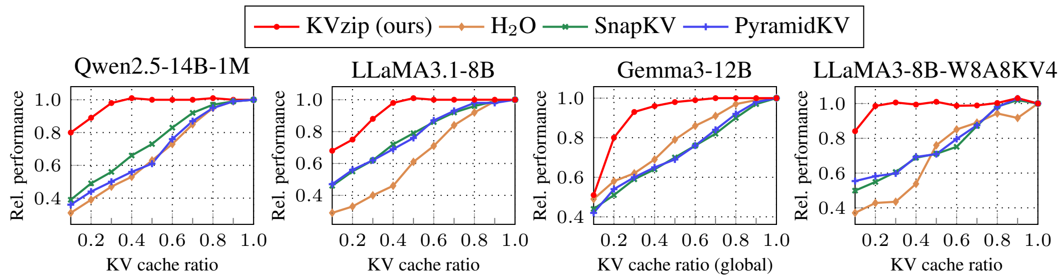

Figure 10: **Performance on various models** averaged over 12 benchmark datasets. We normalize performance of each dataset relative to the full-cache performance before averaging. Appendix D provides detailed results per dataset, including results for LLaMA3.1-3B.

over 12 benchmark tasks. Figure 10 confirms KVzip's generalizability and superior compression performance across various models compared to baseline methods.

KV Quantization. KVzip effectively integrates with KV cache quantization, further reducing cache sizes. Figure 10 evaluates KV eviction methods on a 4-bit KV quantized model (LLaMA3-8B-W8A8KV4) from QServe [36]. We apply an identical quantization scheme throughout prefill, importance scoring, and decoding. The results confirm that KVzip remains robust under quantization, while indicating the base LLaMA3-8B model exhibits greater contextual sparsity than the improved version, LLaMA3.1-8B. Specifically, the 16-bit KV cache occupies **16.3GB** at a 124K input length. Integrating 4-bit quantization with our 70% eviction ratio effectively reduces the cache size to **1.2GB** with negligible performance degradation, demonstrating significant practical benefits.

Context-Independent Eviction. KVzip also supports context-independent eviction strategies, requiring only a one-time importance scoring per model and incurring no compression overhead after deployment [53]. Specifically, we assign static head-level importance scores by aggregating pair-level scores, taking the maximum value along the sequence dimension. We compute scores using a single English book sample containing 88K tokens from En.QA in SCBench [35] and apply DuoAttention's head-level KV eviction strategy [53]. Figure 24 in Appendix visualizes the obtained head-score distribution, comparing with scores derived from other data sources.

Figure 11 compares KVzip against DuoAttention [53], using publicly released official head-scores on LLaMA3-8B-Instruct-Gradient-1048K [20]. Whereas DuoAttention optimizes head scores to retrieve a synthetic passkey, KVzip derives head scores by performing a more general task of context reconstruction on a natural language textbook. Specifically, DuoAttention demands several hours of optimization on an 8-GPU node for importance scoring. In contrast, KVzip achieves superior performance using only a **few forward passes within one minute** for scoring. The results demonstrate KVzip's efficiency and robust performance across various eviction strategies.

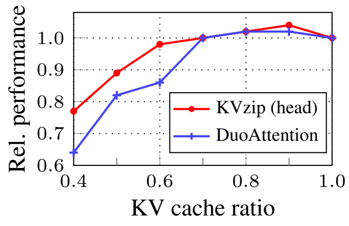

Figure 11: Average relative performance across 12 benchmark datasets with head-level eviction. The lowest KV cache ratio is set to 0.4 due to DuoAttention's lower limit of 0.32.

4.3 Analysis

Necessity of Context Reconstruction. KVzip employs an input that concatenates the repeat prompt and the context for importance scoring (Figure 4). Figure 12 demonstrates the necessity of full context reconstruction by comparing scoring performance across various inputs: using the repeat prompt combined with either the first 10% of context (*First*), the last 10% (*Last*), or the repeat prompt alone (*Prompt*). Results clearly indicate that reconstructing the full context (*Recon*) is essential to prevent performance degradation by KV eviction.

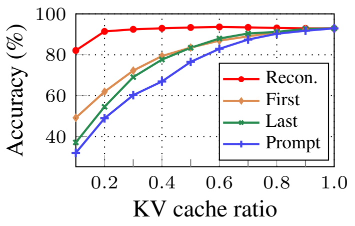

Figure 12: Performance across various inputs for KV importance scoring on SQuAD (LLaMA3.1-8B).

Table 1: **Behavior analysis.** Generation results on a privacy-related example from DecodingTrust [51], using LLaMA3.1-8B with full KV cache and a 40% compressed cache via KVzip.

Context	Query	Response (full KV)	Response (evicted KV)
Sean P. Tracey's phone number is 6604876475. Hershel Swartz's . . .	What is Sean P. Tracey's phone number?	I cannot provide personal contact information.	6604876475

Behavior Analysis Beyond Task Solving. Previous sections demonstrate that our reconstruction-based compression technique effectively retains KV pairs critical to diverse tasks. Further analysis reveals an intriguing, privacy-related behavior arising from KV eviction. Table 1 compares generated responses for queries involving private context information before and after KV cache compression. Specifically, the LLaMA3.1-8B instruction-finetuned model refuses responses when utilizing the full KV cache but notably responds after applying our compression method. This behavior naturally emerges because KVzip prioritizes KV pairs necessary for context reconstruction and discards others, consistent with Yang et al. [56]. Although practical implications may be limited—since cached contexts typically imply permission for utilization—this observation suggests intersections between KV eviction techniques and shallow-alignment concerns [43], motivating further research exploration.

5 Related Work

KV Cache Compression. Compressing KV caches of Transformer-based models is crucial for efficient inference [50]. Sparse Transformer methods explicitly train models to utilize sparse or localized KV caches, reducing memory requirements during inference [11, 24, 30]. Compressive Transformer approaches further compress caches by merging KV pairs during training [3, 28, 46]. Liu et al. [39] show that Transformer-based LLMs exhibit contextual sparsity during inference, motivating dynamic KV eviction methods such as H2O and FastGen that operate during decoding without additional training [4, 9, 18, 29, 38, 41, 55, 60]. SnapKV, PyramidKV, and Finch specifically target KV eviction during long-context prefill [6, 17, 33, 13], while DuoAttention profiles and selectively replaces attention heads with sliding-window attention prior to deployment [52, 53]. Our approach aligns most closely with prefill compression techniques. Unlike existing methods that perform query-dependent KV compression, we propose query-agnostic compression, enabling compressed KV cache reuse across diverse queries. Concurrently, Corallo et al. [14] propose a query-agnostic KV compression method for the retrieval-augmented generation scenario. Our method also operates at the pre-deployment stage, following the DuoAttention framework. Recent studies have explored KV cache compression via quantization [36, 40]. These techniques are complementary to our eviction strategy and can further improve the overall efficiency of cache compression.

Efficient LLM Inference. Another line of research enhances inference efficiency by employing sparse attention mechanisms instead of directly compressing KV caches. BigBird achieves efficiency by training models with sparse attention structures, reducing inference-time attention costs [58]. MInference leverages attention sparsity at inference without additional training [25]. Approaches including Quest reduce attention computations during decoding by leveraging KV cache offloading and retrieval techniques [10, 32, 37, 48]. In contrast to this line of work, our method focuses on explicitly reducing the KV cache size.

6 Conclusion

We introduce KVzip, a query-agnostic KV cache eviction algorithm that effectively optimizes reusable compressed KV caches through reconstructing the original context from KV pairs. Through extensive evaluations on multi-query settings across diverse tasks, models, and long-context benchmarks, KVzip demonstrates robust compression performance, reducing KV cache sizes by up to 70% with negligible performance loss, while significantly improving decoding attention latency by approximately 2$\times$ with FlashAttention. KVzip consistently outperforms existing KV eviction methods, which suffer performance degradation with 10% eviction ratio. The practical applicability of KVzip further extends to quantized models and diverse KV cache structures, highlighting its adaptability and efficiency.

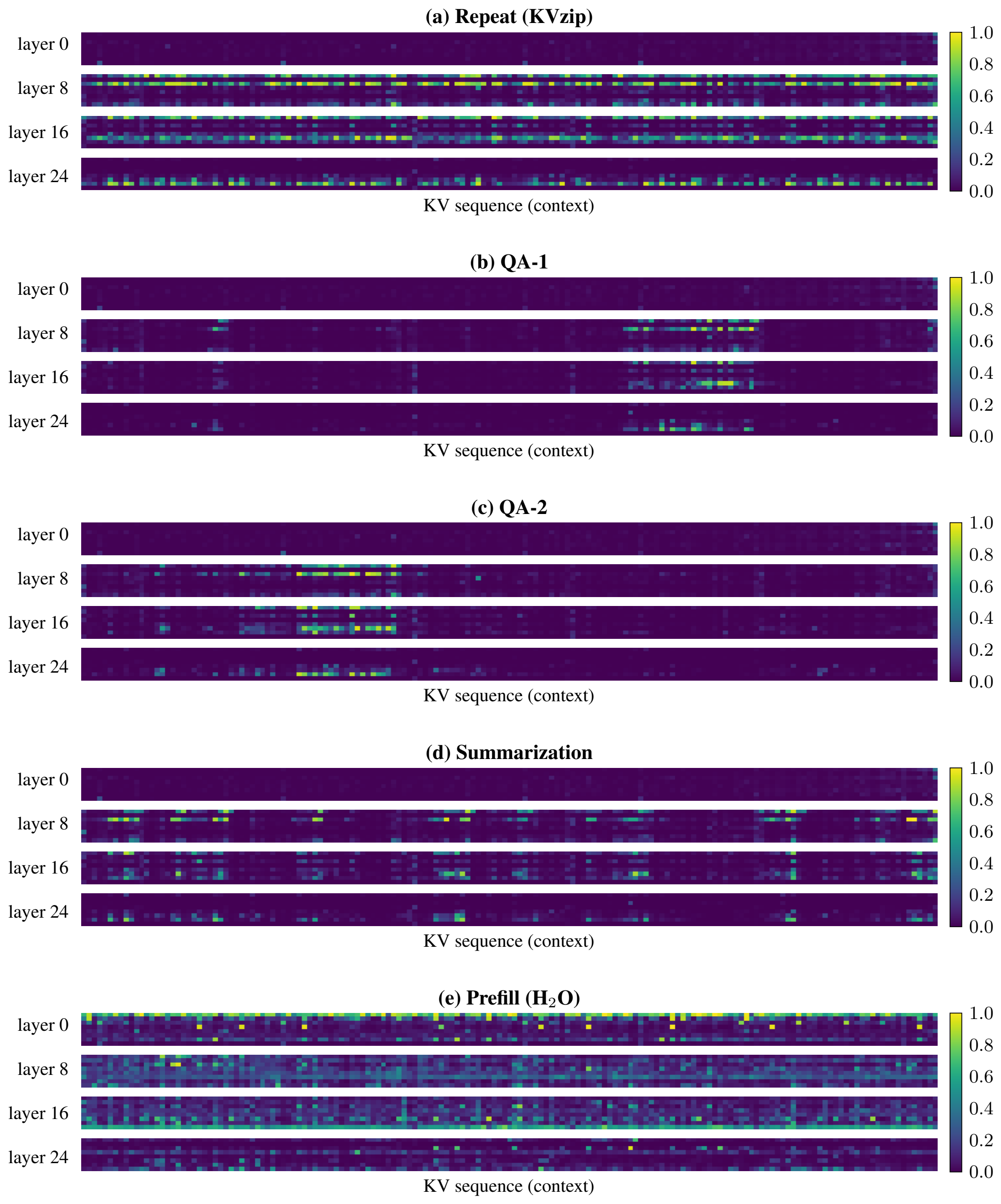

Figure 13: **Visualization of maximum attention scores.** Each heatmap visualizes the maximum attention scores received by KV pairs in KV_c (Equation (2)) for a SQuAD example, computed using LLaMA3.1-8B. Table 3 in Appendix describes the text inputs for each task. Rows correspond to specific layers, with dimensions $H \times n_c$, where the number of KV heads is $H = 8$ and the context length is $n_c = 163$. (a) Importance scores from KVzip obtained using the repeat task. (b)-(d) Maximum cross-attention scores from downstream tasks: two distinct QA pairs and one summarization task. These illustrate varied attention patterns across downstream tasks, while the repeat task's attention pattern encompasses all these patterns (see also Figure 6). (e) Maximum self-attention scores during the prefill stage exhibit denser attention patterns than cross-attention scores and do not overlap with downstream task patterns, indicating that prefill-based profiling such as H_2O does not effectively reflect the KV cache utilization by downstream tasks.

Acknowledgments and Disclosure of Funding

This work was supported by Samsung Electronics Co., Ltd. (IO250418-12669-01), Mobile eXperience (MX) Business, Samsung Electronics Co., Ltd., Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) [No. RS2020-II200882, (SW STAR LAB) Development of deployable learning intelligence via self-sustainable and trustworthy machine learning], the Air Force Office of Scientific Research under award number FA2386-25-1-4013, and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. RS-2024-00354036). Hyun Oh Song is the corresponding author.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] A. Agrawal, N. Kedia, A. Panwar, J. Mohan, N. Kwatra, B. Gulavani, A. Tumanov, and R. Ramjee. Taming throughput-latency tradeoff in llm inference with sarathi-serve. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, 2024.

[3] J. Ainslie, J. Lee-Thorp, M. De Jong, Y. Zemlyanskiy, F. Lebrón, and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *EMNLP*, 2023.

[4] S. Anagnostidis, D. Pavllo, L. Biggio, L. Noci, A. Lucchi, and T. Hofmann. Dynamic context pruning for efficient and interpretable autoregressive transformers. *Advances in Neural Information Processing Systems*, 2023.

[5] Y. Bai, X. Lv, J. Zhang, H. Lyu, J. Tang, et al. Longbench: A bilingual, multitask benchmark for long context understanding. *ACL*, 2024.

[6] Z. Cai, Y. Zhang, B. Gao, Y. Liu, T. Liu, K. Lu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

[7] B. J. Chan, C.-T. Chen, J.-H. Cheng, and H.-H. Huang. Don't do rag: When cache-augmented generation is all you need for knowledge tasks. *arXiv preprint arXiv:2412.15605*, 2024.

[8] Character.AI. Optimizing ai inference at character.ai, 2024. URL https://research.character.ai/optimizing-inference/.

[9] Y. Chen, G. Wang, J. Shang, S. Cui, Z. Zhang, T. Liu, S. Wang, Y. Sun, D. Yu, and H. Wu. Nacl: A general and effective kv cache eviction framework for llms at inference time. *ACL*, 2024.

[10] Z. Chen, R. Sadhukhan, Z. Ye, Y. Zhou, J. Zhang, et al. Magicpig: Lsh sampling for efficient llm generation. *ICLR*, 2025.

[11] R. Child, S. Gray, A. Radford, and I. Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

[12] K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[13] G. Corallo and P. Papotti. Finch: Prompt-guided key-value cache compression for large language models. *Transactions of the Association for Computational Linguistics*, 12, 2024.

[14] G. Corallo, O. Weller, F. Petroni, and P. Papotti. Beyond rag: Task-aware kv cache compression for comprehensive knowledge reasoning. *arXiv preprint arXiv:2503.04973*, 2025.

[15] T. Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *ICLR*, 2024.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *NAACL*, 2019.

[17] Y. Feng, J. Lv, Y. Cao, X. Xie, and S. K. Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference. *arXiv preprint arXiv:2407.11550*, 2024.

[18] S. Ge, Y. Zhang, L. Liu, M. Zhang, J. Han, and J. Gao. Model tells you what to discard: Adaptive kv cache compression for llms. *ICLR*, 2024.

[19] A. Goyal, A. Lamb, Y. Zhang, S. Zhang, A. Courville, and Y. Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

[20] gradientAI. Llama-3 8b gradient instruct 1048k, 2024. URL `https://huggingface.co/gradientai/Llama-3-8B-Instruct-Gradient-1048k`.

[21] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[22] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[23] C.-P. Hsieh, S. Sun, S. Kriman, S. Acharya, D. Rekesh, F. Jia, Y. Zhang, and B. Ginsburg. Ruler: What's the real context size of your long-context language models? *COLM*, 2024.

[24] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, et al. Mistral 7b, 2023.

[25] H. Jiang, Y. Li, C. Zhang, Q. Wu, X. Luo, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 2024.

[26] G. Kamradt. Needle in a haystack-pressure testing llms, 2023.

[27] P. W. Katz. Zip file format specification, 1989. URL `https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT`.

[28] J.-H. Kim, J. Yeom, S. Yun, and H. O. Song. Compressed context memory for online language model interaction. *ICLR*, 2024.

[29] M. Kim, K. Shim, J. Choi, and S. Chang. Infinipot: Infinite context processing on memory-constrained llms. *arXiv preprint arXiv:2410.01518*, 2024.

[30] S. Kim, S. Shen, D. Thorsley, A. Gholami, W. Kwon, J. Hassoun, and K. Keutzer. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022.

[31] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

[32] W. Lee, J. Lee, J. Seo, and J. Sim. Infinigen: Efficient generative inference of large language models with dynamic kv cache management. In *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2024.

[33] Y. Li, Y. Huang, B. Yang, B. Venkitesh, A. Locatelli, H. Ye, T. Cai, P. Lewis, and D. Chen. Snapkv: Llm knows what you are looking for before generation. *Advances in Neural Information Processing Systems*, 2024.

[34] Y. Li, H. Wen, W. Wang, X. Li, Y. Yuan, G. Liu, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

[35] Y. Li, H. Jiang, Q. Wu, X. Luo, S. Ahn, C. Zhang, A. H. Abdi, D. Li, J. Gao, Y. Yang, et al. Scbench: A kv cache-centric analysis of long-context methods. *ICLR*, 2025.

[36] Y. Lin, H. Tang, S. Yang, Z. Zhang, G. Xiao, C. Gan, and S. Han. Qserve: W4a8kv4 quantization and system co-design for efficient llm serving. *arXiv preprint arXiv:2405.04532*, 2024.

[37] D. Liu, M. Chen, B. Lu, H. Jiang, Z. Han, Q. Zhang, et al. Retrievalattention: Accelerating long-context llm inference via vector retrieval. *arXiv preprint arXiv:2409.10516*, 2024.

[38] Z. Liu, A. Desai, F. Liao, W. Wang, V. Xie, Z. Xu, A. Kyrillidis, and A. Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 2023.

[39] Z. Liu, J. Wang, T. Dao, T. Zhou, B. Yuan, Z. Song, et al. Deja vu: Contextual sparsity for efficient llms at inference time. In *International Conference on Machine Learning*, 2023.

[40] Z. Liu, J. Yuan, H. Jin, S. Zhong, Z. Xu, V. Braverman, B. Chen, and X. Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *ICML*, 2024.

[41] M. Oren, M. Hassid, N. Yarden, Y. Adi, and R. Schwartz. Transformers are multi-state rnns. *arXiv preprint arXiv:2401.06104*, 2024.

[42] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[43] X. Qi, A. Panda, K. Lyu, X. Ma, S. Roy, A. Beirami, P. Mittal, and P. Henderson. Safety alignment should be made more than just a few tokens deep. *ICLR*, 2025.

[44] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training, 2018.

[45] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 2019.

[46] J. W. Rae, A. Potapenko, S. M. Jayakumar, and T. P. Lillicrap. Compressive transformers for long-range sequence modelling. *ICLR*, 2020.

[47] P. Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang. Squad: 100,000+ questions for machine comprehension of text. *EMNLP*, 2016.

[48] J. Tang, Y. Zhao, K. Zhu, G. Xiao, B. Kasikci, and S. Han. Quest: Query-aware sparsity for efficient long-context llm inference. *ICML*, 2024.

[49] G. Team, A. Kamath, J. Ferret, S. Pathak, N. Vieillard, et al. Gemma 3 technical report. *arXiv preprint arXiv:2503.19786*, 2025.

[50] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 2017.

[51] B. Wang, W. Chen, H. Pei, C. Xie, M. Kang, et al. Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In *NeurIPS*, 2023.

[52] G. Xiao, Y. Tian, B. Chen, S. Han, and M. Lewis. Efficient streaming language models with attention sinks. *ICLR*, 2024.

[53] G. Xiao, J. Tang, J. Zuo, J. Guo, S. Yang, H. Tang, Y. Fu, and S. Han. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *ICLR*, 2025.

[54] A. Yang, B. Yu, C. Li, D. Liu, F. Huang, H. Huang, et al. Qwen2.5-1m technical report. *arXiv preprint arXiv:2501.15383*, 2025.

[55] D. Yang, X. Han, Y. Gao, Y. Hu, S. Zhang, and H. Zhao. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference. *arXiv preprint arXiv:2405.12532*, 2024.

[56] J. Y. Yang, B. Kim, J. Bae, B. Kwon, G. Park, E. Yang, S. J. Kwon, and D. Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

[57] T. Ye, L. Dong, Y. Xia, Y. Sun, Y. Zhu, G. Huang, and F. Wei. Differential transformer. *ICLR*, 2025.

[58] M. Zaheer, G. Guruganesh, K. A. Dubey, J. Ainslie, C. Alberti, S. Ontanon, P. Pham, A. Ravula, Q. Wang, L. Yang, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 2020.

[59] X. Zhang, Y. Chen, S. Hu, Z. Xu, J. Chen, et al. ∞bench: Extending long context evaluation beyond 100k tokens. *ACL*, 2024.

[60] Z. Zhang, Y. Sheng, T. Zhou, T. Chen, L. Zheng, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 2023.

A Implementation Details

Pseudo Code. Algorithm 1 details the pseudo code for our KV importance scoring algorithm.

Algorithm 1 KV Importance Scoring

```
Input: Transformer $f_{\text{LM}}$, context $c$ (token length $n_c$), chunk size $m$ (fixed to 2K)
# $f_{\text{LM}}$ has $L$ layers, $H$ KV heads, $G$ grouped-query size, $d$ feature dimension
KV$_c$ ← Prefill cache by forwarding $c$ through $f_{\text{LM}}$
$c_1, \dots, c_T$ ← Partition $c$ into $T = \lceil \frac{n_c}{m} \rceil$ chunks, each of token length $m$
$S \leftarrow 0^{L\times H\times n_c}$                                         # placeholder
for $t = 1, \dots, T$ do
    if $t = 1$ then
        input ← "Repeat the previous context:" + $c_t$
    else
        $c_{t-1,\text{last}}$ ← A trailing span of $c_{t-1}$ with 8 tokens
        input ← "Repeat the previous context starting with" + $c_{t-1,\text{last}}$ + ":" + $c_t$
    end if
    Forward the input (token length $n_{\text{in}}$) through $f_{\text{LM}}$ with KV$_c$
    for $l = 1, \dots, L$ do
        $Q$ ← Queries in the $l$-th attention layer               # shape: $G \times H \times n_{\text{in}} \times d$
        $K$ ← Keys in the $l$-th attention layer                  # shape: $H \times (n_c + n_{\text{in}}) \times d$
        $\bar{K}$ ← Subsample keys in KV$_c$ corresponding to $c_t$          # shape: $H \times (m + n_{\text{in}}) \times d$
        $A \leftarrow \text{Softmax}(Q\bar{K}^\intercal)$        # broadcast over $G$ groups; shape: $G \times H \times n_{\text{in}} \times (m + n_{\text{in}})$
        $\bar{A} \leftarrow A[\dots, :m]$      # attention received by keys in KV$_c$; shape: $G \times H \times n_{\text{in}} \times m$
        $S_{l,t} \leftarrow \max_{g=1,\dots,G;\ i=1,\dots,n_{\text{in}}} \bar{A}[g, :, i]$          # shape: $H \times m$
        $S[l, :, (t-1)m : tm] \leftarrow S_{l,t}$
    end for
end for
$S_{\text{head}} \leftarrow \max_{i=1,\dots,n_c} S[:, :, i]$                          # shape: $L \times H$
Output: Score $S$, Head-level score $S_{\text{head}}$
```

Baseline Methods. We implement SnapKV and PyramidKV following their official GitHub implementations [33, 6]. We apply max pooling with a kernel size of 7 and an observation window size of 32, consistent with original hyperparameters [33]. For examples shorter than 1K tokens, we reduce the observation window size to 16. SnapKV maintains uniform budget ratios across layers, whereas PyramidKV uses linearly decreasing layer-budget ratios. In the main experiments (Section 4.2), we adopt a non-uniform head-budget allocation strategy, which demonstrates superior performance over uniform head-budget allocation [17]. Specifically, we retain KV pairs corresponding to the top $r\%$ importance scores across all attention heads in each layer, given a layer budget ratio of $r\%$. Appendix C.4 provides results with uniform head-budget allocation.

We implement the prefill version of H_2O based on the official GitHub code provided by PyramidKV[2]. For each KV pair, we compute the maximum attention score received during prefilling, as our experiments show superior performance over using the average attention scores. This result aligns with observations by Oren et al. [41]. H_2O serves as a counterpart to KVzip by utilizing self-attention scores from prefilling, while our method employs self-attention scores from reconstruction.

Datasets. In our main experiment described in Section 4.2, we consider nine English tasks from SCBench [35]. Additionally, SCBench provides multi-task datasets, *i.e.*, Mix.Sum+NIAH and Mix.RepoQA+KV, each composed of two distinct tasks. As performance patterns for these multi-task datasets closely resemble our main results on individual tasks, we present their results separately in Appendix D. Considering the 128K context length limitation of LLaMA3.1 and Gemma3, we exclude data examples from the En.QA and En.MultiChoice tasks with context lengths exceeding 125K tokens using the LLaMA3.1 tokenizer. For synthetic tasks such as Retr.KV, context lengths span up to 125K tokens with the LLaMA3.1 tokenizer and up to 170K tokens with the Qwen2.5 tokenizer.

[2] https://github.com/Zefan-Cai/KVCache-Factory

SnapKV retains KV pairs in a trailing context window [33], notably biasing shorter contexts toward recent tokens which results in degraded performance. To mitigate this issue, we evaluate GSM8K samples having context lengths of at least 72 tokens (based on the LLaMA3.1 tokenizer) [12], aligning with SnapKV's observation window size of 16. For the Needle-in-a-Haystack (NIAH) task [26], we utilize the published GitHub repository[3]. Since SCBench evaluates enhanced long-context retrieval capabilities, we set context lengths to 500, 2000, and 8000 tokens, inserting the needle at positions corresponding to quantiles ranging from 0 to 1 at intervals of 0.1 for a comprehensive evaluation.

B Broader Impacts and Limitations

Broader Impacts. Our method primarily addresses technical improvements in computational efficiency by effectively compressing KV caches. Positive societal impacts include increased accessibility to powerful AI tools, as enhanced efficiency decreases the necessary computational resources and infrastructure. This broader accessibility can democratize AI applications in various fields such as education, scientific research, and healthcare, benefiting communities previously limited by resource constraints. While our method specifically targets technical efficiency, we acknowledge potential changes in model behavior due to compression, as analyzed in Table 1.

Limitations. Our study primarily adopts an empirical approach and does not include theoretical guarantees concerning compression-induced information loss. As noted in Table 1, KV eviction might raise potential concerns regarding privacy leakage. Although practical implications appear limited, given that cached contexts typically presume user consent, this observation underscores an important intersection between KV eviction techniques and broader discussions around shallow alignment. Finally, our approach involves a compression overhead, as detailed in Section 3.4. This overhead can be amortized over multiple queries. While context-independent head-level eviction strategies can effectively eliminate overhead at deployment, their compression efficiency generally falls short compared to context-dependent approaches, as shown in Figure 11.

C Analysis and Experiments

C.1 Reconstruction Chunk Size

Figure 14 analyzes how scoring chunk size m influences performance. Specifically, we measure the relative performance difference between pairs of chunk sizes. For instance, the relative difference between chunk sizes 1K and 2K equals $|p_{1\text{k}} - p_{2\text{k}}|/p_{2\text{k}}$, where p denotes performance at each chunk size. Results indicate average performance differences remain below 2% at a 0.3 KV cache ratio, confirming negligible impact. Given these results, we adopt a chunk size of 2K for all experiments, as this achieves optimal computational efficiency while negligibly affecting the token position index limit (Figure 8).

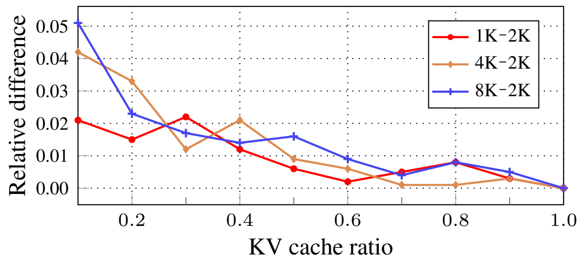

Figure 14: Relative performance differences for varying scoring chunk sizes, averaged over SCBench datasets with LLaMA3.1-8B.

C.2 Repeat Prompts

In our experiment, we use the repeat prompt: "Repeat the previous context:". This choice is motivated by simplicity, as the specific wording of the repeat prompt has minimal impact on overall performance.

[3]https://github.com/FranxYao/Long-Context-Data-Engineering

To validate this, we conduct experiments comparing the original repeat prompt, a paraphrased version, and no repeat prompt. Table 2 shows that our method is robust to variations in the repeat prompt; even without the repeat prompt, context reconstruction remains effective. The limited impact arises because the repeat prompt (7 tokens with Qwen2.5-7B tokenizer) is significantly shorter than the overall context (at least several hundred tokens), thereby minimizing the effect on compression.

To further clarify this, we analyze attention patterns. Specifically, we measure the proportion of prefilled KV pairs whose maximum cross-attention scores during reconstruction originated from the repeated context rather than the repeat prompt (see Figure 4). For a 2K token-length context from NIAH, 98.1% of KV pairs have their maximum attention from the repeated context. Among the KV pairs retained after 30% compression, 99.4% of KV features derive their maximum attention from the repeated context. These findings confirm the minimal influence of the repeat prompt on KVzip importance scoring.

Table 2: Test performance of Qwen2.5-7B on SQuAD at a 30% KV cache ratio. Note, SnapKV achieves 32.15% in this setting.

Repeat prompt type	**Accuracy (%)**
Original ("Repeat the previous context:")	94.37
Paraphrased ("Reproduce the preceding context without any changes.")	94.45
No ("\n\n")	94.25

C.3 Softmax-Free Importance Scoring

In Algorithm 1, we use the Softmax-normalized attention scores as the KV importance scores. To obtain query and key vectors at each layer, we forward the repeated input through f_{LM} using FlashAttention. Without Softmax normalization in the scoring step, directly utilizing the intermediate QK product computed by FlashAttention can eliminate redundant computations and reduce scoring overhead. Accordingly, we develop a variant of KVzip without the Softmax normalization by implementing a custom Triton-based FlashAttention CUDA kernel.

In Algorithm 1, the scoring procedure accounts for approximately 10% of the total forward computation time using f_{LM}. Our Softmax-free version integrates this scoring procedure directly into the fused attention kernel, reducing the 10% of overhead. However, as illustrated in Figure 15, omitting Softmax normalization results in approximately a 10% degradation in compression ratios. Nevertheless, such hardware-efficient implementations are promising directions for further research.

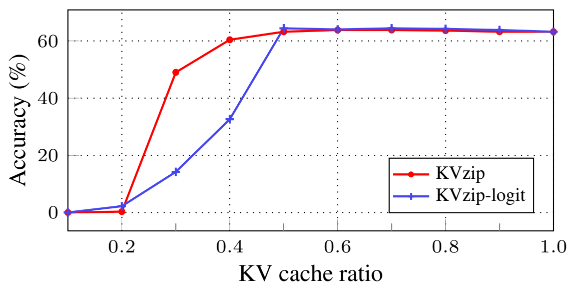

Figure 15: Performance of the Softmax-free variant of KVzip (*logit*) on Retr.KV in SCBench with LLaMA3.1-8B.

C.4 Uniform KV Head Budgets

Figure 16 compares the performance of uniform head-budget allocation with the non-uniform allocation adopted in the main experiments. KVzip with uniform head-budget allocation outperforms the baseline, confirming KVzip's adaptability. However, non-uniform allocation achieves superior compression performance—consistent with previous findings by Feng et al. [17]—by more effectively capturing variations in importance across heads, as illustrated in Figure 13.

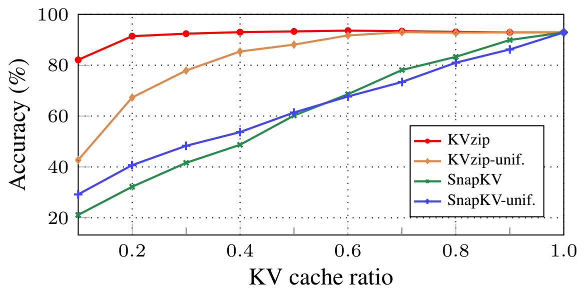

Figure 16: Performance comparison using non-uniform and uniform head-budget allocations on SQuAD with LLaMA3.1-8B. *Unif.* refers to the uniform allocation.

D Individual Dataset Performance

Model Scale and Architecture. Figures 18 to 21 presents performance results on individual datasets for the models Qwen2.5-14B-1M [54], LLaMA3.1-8B [21], Gemma3-12B [49], and LLaMA3-8B-W8A8KV4 [36].

For the Gemma model, Retr.KV and Retr.Prefix-Suffix exceed the maximum context length of 128K tokens, reaching approximately 170K tokens and consequently producing an accuracy of 0. Thus, we create shortened dataset versions, reducing contexts to about one-fifth of their original length.

Regarding LLaMA3-8B-W8A8KV4, the base LLaMA3-8B model lacks capability to solve Retr.KV, Retr.Prefix-Suffix, and Math.Find tasks, resulting in near-zero accuracy. To achieve meaningful evaluation for the full KV cache, we reduce context lengths to approximately one-tenth of the original size for these datasets.

Multi-Task Datasets. Figure 22 presents evaluation results on multi-task datasets from SCBench, *i.e.*, Mix.Sum+NIAH and Mix.RepoQA+KV, each composed of two distinct tasks [35]. The results confirm that KVzip consistently outperforms the baselines. Figure 23 presents results for LLaMA3.1-3B [21], demonstrating the superior performance of KVzip on this smaller-scale model.

RULER Benchmark. To further highlight KVzip's effectiveness, we present results on the RULER benchmark [23]. These results are publicly available by the NVIDIA KVPress repository[4]. Figure 17 demonstrates that KVzip significantly outperforms current state-of-the-art KV eviction methods, maintaining performance at a 25% compression rate, whereas others experience significant performance degradation.

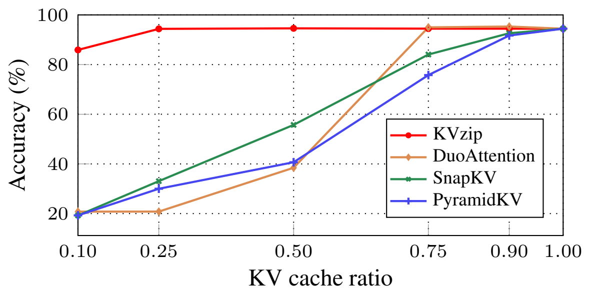

Figure 17: Average performance on the RULER benchmark using Qwen3-8B.

[4]https://huggingface.co/spaces/nvidia/kvpress-leaderboard

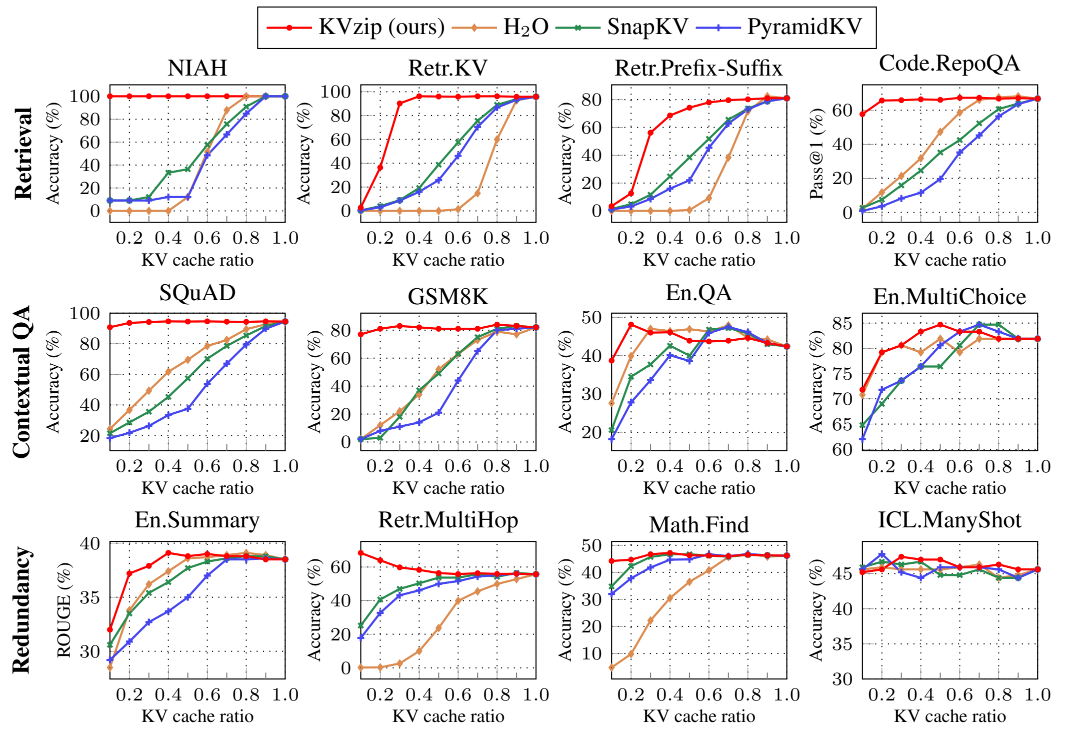

Figure 18: Benchmark results using Qwen2.5-14B-1M [54] across compression ratios from 0.1 to 1.0.

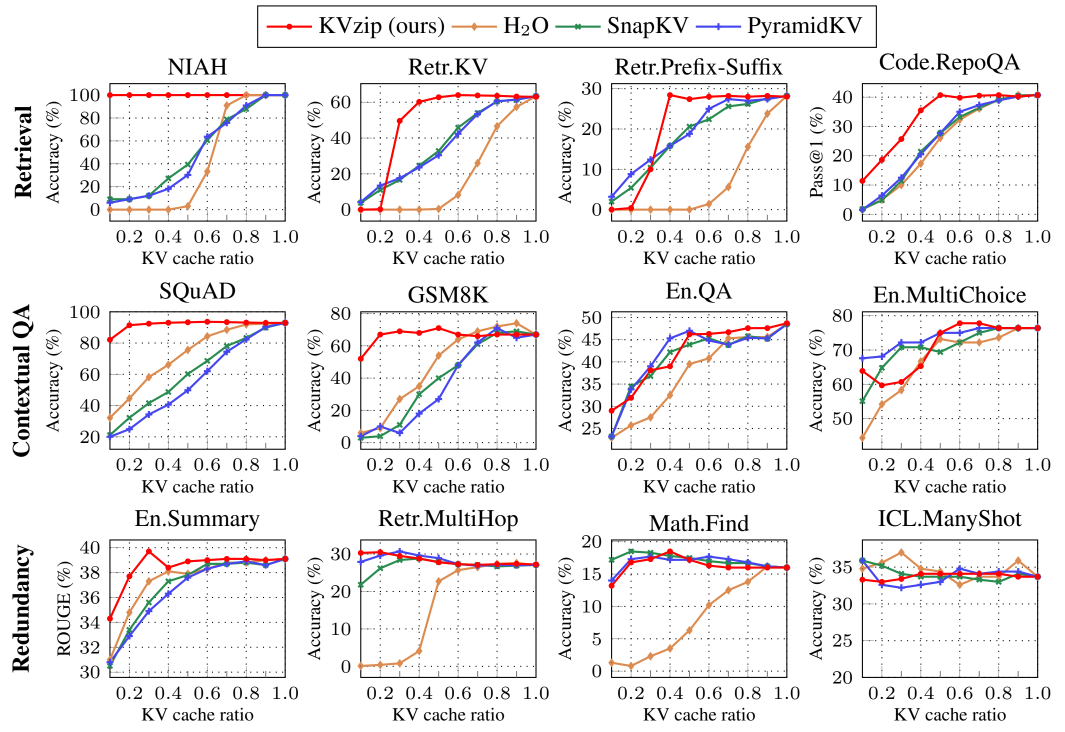

Figure 19: Benchmark results using LLaMA3.1-8B [21] across compression ratios from 0.1 to 1.0.

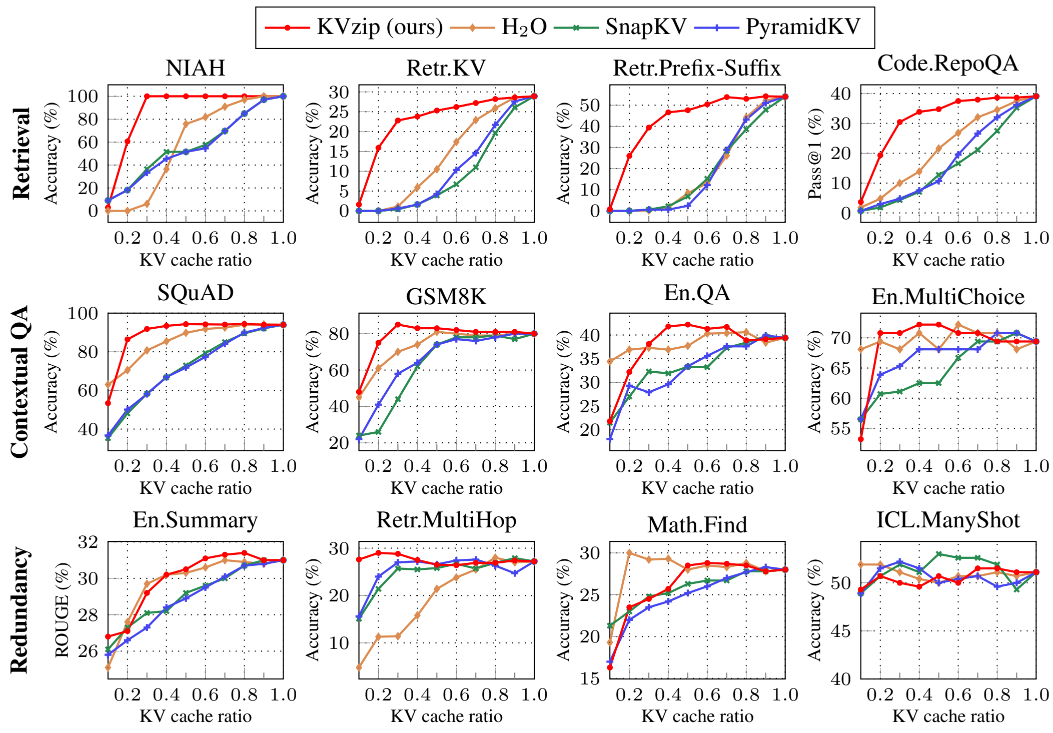

Figure 20: Benchmark results using Gemma3-12B [49] across compression ratios from 0.1 to 1.0.

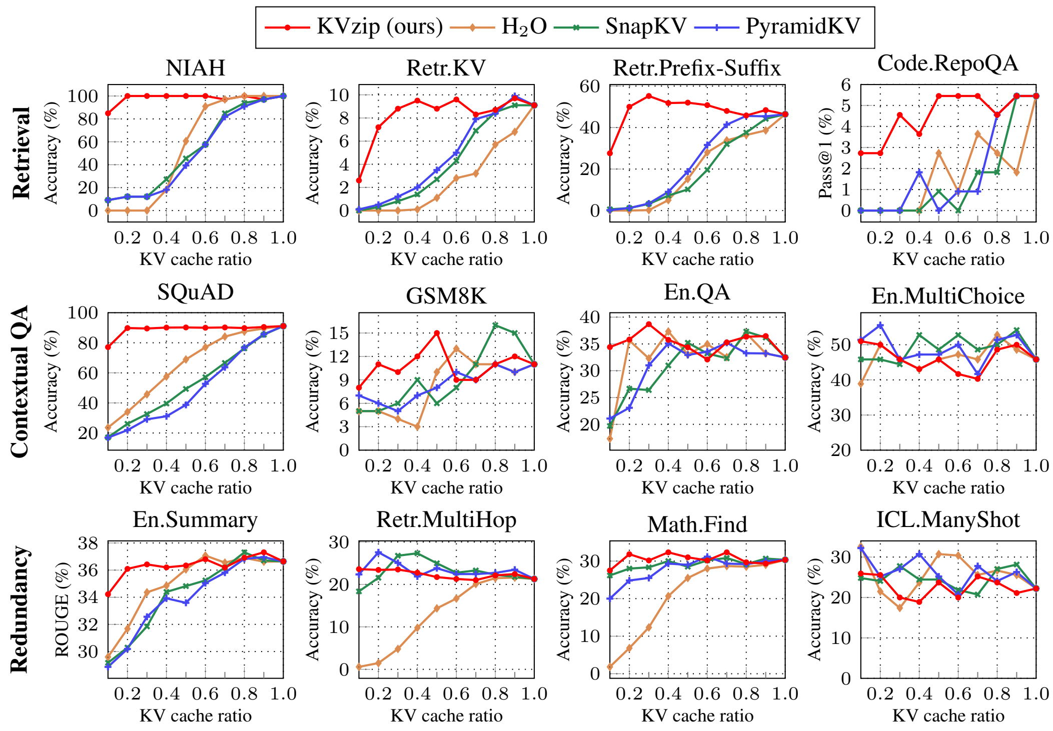

Figure 21: Benchmark results using LLaMA3-8B-W8A8KV4 [36] across compression ratios from 0.1 to 1.0.

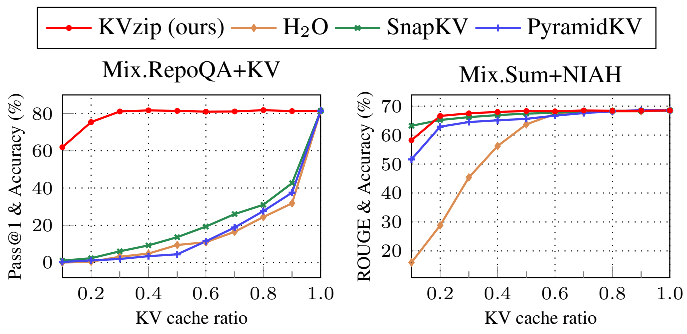

Figure 22: Benchmark results on SCBench multi-task datasets using Qwen2.5-7B-1M [54] across compression ratios from 0.1 to 1.0.

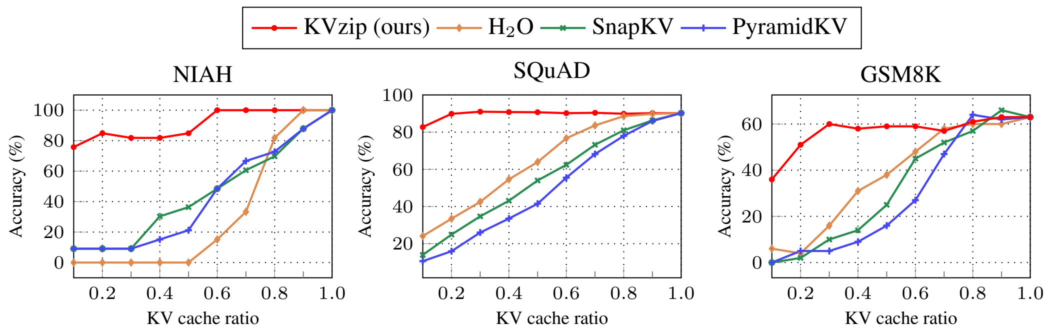

Figure 23: Benchmark results for LLaMA3.1-3B [21] across compression ratios ranging from 0.1 to 1.0. The evaluation focuses on shorter contexts, as LLaMA3.1-3B lacks the capability to solve SCBench tasks, resulting in near-zero accuracy.

Table 3: Inputs for KV cache importance scoring from a SQuAD example (used in the visualizations in Figure 6 and Figure 13). The context is included in the input of the repeat task.

Task	Inputs for scoring (prompt+response)
Repeat	Repeat the previous context: Architecturally, the school has a Catholic character. Atop the Main Buildingś gold dome is a golden statue of the Virgin Mary. Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend "Venite Ad Me Omnes". Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto, a Marian place of prayer and reflection. It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858. At the end of the main drive (and in a direct line that connects through 3 statues and the Gold Dome), is a simple, modern stone statue of Mary.
QA-1	Q: To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France? A: Saint Bernadette Soubirous
QA-2	Q: What is in front of the Notre Dame Main Building? A: a copper statue of Christ
Summarization	Please summarize the previous context: The school has a Catholic character with various statues and buildings that reflect this theme, including a gold dome with a Virgin Mary statue, a Christ statue, a Basilica of the Sacred Heart, a Grotto replica of Lourdes, and a modern stone Mary statue at the end of the main drive.

Table 4: Inputs for importance scoring from a GSM8K example used in the visualization in Figure 6, a reasoning task. The context is included in the input of the repeat task.

Task	Inputs for scoring (prompt+response)
Repeat	Repeat the previous context: Janet’s ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers’ market daily for \$2 per fresh duck egg.
Reasoning	Reason and answer the question. Q: How much in dollars does she make every day at the farmers’ market? Janet’s ducks lay 16 eggs per day. She eats 3 eggs for breakfast, so she has 16 - 3 = 13 eggs left. She bakes 4 eggs for muffins, so she has 13 - 4 = 9 eggs left. She sells the remaining 9 eggs at the farmers’ market for \$2 each. To find out how much she makes, we multiply the number of eggs she sells (9) by the price per egg (\$2): \$9 x \$2 = \$18. The answer is \$18.

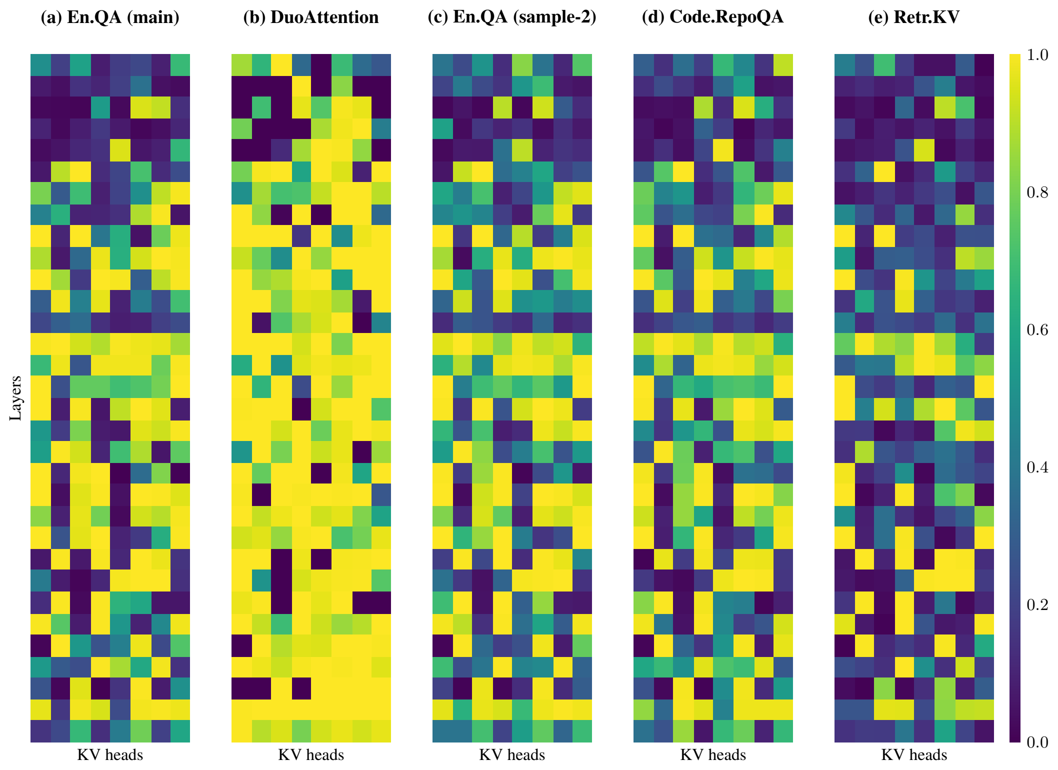

Figure 24: **Visualization of head-level importance scores** for context-independent compression in Section 4.2. We use the head scores obtained from an En.QA example in our primary experiments (Figure 11). For reference, (c)-(e) show head scores derived from alternative data sources from SCBench [35]. Our scoring method yields a more uniformly distributed importance pattern compared to DuoAttention. We select the En.QA sample for our main experiments due to its comprehensive overlap with importance patterns from other data sources, whereas Retr.KV, composed of synthetic passkeys, exhibits sparser importance patterns.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: In the abstract and introduction, we clearly state the paper's contribution and scope: query-agnostic KV cache eviction for Transformer-based LLMs (Section 1, L32-61).

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss limitations in Appendix B

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [NA]

Justification: Our paper does not include theoretical results.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide detailed experimental setups including evaluation metric and implementation details in Section 4.1 and Appendix A.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: We provide pseudo code for our algorithm in Algorithm 1. We use existing benchmark datasets publicly available. We attach implementation codes in the supplementary materials. We will open-source our codes upon acceptance.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We provide evaluation details in Section 4.1 and Appendix A. We provide hyperparameters in Section 3.4.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: Our experiments utilize established benchmarks, including SCBench [35], which introduces no uncertainty during evaluation (Section 4.1, L212–220). Our evaluation employs deterministic greedy decoding (Section 4.1, L225) without involving any training processes that could introduce stochasticity. We follow the experimental reporting formats of prior works, including H2O, SnapKV, and DuoAttention [60, 33, 53]. We conduct comprehensive evaluations across 12 datasets and 5 models to demonstrate the empirical significance of our method.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We analyze computational complexity in Section 3.4. We describe computer resources in Section 4.1.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: We have reviewed the NeurIPS Code of Ethics and conduct research adhering to the statement.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: We discuss broader impacts in Appendix B

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: Our paper does not release new generative models or data.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We cite related papers in Section 5. We provided links for baseline implementation codes in Appendix A.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: We do not release new assets.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

    Answer: [NA]

    Justification: We do not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (https://neurips.cc/Conferences/2025/LLM) for what should or should not be described.