

000 001 002 003 004 005 HOLD ONTO THAT THOUGHT: ASSESSING KV CACHE 006 COMPRESSION ON REASONING 007 008 009 010

011 **Anonymous authors**
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027

028 Paper under double-blind review
029
030

031 ABSTRACT

032 Large language models (LLMs) have demonstrated remarkable performance on
033 long-context tasks, but are often bottlenecked by memory constraints. Namely,
034 the KV cache, which is used to significantly speed up attention computations,
035 grows linearly with context length. A suite of compression algorithms has been
036 introduced to alleviate cache growth by evicting unimportant tokens. However,
037 several popular strategies are targeted towards the prefill phase, i.e., processing
038 long prompt context, and their performance is rarely assessed on reasoning tasks
039 requiring long decoding. In particular, short but complex prompts, such as those in
040 benchmarks like GSM8K and MATH500, often benefit from multi-step reasoning
041 and self-reflection, resulting in thinking sequences thousands of tokens long. In this
042 work, we benchmark the performance of several popular compression strategies on
043 long-reasoning tasks. For the non-reasoning Llama-3.1-8B-Instruct, we determine
044 that no singular strategy fits all, and that performance is heavily influenced by
045 dataset type. However, we discover that H2O and our decoding-enabled variant
046 of SnapKV are dominant strategies for reasoning models, indicating the utility of
047 heavy-hitter tracking for reasoning traces. We also find that eviction strategies
048 at low budgets can produce longer reasoning traces, revealing a tradeoff between
049 cache size and inference costs.

050 1 INTRODUCTION

051 Large language models (LLMs) have demonstrated remarkable performance on complex NLP tasks
052 that require multi-step reasoning. Unlike summarization tasks (Bai et al., 2023; Fabbri et al., 2021)
053 and keyword tracking tasks (Hsieh et al., 2024), which scale task complexity with context length,
054 reasoning benchmarks challenge models to generate answers that are not clearly contained in the
055 prompt. Such tasks include reading comprehension (Dua et al., 2019; Yu et al., 2020), commonsense
056 reasoning (Zellers et al., 2019; Talmor et al., 2018; Geva et al., 2021), first-order logic (Han et al.,
057 2022; Kwon et al., 2025), and mathematical problem-solving (Cobbe et al., 2021).

058 Reasoning benchmarks differ from long-context tasks in that they normally compel the LLM to
059 provide answers that are longer than the question itself. This can pose a serious resource problem
060 for the LLM, as past token key and value embeddings are maintained in memory to avoid redundant
061 attention calculations. This key-value (KV) cache grows linearly with sequence length, which
062 can result in memory blowup for older or single-GPU setups. Furthermore, specialized reasoning
063 models such as DeepSeek-R1 (Guo et al., 2025) and the Llama-Nemotron series are known to output
064 excessively long reasoning traces (Cai et al., 2025; Fatemi et al., 2025) which outnumber the length
065 of the prompt itself by hundreds to thousands of tokens.

066 To address the memory demands of long sequences, numerous KV cache compression methods have
067 been proposed. These techniques generally maintain a fixed KV cache size by selectively discarding
068 tokens deemed “unimportant”. However, defining token importance” is non-trivial, and different
069 approaches rely on distinct heuristics: attention scores (Zhang et al., 2023; Liu et al., 2023; Li et al.,
070 2024), cosine similarity (Liu et al., 2024a; Han et al., 2023), embedding norms (Devoto et al., 2024),
071 and head-specific token-type preferences (Ge et al., 2023). Despite this variety, most evaluations of
072 cache compression have focused on long-context benchmarks such as LongBench (Bai et al., 2023)
073 and RULER (Hsieh et al., 2024), or on heterogeneous batteries like LM Eval Harness ((Gao et al.,
074 2024)), rather than tasks where the generation length, not the prompt, dominates memory usage.

In this work, we conduct a comprehensive assessment of the major state-of-the-art KV cache compression strategies across eight reasoning benchmarks: FOLIO (Han et al., 2022), DROP (Dua et al., 2019), GSM8K (Cobbe et al., 2021), MATH-500 (Lightman et al., 2023), ReClor (Yu et al., 2020), StrategyQA (Geva et al., 2021), CommonSenseQA (Talmor et al., 2018), and OpenBookQA (Mihaylov et al., 2018). Together, these benchmarks span four critical reasoning categories: reading comprehension, common sense, logical reasoning, and mathematical reasoning. We evaluate these strategies on Llama-3.1-8B-Instruct as well as four reasoning models: Llama-3.1-Nemotron-Nano-8B-v1, DeepSeek-R1-Distill-Llama-8B, and DeepSeek-R1-Distill-Qwen-7B/14B. By focusing on long-generation rather than long-prompt scenarios, our study fills a notable gap in the existing literature. Our primary contributions are threefold:

A comprehensive benchmark: We conduct a comprehensive evaluation of major KV cache compression strategies, including StreamingLLM, H2O (Zhang et al., 2023), a decoding-enabled SnapKV (Li et al., 2024), R-KV (Cai et al., 2025), and KNorm (Devoto et al., 2024), across a suite of eight benchmarks spanning mathematical, logical, and commonsense reasoning. We evaluate over several realistic settings, cache, and max token budgets for a single-GPU system.

Renewed attention for attention-based compression: Our analysis reveals that classical attention-based “heavy-hitter” strategies, which evict tokens based on accumulated attention scores, significantly outperform other methods, even defeating full-cache reasoning occasionally. Namely, this includes H2O and our novel and simple extension of SnapKV (prompt-only compression method) to a decoding-enabled variant, SnapKV-Decoding. Both methods, especially SnapKV-D, win over *all* budgets and datasets for reasoning models.

A library for analyzing decoding compression: We implement a fork of the NVIDIA kvpress¹ library, which adds support for decoding phase compression for any kvpress method. We add support for R-KV and H2O to the kvpress. Our goal is to provide an open-source playground for analyzing end-to-end KV cache compression strategies.

2 PRELIMINARIES

In this section, we briefly review the concepts of large language models, LLM inference and autoregressive generation, the KV cache, and the chain-of-thought (CoT) reasoning.

Transformer Architectures and Autoregressive Generation. Modern Large Language Models (LLMs) predominantly operate as autoregressive, decoder-only Transformers (Vaswani et al., 2017; Radford et al., 2019; Achiam et al., 2023; Touvron et al., 2023). This architecture generates text sequentially, producing one token at a time by conditioning on the entire preceding sequence of tokens, which includes both the initial prompt and any previously generated output (Brown et al., 2020). Importantly, the model’s ability to maintain coherent and contextually relevant generation over time is crucial to its capabilities, especially in tasks requiring reasoning or narrative development (Lee et al., 2024; Zhang et al., 2025).

Self-Attention Mechanism and the KV Cache Bottleneck. During generation, a query (q) vector for the current token is *attends* to a series of Key (k) and Value (v) vectors corresponding to every token in the preceding context. In this process, notably, for the generation of every new token, the entire sequence of Key and Value vectors for *all* previous tokens should be accessed. To avoid recomputing these K-V pairs at each step, they are stored in the Key-Value (KV) cache, the size of which grows linearly with the sequence length (n), resulting in an $O(n)$ memory complexity that creates a significant bottleneck. Formally, for a sequence of n tokens, we denote the query cache $Q_l^h \in \mathbb{R}^{n \times d}$, key cache $K_l^h \in \mathbb{R}^{n \times d}$, and value cache $V_l^h \in \mathbb{R}^{n \times d}$, where d is the embedding dimension, l is the layer, and h denotes a head for multi-head attention layers (Vaswani et al., 2017). The dot-product self-attention mechanism is defined as $A_l^h(Q_l^h, K_l^h, V_l^h) = \text{softmax}(Q_l^h(K_l^h)^\top / \sqrt{d})V_l^h$. To avoid linear scaling with sequence length, *token eviction* methods, the key focus of work, discard embeddings of previous tokens which are no longer “important” to the current decoding step.

¹<https://github.com/NVIDIA/kvpress>

108 **Objective.** A common objective to study the quality of an importance heuristic is to minimize
 109 the deviation between the outputs of a non-evicted and evicted attention layer. More specifically,
 110 let \bar{K}_l^h and \bar{V}_l^h , respectively, denote an evicted key and value cache. We may interpret these
 111 caches as sparse matrices by dropping all but B rows (the budget) of K_l^h and V_l^h . Attention is
 112 typically followed by multiplication with an output projector $W_O \in \mathbb{R}^{d \times p}$ and passage through
 113 a 2-layer MLP $\mathcal{F}(x) := x + W_2 \text{relu}(W_1 x)$, where W_1, W_2 are trained hidden weights. Let $x =$
 114 $A_l^h(Q_l^h, K_l^h, V_l^h)$ and $\bar{x} = A_l^h(Q_l^h, \bar{K}_l^h, \bar{V}_l^h)$. The objective of any KV eviction algorithm is to
 115 minimize $E[||\mathcal{F}(x) - \mathcal{F}(\bar{x})||_2]$, where randomness is with regards to the attention distribution.
 116 Guarantees on this error are scarce in the eviction landscape, with the most prominent presented for
 117 heavy-hitter Scissorhands approach in Liu et al. (2023), which asserts an upper bound that scales
 118 with $1 - B/N$, where N is the sequence length and assumes a heavy-tailed distribution of attention
 119 scores, which is frequently observed (Devoto et al., 2024; Liu et al., 2024a; 2023). However, many
 120 popular eviction strategies, such as StreamingLLM, KNorm, and PyramidKV are based on empirical
 121 observations as opposed to attention-tracking and thus are currently not guaranteed.
 122

123 **Chain-of-Thought and Multi-Step Reasoning.** While many long-context applications involve
 124 processing long prompts, a critical class of tasks requires long-form generation from short and
 125 complex prompts. Prompting strategies such as Chain-of-Thought (CoT) encourage models to
 126 externalize their reasoning process, generating intermediate “thinking” steps that can extend for
 127 hundreds or thousands of tokens to solve a problem (Wei et al., 2022; Wang et al., 2022). Benchmarks
 128 such as GSM8K (Cobbe et al., 2021) are representative of this domain, where the path to the correct
 129 answer necessitates a lengthy, self-generated chain of reasoning.
 130

3 RELATED WORK

3.1 KV CACHE COMPRESSION

134 KV cache compression is a rich field of study composed of strategies ranging from quantization
 135 (Hooper et al., 2024; Ashkboos et al., 2024; Liu et al., 2024b) to offloading methods that move the
 136 entire cache to the CPU which is significantly less memory bound (Sun et al., 2024; Chen et al., 2024;
 137 Tang et al., 2024). However, in this work, we are focused on strategies which maintain a constant
 138 cache size, thus permitting arbitrary generation length.
 139

3.1.1 TOKEN EVICTION



154 Figure 1: A Conceptual Comparison of Token Retention Strategies in Different KV Cache Compre-
 155 hension Methods. Each row illustrates a method’s logic for retaining tokens (colored) versus evicting
 156 them (gray) from the KV cache during a long sequence divided into a prefill and decoding phase.
 157

158 A primary line of research for mitigating the memory burden of the KV cache involves *token eviction*.
 159 These methods aim to reduce the cache size by selectively removing or merging tokens deemed
 160 less important. To achieve this, multiple approaches have been developed, including recency-based
 161 approaches such as simple sliding window (Beltagy et al., 2020), importance-based methods that
 retain “attention sinks” or heavy-hitter tokens from the prompt (Xiao et al., 2023; Zhang et al.,
 162

162 2023; Li et al., 2024; Liu et al., 2023), dynamically adjustment of KV caches per layer for optimal
 163 efficiency-utility balancing (Cai et al., 2024), redundancy-aware techniques that merge semantically
 164 similar states (Cai et al., 2025). Figure 1 provides a conceptual comparison of the most important
 165 approaches we cover in this work.

166 **StreamingLLM**’s (Xiao et al., 2023) strategy is to always maintain the KV states of the first few (e.g.,
 167 four) tokens, which serve as the attention sinks, and combine them with a sliding window of the most
 168 recent tokens (up to the available budget). **H2O** (Zhang et al., 2023) dynamically identifies important
 169 or “heavy hitter” tokens based on their cumulative attention scores received during generation. The
 170 H2O cache is composed of two parts: a budget for the most recent tokens and a budget for the H2
 171 tokens. **SnapKV** (Li et al., 2024) focuses primarily on compressing the KV cache of the initial
 172 prompt during the prefill stage. SnapKV uses a small “observation window” at the end of the prompt
 173 to predict importance. The attention scores from queries in this observation window are aggregated
 174 to “vote” for important positions (heavy hitters) in the prefix. **R-KV** (Cai et al., 2025), designed for
 175 reasoning trace compression, uses a combination of accumulated attention score and pair-wise key
 176 cosine similarities to identify unimportant tokens. A distinct and computationally efficient approach,
 177 which we refer to as the **KNorm** strategy (Devoto et al., 2024), bypasses the need for attention
 178 scores entirely. Specifically, the authors observe that tokens whose key vectors have a low L_2 norm
 179 consistently attract high attention scores from subsequent queries.

180 3.2 BENCHMARKING REASONING

182 **GSM8K** (Grade School Math 8K) is a widely-used dataset of grade-school level math word problems
 183 (Cobbe et al., 2021). More advanced challenges are drawn from the **MATH-500** dataset (Lightman
 184 et al., 2023), which contains competition-level problems across algebra, geometry, and number
 185 theory. **ReClor** (Yu et al., 2020) is a reading comprehension dataset built from GMAT and LSAT
 186 logical reasoning questions. Similarly, **LogiQA** (Liu et al., 2020) provides multiple-choice questions
 187 from civil service exams that require a deep understanding of logical puzzles and deductions. For
 188 evaluating capabilities in more formal systems, the **FOLIO** (Han et al., 2022) dataset assesses natural
 189 language reasoning in the context of First-Order Logic (FOL). Beyond formal and mathematical
 190 logic, a significant portion of research focuses on commonsense reasoning. **StrategyQA** (Geva et al.,
 191 2021) tests a model’s ability to infer the implicit reasoning steps needed to answer a yes/no question
 192 by asking for the underlying strategy. Another tested benchmarks is **CommonsenseQA**, which
 193 tests a model’s ability to reason with general world knowledge. Finally, the integration of textual
 194 understanding with quantitative skills is measured by benchmarks such as **DROP** (Dua et al., 2019).
 195 This reading comprehension dataset is unique in that answering its questions requires performing
 196 discrete operations like counting, sorting, or simple arithmetic directly on the information presented.

197 4 EXPERIMENTS & ANALYSIS

198 4.1 SETUP

201 *KV Compression Methods.* We test **H2O**, **R-KV**, **StreamingLLM**, **KNorm**, our own decoding-
 202 variant of SnapKV which we call **SnapKV-D**, and **ShadowKV** (Sun et al., 2024). We note that
 203 ShadowKV uses the CPU to offload the cache and thus is not a true compression strategy. However,
 204 offloading strategies represent an important class of compression methods; thus, we include them as a
 205 baseline. For SnapKV-D, we now allow the observation window to slide along the decoded sequence
 206 at regular intervals (every window size $w = 128$ steps). This is detailed further in Appendix 10.
 207 Further hyperparameter details are described in Appendix A.11.

208 *Models.* We test the base, non-reasoning Llama-3.1-8B-Instruct and three reasoning models:
 209 DeepSeek-R1-Distill-Qwen-7B/14B, Nemotron-Nano-8B-v1 and DeepSeek-R1-Distill-Llama-8B.

210 *Datasets.* We divide our benchmark into 4 distinct groups: (1) **Reading Comprehension:** DROP,
 211 ReClor; (2) **Logical Reasoning:** StrategyQA, FOLIO; (3) **Commonsense Reasoning:** OpenBookQA
 212 (OBQA), CommonsenseQA (CSQA); (4) **Math Reasoning:** MATH-500, GSM8K. For each dataset,
 213 we randomly sample 100 questions for two different seeds.

214 *Performance.* For benchmarking the individual compression strategies, we use the NVIDIA kvpress
 215 library, which natively provides most of the targeted algorithms. We provide each dataset to each

216 Table 1: **Llama-3.1-8B-Instruct.** Varying compressions and budgets on a *non-reasoning* model.
217

Llama-3.1-8B-Instruct	GSM8K				Math500				CSQA				OBQA			
	128	256	384	512	128	256	384	512	128	256	384	512	128	256	384	512
Full	0.88				0.39				0.77				0.84			
ShadowKV	0.32				0.22				0.20				0.31			
H2O	0.63	0.77	0.82	0.83	0.30	0.33	0.33	0.36	0.74	0.76	0.77	0.77	0.83	0.86	0.86	0.86
Knorm	0.05	0.53	0.73	0.82	0.03	0.18	0.22	0.33	0.34	0.77	0.75	0.76	0.41	0.79	0.84	0.82
RKV	0.12	0.34	0.50	0.49	0.03	0.10	0.16	0.20	0.36	0.62	0.76	0.77	0.22	0.66	0.77	0.84
SnapKV	0.53	0.55	0.56	0.53	0.20	0.21	0.19	0.20	0.70	0.64	0.71	0.72	0.73	0.77	0.72	0.76
StreamingLLM	0.26	0.75	0.84	0.87	0.11	0.26	0.32	0.35	0.20	0.75	0.76	0.77	0.14	0.72	0.84	0.84
	ReClor				DROP				StrategyQA				FOLIO			
Full	0.60				0.15				0.83				0.51			
ShadowKV	0.27				0.28				0.68				0.33			
H2O	0.32	0.56	0.60	0.58	0.12	0.14	0.17	0.17	0.81	0.87	0.88	0.89	0.22	0.43	0.41	0.43
Knorm	0.01	0.19	0.46	0.59	0.01	0.08	0.13	0.13	0.47	0.85	0.88	0.87	0.02	0.28	0.39	0.38
RKV	0.04	0.21	0.40	0.54	0.06	0.07	0.14	0.11	0.60	0.79	0.77	0.79	0.07	0.36	0.44	0.34
SnapKV	0.53	0.57	0.58	0.55	0.15	0.12	0.11	0.12	0.78	0.78	0.81	0.76	0.44	0.40	0.45	0.46
StreamingLLM	0.05	0.21	0.59	0.58	0.09	0.11	0.15	0.16	0.11	0.76	0.89	0.85	0.03	0.09	0.25	0.35

235
236 Table 2: **Deepseek-R1-Distill-Qwen-7B.** Performance using varying compressions and budgets.
237

Deepseek-R1-Distill-Qwen-7B	GSM8K				Math500				CSQA				OBQA			
	128	256	384	512	128	256	384	512	128	256	384	512	128	256	384	512
Full	0.70				0.47				0.67				0.78			
ShadowKV	0.47				0.33				0.20				0.31			
H2O	0.21	0.44	0.51	0.52	0.14	0.21	0.29	0.31	0.44	0.61	0.60	0.64	0.42	0.64	0.69	0.67
Knorm	0.00	0.00	0.08	0.16	0.00	0.01	0.03	0.05	0.05	0.13	0.30	0.42	0.03	0.05	0.23	0.38
RKV	0.04	0.07	0.18	0.30	0.04	0.04	0.05	0.17	0.10	0.09	0.27	0.34	0.10	0.10	0.21	0.26
SnapKV	0.67	0.67	0.70	0.71	0.38	0.36	0.36	0.32	0.65	0.62	0.59	0.61	0.71	0.75	0.68	0.76
StreamingLLM	0.02	0.19	0.32	0.44	0.03	0.12	0.19	0.26	0.08	0.14	0.31	0.48	0.02	0.11	0.28	0.37
	ReClor				DROP				StrategyQA				FOLIO			
Full	0.45				0.16				0.67				0.36			
ShadowKV	0.27				0.14				0.60				0.33			
H2O	0.01	0.04	0.18	0.28	0.04	0.07	0.10	0.10	0.33	0.64	0.74	0.72	0.03	0.21	0.23	0.23
Knorm	0.00	0.00	0.01	0.01	0.00	0.01	0.01	0.03	0.00	0.12	0.44	0.59	0.00	0.01	0.03	0.05
RKV	0.04	0.03	0.02	0.01	0.04	0.04	0.03	0.04	0.05	0.14	0.34	0.42	0.04	0.03	0.02	0.06
SnapKV	0.45	0.39	0.40	0.43	0.13	0.11	0.12	0.16	0.60	0.59	0.57	0.63	0.30	0.25	0.31	0.29
StreamingLLM	0.00	0.01	0.01	0.04	0.04	0.05	0.08	0.13	0.00	0.05	0.22	0.42	0.00	0.01	0.02	0.03

258 model over the cache budgets $\{128, 256, 384, 512\}$. Each model is allowed to generate a maximum
259 of 2048 new tokens via greedy decoding. This token limit is enforced to better simulate a resource-
260 constrained setting for inference and also based on mean generation lengths reported in Table 7.. We
261 use author-recommended hyperparameters for all methods. Accuracy benchmarks were performed
262 on an HPC cluster using an NVIDIA RTX A6000 48GB GPU.

263 4.2 LATENCY EXPERIMENT

264 Although this benchmark is primarily concerned with accuracy, we assess the latency of our tested
265 methods in Figure 2 and Table 5 to gather a more complete picture of efficiency. **StreamingLLM** and
266 **KNorm** do not compute accumulated attention scores thus they incur less overhead than **H2O** and
267 **SnapKV-D**. These results concurs with the computational overhead summarized in Table 13.

270 Table 3: **Nemotron-Nano-8B.** Performance using varying compressions and budgets.
271

272 Nemotron- 273 Nano-8B	274 GSM8K				275 Math500				276 CSQA				277 OBQA			
	128	256	384	512	128	256	384	512	128	256	384	512	128	256	384	512
Full	0.64				0.45				0.51				0.64			
ShadowKV	0.44				0.28				0.20				0.31			
H2O	0.22	0.45	0.52	0.57	0.16	0.24	0.31	0.33	0.47	0.49	0.52	0.51	0.59	0.59	0.58	0.62
Knorm	0.01	0.02	0.09	0.18	0.01	0.01	0.03	0.06	0.36	0.40	0.44	0.46	0.32	0.44	0.48	0.57
RKV	0.04	0.03	0.09	0.15	0.02	0.04	0.03	0.06	0.28	0.30	0.42	0.41	0.35	0.44	0.51	0.51
SnapKV	0.65	0.63	0.66	0.66	0.41	0.44	0.45	0.43	0.49	0.50	0.51	0.53	0.68	0.63	0.66	0.66
StreamingLLM	0.03	0.20	0.40	0.53	0.02	0.13	0.22	0.34	0.36	0.44	0.46	0.50	0.36	0.46	0.52	0.62
	ReClor				DROP				StrategyQA				FOLIO			
	128	256	384	512	128	256	384	512	128	256	384	512	128	256	384	512
Full	0.48				0.11				0.89				0.36			
ShadowKV	0.27				0.11				0.65				0.33			
H2O	0.20	0.22	0.35	0.40	0.05	0.06	0.10	0.09	0.76	0.84	0.85	0.83	0.22	0.36	0.35	0.37
Knorm	0.01	0.03	0.07	0.07	0.01	0.01	0.02	0.03	0.38	0.55	0.68	0.76	0.03	0.04	0.07	0.13
RKV	0.03	0.08	0.08	0.07	0.02	0.06	0.05	0.03	0.42	0.45	0.64	0.71	0.06	0.08	0.11	0.14
SnapKV	0.42	0.42	0.42	0.37	0.11	0.11	0.12	0.10	0.83	0.85	0.84	0.84	0.38	0.42	0.41	0.41
StreamingLLM	0.03	0.06	0.09	0.14	0.03	0.02	0.06	0.08	0.24	0.39	0.52	0.69	0.03	0.03	0.06	0.15

289
290 Table 4: **DeepSeek-R1-Distill-Llama-8B.** Performance using varying compressions and budgets.
291

292 DeepSeek-R1- 293 Distill-Llama-8B	294 GSM8K				295 Math500				296 CSQA				297 OBQA			
	128	256	384	512	128	256	384	512	128	256	384	512	128	256	384	512
Full	0.70				0.46				0.75				0.84			
ShadowKV	0.51				0.34				0.20				0.31			
H2O	0.37	0.53	0.62	0.61	0.20	0.31	0.36	0.36	0.48	0.72	0.73	0.73	0.48	0.78	0.83	0.84
Knorm	0.00	0.09	0.19	0.28	0.00	0.01	0.02	0.06	0.05	0.28	0.54	0.66	0.03	0.27	0.57	0.70
RKV	0.05	0.04	0.14	0.17	0.03	0.05	0.02	0.02	0.07	0.11	0.16	0.35	0.07	0.07	0.19	0.32
SnapKV	0.72	0.72	0.74	0.72	0.42	0.44	0.41	0.41	0.74	0.73	0.74	0.73	0.82	0.83	0.83	0.81
StreamingLLM	0.06	0.25	0.39	0.56	0.03	0.09	0.21	0.29	0.04	0.14	0.35	0.50	0.07	0.15	0.32	0.52
	ReClor				DROP				StrategyQA				FOLIO			
	128	256	384	512	128	256	384	512	128	256	384	512	128	256	384	512
Full	0.51				0.14				0.74				0.47			
ShadowKV	0.27				0.09				0.80				0.33			
H2O	0.03	0.08	0.23	0.38	0.06	0.07	0.10	0.11	0.25	0.69	0.77	0.79	0.07	0.37	0.41	0.46
Knorm	0.00	0.00	0.02	0.10	0.00	0.01	0.01	0.05	0.06	0.36	0.57	0.70	0.00	0.03	0.11	0.21
RKV	0.04	0.03	0.03	0.11	0.03	0.03	0.05	0.07	0.08	0.35	0.50	0.63	0.03	0.08	0.13	0.26
SnapKV	0.52	0.53	0.56	0.51	0.17	0.15	0.15	0.16	0.68	0.66	0.64	0.68	0.46	0.45	0.49	0.46
StreamingLLM	0.00	0.00	0.01	0.06	0.02	0.02	0.09	0.13	0.03	0.11	0.36	0.56	0.00	0.01	0.04	0.09

309
310 4.3 MAX TOKEN ABLATION

312 As explained in setup, we chose a max token length of 2048 both because we find that the mean token
313 length over datasets is under this budget and to better assess performance in a compute-bound setting.
314 However, we study the effect of max token limit on performance under a fixed budget of 1024 for
315 MATH500 for R1-Distill-Qwen7B in Figure 7. We find that performance improves significantly for
316 all methods initially, but then SnapKV-D overtakes all methods for all other max token limits.
317

318 4.4 LARGE MODEL COMPARISON
319

320 We determine whether our observed trends hold for a larger reasoning model, R1-Distill-Qwen-14B
321 in Table 6. We examine the performance of all methods on the more challenging GSM8K and
322 MATH500. Unsurprisingly, base accuracies do improve, but more importantly, we observe that again,
323 the heavy-hitter methods H2O and SnapKV-D outperform their competitors by a significant margin
indicating that larger reasoning models still benefit from attention-based eviction.

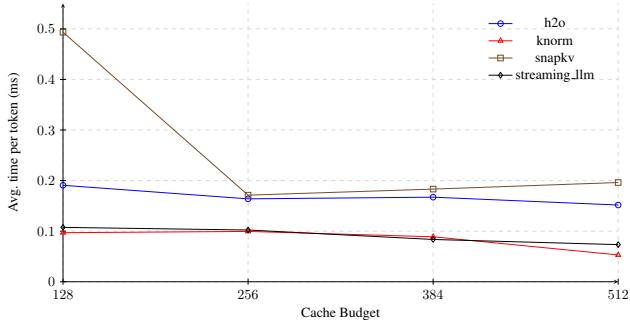


Figure 2: **Latency vs Budget.** Average generation time per token (ms) versus KV-cache budget for eviction strategies. KNorm and StreamingLLM speed up markedly with larger budgets, H2O improves more modestly, while SnapKV-D is slow at small budgets.

Table 5: **End-to-end token throughput (tokens/s) on GSM8K.** 50 sample average is reported.

Method / Budget	128	256	384	512
Full	30.71	30.71	30.71	30.71
H2O	25.81	23.41	23.93	25.16
Knorm	27.38	28.07	28.35	27.56
R-KV	29.10	27.72	28.55	30.04
StreamingLLM	27.04	27.55	27.12	29.39
SnapKV	27.46	25.57	26.24	27.60

Table 6: **R1-Distill-Qwen14B.** Cache budgets = [128, 256, 384, 512]. We examine the performance of various compression methods for a larger reasoning model. Winner per budget in bold.

Method	GSM8K				MATH500			
	128	256	384	512	128	256	384	512
full	0.81				0.47			
shadowkv	0.53				0.38			
h2o	0.33	0.56	0.62	0.64	0.20	0.27	0.31	0.31
knorm	0	0.02	0.08	0.21	0	0	0	0.02
rkv	0.02	0.05	0.16	0.30	0.00	0.00	0.03	0.09
snapkv	0.80	0.82	0.81	0.78	0.43	0.44	0.42	0.45
streaming_llm	0.07	0.27	0.50	0.59	0.02	0.17	0.26	0.35

4.5 CACHE BUDGET VS OUTPUT LENGTH

We study the effects of cache budget on output generation lengths in Figure 4. Fascinatingly, lower budgets are capable of triggering longer reasoning traces, revealing a hidden tradeoff between cache budget and inference costs specifically for reasoning models. KNorm, arguably the lowest performing strategy, tends to cause the greatest elongation of outputs. In Section A.2, we examine one such non-terminating output that demonstrates repetitive, dead-end chain-of-thought.

4.6 ATTENTION AS AN INDICATOR OF PERFORMANCE

All eviction methods tested propose to capture important tokens via ad-hoc strategies either explicitly or implicitly relating to attention: H2O examines accumulated attention across the entire sequence, SnapKV examines attention with regards to an observation window, KNorm uses small key norms as a proxy for high-attention, StreamingLLM retains recent tokens and the sink (initial) to effectively approximate the attention distribution. We examine how much attention is actually lost through these various compression methods. For this study, we compare the absolute difference between the attention scores of each head pre- and post-eviction for GSM8K, which we refer to as attention loss following other recent literature (Liu et al., 2024a; Devoto et al., 2024). The trend is striking: in order

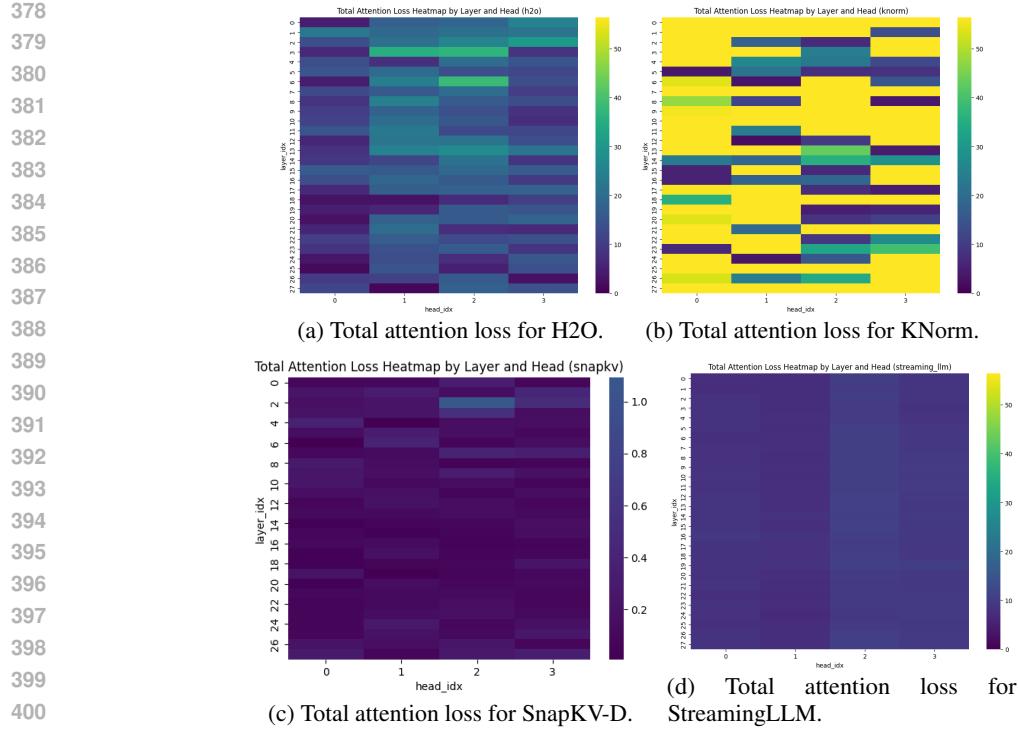


Figure 3: **Attention Loss Heatmaps.** We visualize attention loss at every compression step for a question in GSM8K. The attention loss over each head is summed up over every layer. We observe that higher performance correlates with less attention loss.

of least to most attention loss: SnapKV-D, H2O, StreamingLLM, and KNorm. [This correlates with average performance reported in Tables 2-4](#).

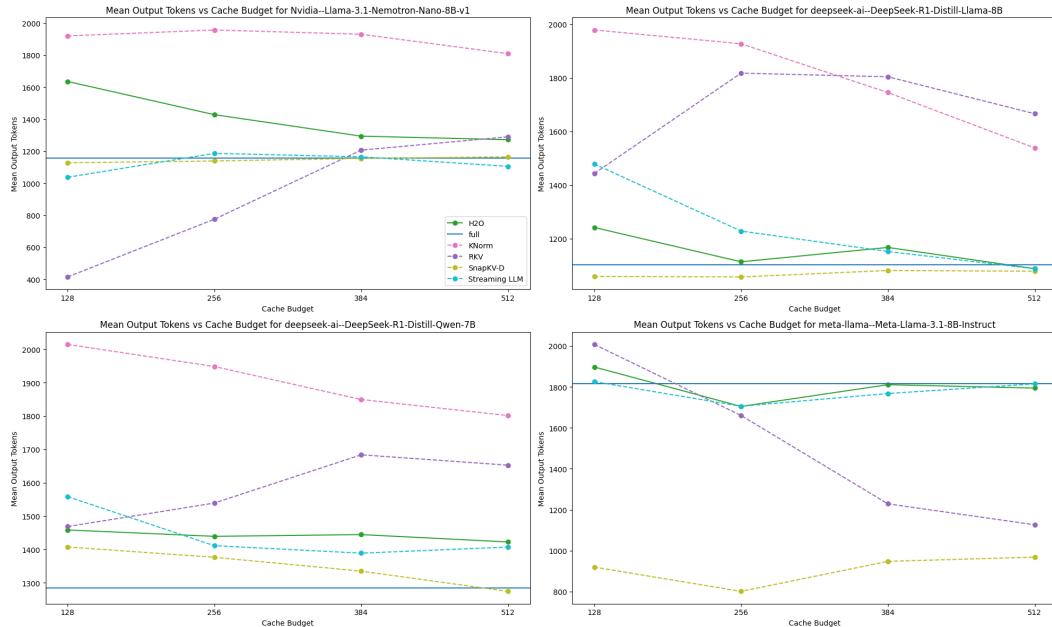


Figure 4: **Budget vs Output Length**. We observe that several compression methods, especially at lower budgets, ultimately produce longer outputs than the base full cache model.

432
433

4.7 ABLATION ANALYSIS & HIGH-LEVEL TRENDS

434
435
436
437
438
439
440
441

Attention is the most versatile estimator for reasoning models. SnapKV-D and H2O are the most dominant, significantly outcompeting nearly all compression strategies across all budget constraints and datasets for our reasoning models. These methods rely on accumulated attention scores to determine the most important tokens to retain. (i.e., “heavy hitters”). While both maintain a recency window, H2O is focused on heavy hitters with regard to the current token, while SnapKV (and consequently, SnapKV-D) finds heavy hitters with respect to an observation window at the end of the current sequence. The latter approach is more effective, routinely defeating H2O. The observation window was previously known to work well for prompt compression, but not for long decoding.

442
443
444
445
446
447
448

To further verify the utility of heavy-hitters for reasoning, in Appendix A.4, we track how many critical keywords for reasoning in GSM8K questions are present in the cache for each method, across models and budgets. Table 8 demonstrates that SnapKV-D and H2O retain critical tokens at a higher rate than other methods. Since they are exclusively focused on extracting heavy-hitters, *this implies that critical keywords display high accumulated attention throughout reasoning*. Figure 5 shows that the density of keywords in decoded context is much higher for uncompressed reasoning models versus non-reasoning models, thus their maintenance in the cache is ostensibly crucial for computation.

449
450
451
452
453
454

No singular strategy is dominant for the non-reasoning Llama-3.1-8B-Instruct. For models that do not produce reasoning traces, the optimal choice of strategy is dataset-dependent. For example, while StreamingLLM excels at GSM8K, it is less effective on all other task types. While SnapKV-D and H2O are capable of winning most settings for several datasets, other methods, such as KNorm and StreamingLLM, can win. According to Figure 5, keywords occur at a lower density in non-reasoning outputs, thus their maintenance might be less critical compared to reasoning outputs.

455
456
457
458
459
460

Eviction lags full cache performance for reasoning models. According to Table 1, all compression strategies can defeat the full cache performance of Llama-3.1-8B-Instruct on at least one setting (with H2O and SnapKV-D frequently achieving this). However, for reasoning models, this trend only holds true for SnapKV-D. While H2O is still second best compared to other strategies, it significantly lags full cache performance on nearly every dataset. As noted in Figure 4, H2O results in significantly longer reasoning traces than SnapKV-D, which occasionally do not terminate.

461
462
463
464
465
466

Cache compression can cost more computation. Interestingly, according to Figure 4, eviction strategies can result in more “talkative” reasoning models, generating noticeably longer sequences compared to the full cache setting, while this does not occur for Llama-3.1-8B-Instruct. In Section A.2, we show this phenomenon at work, where KNorm results in long circular babble for Deepseek-R1-Distill-Llama-8B that never produces an answer. At lower budgets, eviction occurs more frequently, resulting in a higher likelihood of critical token eviction, resulting in longer reasoning.

467
468
469
470
471
472
473
474
475

Practical Guidance. Although heavy-hitter methods dominate, selection of other methods may still prove appropriate. We summarize key selection rules. (1) Regardless of method, avoid a micro-budget. Performance stabilizes rapidly and very small budgets can counterintuitively increase the length of the output. (2) For large budgets, $B > 1024$, StreamingLLM is superior with smaller max token limits according to Figure 7. For any other budget and max token limit, SnapKV-D and H2O are preferable. (3) Use a larger window size for SnapKV-D. This decreases the frequency of eviction and, consequently, computational overhead with minimal performance differences (Table 10). (4) For reasoning models, accumulated attention scores are a high-quality token importance metric, therefore, lead with heavy-hitter methods for compression.

476
477

5 CONCLUSION

478
479
480
481
482
483
484
485

In this work, we comprehensively assessed the performance of several popular KV cache compression strategies on reasoning tasks. For a non-reasoning model, we find that no singular method is dominant. However, for reasoning models, we demonstrate that attention-based eviction methods such as H2O and SnapKV-D perform extraordinarily well on a variety of reasoning tasks, even occasionally exceeding full cache performance. Furthermore, this generalizes to a larger model, R1-Distill-Qwen-14B. We also discover that it is possible, especially at lower budgets, for compression strategies to produce longer reasoning traces, thus revealing an under-considered tradeoff between memory and inference costs.

486 6 ETHICS STATEMENT
487488 We do not anticipate any notable negative societal impacts stemming from this results discussed
489 in this work. However, we do note that KV cache compression is capable of altering outputs and
490 thus must be exercised with care in sensitive domains to ensure that content is not produced which
491 significantly deviates from uncompressed models.
492493 REFERENCES
494495 Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman,
496 Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report.
497 *arXiv preprint arXiv:2303.08774*, 2023.498 Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin
499 Jaggi, Dan Alistarh, Torsten Hoefer, and James Hensman. Quarot: Outlier-free 4-bit inference in
500 rotated llms. *Advances in Neural Information Processing Systems*, 37:100213–100240, 2024.501 Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao
502 Liu, Aohan Zeng, Lei Hou, et al. Longbench: A bilingual, multitask benchmark for long context
503 understanding. *arXiv preprint arXiv:2308.14508*, 2023.505 Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer.
506 *arXiv preprint arXiv:2004.05150*, 2020.507 Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal,
508 Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are
509 few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.511 Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong,
512 Yue Dong, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal
513 information funneling. *arXiv preprint arXiv:2406.02069*, 2024.514 Zefan Cai, Wen Xiao, Hanshi Sun, Cheng Luo, Yikai Zhang, Ke Wan, Yucheng Li, Yeyang Zhou, Li-
515 Wen Chang, Jiuxiang Gu, et al. R-kv: Redundancy-aware kv cache compression for training-free
516 reasoning models acceleration. *arXiv preprint arXiv:2505.24133*, 2025.517 Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong
518 Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, et al. Magicpig: Lsh sampling for efficient llm
519 generation. *arXiv preprint arXiv:2410.16179*, 2024.521 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
522 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve
523 math word problems. *arXiv preprint arXiv:2110.14168*, 2021.524 Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective l_2
525 norm-based strategy for kv cache compression. *arXiv preprint arXiv:2406.11430*, 2024.527 Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner.
528 Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *arXiv*
529 *preprint arXiv:1903.00161*, 2019.530 Alexander R Fabbri, Wojciech Kryściński, Bryan McCann, Caiming Xiong, Richard Socher, and
531 Dragomir Radev. Summeval: Re-evaluating summarization evaluation. *Transactions of the*
532 *Association for Computational Linguistics*, 9:391–409, 2021.533 Mehdi Fatemi, Banafsheh Rafiee, Mingjie Tang, and Kartik Talamadupula. Concise reasoning via
534 reinforcement learning. *arXiv preprint arXiv:2504.05185*, 2025.536 Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster,
537 Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff,
538 Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika,
539 Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation
harness, 07 2024. URL <https://zenodo.org/records/12608602>.

- 540 Yizhao Gao, Shuming Guo, Shijie Cao, Yuqing Xia, Yu Cheng, Lei Wang, Lingxiao Ma, Yutao Sun,
 541 Tianzhu Ye, Li Dong, et al. Seerattention-r: Sparse attention adaptation for long reasoning. *arXiv*
 542 *preprint arXiv:2506.08889*, 2025.
- 543 Suyu Ge, Yunan Zhang, Liyuan Liu, Minjia Zhang, Jiawei Han, and Jianfeng Gao. Model tells you
 544 what to discard: Adaptive kv cache compression for llms. *arXiv preprint arXiv:2310.01801*, 2023.
- 545 Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. Did aristotle
 546 use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of*
 547 *the Association for Computational Linguistics*, 9:346–361, 2021.
- 548 Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu,
 549 Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms
 550 via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.
- 551 Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P Woodruff, and Amir Zandieh.
 552 Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*,
 553 2023.
- 554 Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James
 555 Coady, David Peng, Yujie Qiao, Luke Benson, et al. Folio: Natural language reasoning with
 556 first-order logic. *arXiv preprint arXiv:2209.00840*, 2022.
- 557 Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Yakun S Shao, Kurt
 558 Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv
 559 cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.
- 560 Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang
 561 Zhang, and Boris Ginsburg. Ruler: What’s the real context size of your long-context language
 562 models? *arXiv preprint arXiv:2404.06654*, 2024.
- 563 Yejin Kwon, Daeun Moon, Youngje Oh, and Hyunsoo Yoon. Logicqa: Logical anomaly detection
 564 with vision language model generated questions. *arXiv preprint arXiv:2503.20252*, 2025.
- 565 Seungpil Lee, Woochang Sim, Donghyeon Shin, Wongyu Seo, Jiwon Park, Seokki Lee, Sanha Hwang,
 566 Sejin Kim, and Sundong Kim. Reasoning abilities of large language models: In-depth analysis on
 567 the abstraction and reasoning corpus. *ACM Transactions on Intelligent Systems and Technology*,
 568 2024.
- 569 Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai,
 570 Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation.
 571 *Advances in Neural Information Processing Systems*, 37:22947–22970, 2024.
- 572 Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan
 573 Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step. In *The Twelfth*
 574 *International Conference on Learning Representations*, 2023.
- 575 Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. Logiqa: A
 576 challenge dataset for machine reading comprehension with logical reasoning. *arXiv preprint*
 577 *arXiv:2007.08124*, 2020.
- 578 Minghui Liu, Tahseen Rabbani, Tony O’Halloran, Ananth Sankaralingam, Mary-Anne Hartley,
 579 Furong Huang, Cornelia Fermüller, and Yiannis Aloimonos. Hashevict: A pre-attention kv cache
 580 eviction strategy using locality-sensitive hashing. *arXiv preprint arXiv:2412.16187*, 2024a.
- 581 Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios
 582 Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance
 583 hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing*
 584 *Systems*, 36:52342–52364, 2023.
- 585 Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi
 586 Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint*
 587 *arXiv:2402.02750*, 2024b.

- 594 Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct
 595 electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*,
 596 2018.
- 597 Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language
 598 models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- 600 Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie
 601 Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm
 602 inference. *arXiv preprint arXiv:2410.21465*, 2024.
- 603 Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. Commonsenseqa: A question
 604 answering challenge targeting commonsense knowledge. *arXiv preprint arXiv:1811.00937*, 2018.
- 606 Jiaming Tang, Yilong Zhao, Kan Zhu, Guangxuan Xiao, Baris Kasikci, and Song Han. Quest:
 607 Query-aware sparsity for efficient long-context llm inference. *arXiv preprint arXiv:2406.10774*,
 608 2024.
- 609 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
 610 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and
 611 efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- 613 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz
 614 Kaiser, and Illia Polosukhin. Attention is all you need, 2017.
- 615 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
 616 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models.
 617 *arXiv preprint arXiv:2203.11171*, 2022.
- 619 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny
 620 Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in
 621 neural information processing systems*, 35:24824–24837, 2022.
- 622 Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming
 623 language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.
- 625 Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. Reclor: A reading comprehension dataset
 626 requiring logical reasoning. *arXiv preprint arXiv:2002.04326*, 2020.
- 627 Jingyang Yuan, Huazuo Gao, Damai Dai, Junyu Luo, Liang Zhao, Zhengyan Zhang, Zhenda Xie,
 628 Yuxing Wei, Lean Wang, Zhiping Xiao, et al. Native sparse attention: Hardware-aligned and
 629 natively trainable sparse attention. In *Proceedings of the 63rd Annual Meeting of the Association
 630 for Computational Linguistics (Volume 1: Long Papers)*, pp. 23078–23097, 2025.
- 631 Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine
 632 really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.
- 634 Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. Ratt: A
 635 thought structure for coherent and correct llm reasoning. In *Proceedings of the AAAI Conference
 636 on Artificial Intelligence*, volume 39, pp. 26733–26741, 2025.
- 638 Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song,
 639 Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient
 640 generative inference of large language models. *Advances in Neural Information Processing
 641 Systems*, 36:34661–34710, 2023.
- 642
- 643
- 644
- 645
- 646
- 647

648 **A APPENDIX**
649650 **A.1 GENERATION LENGTHS**
651652 In Table 7, we report the mean generation lengths for all methods across and models on MATH-500,
653 the dataset which elicits the longest responses. To keep these tables concise, we averaged output
654 lengths over all budgets.655 Table 7: Mean output tokens generated by different models under various strategies for Math500.
656

657 Strategy	658 Model	659 Mean Output Tokens
660 full	Nvidia--Llama-3.1-Nemotron-Nano-8B-v1	1616.275
661 full	deepseek-ai--DeepSeek-R1-Distill-Llama-8B	1727.18
662 full	deepseek-ai--DeepSeek-R1-Distill-Qwen-7B	1728.84
663 h2o	Nvidia--Llama-3.1-Nemotron-Nano-8B-v1	1753.4075
664 h2o	deepseek-ai--DeepSeek-R1-Distill-Llama-8B	1763.805
665 h2o	deepseek-ai--DeepSeek-R1-Distill-Qwen-7B	1767.7275
666 knorm	Nvidia--Llama-3.1-Nemotron-Nano-8B-v1	1987.80875
667 knorm	deepseek-ai--DeepSeek-R1-Distill-Llama-8B	2001.07625
668 knorm	deepseek-ai--DeepSeek-R1-Distill-Qwen-7B	1967.4575
669 snapkv	Nvidia--Llama-3.1-Nemotron-Nano-8B-v1	1667.895
670 snapkv	deepseek-ai--DeepSeek-R1-Distill-Llama-8B	1790.165
671 snapkv	deepseek-ai--DeepSeek-R1-Distill-Qwen-7B	1794.0275
672 streaming_llm	Nvidia--Llama-3.1-Nemotron-Nano-8B-v1	1375.70375
673 streaming_llm	deepseek-ai--DeepSeek-R1-Distill-Llama-8B	1655.0375
674 streaming_llm	deepseek-ai--DeepSeek-R1-Distill-Qwen-7B	1698.25625

675 **A.2 A VERBOSE, WRONG ANSWER**676 Below, we present a verbose, incorrect answer generated for Deepseek-R1-Distill-Llama-8B via
677 KNorm on a budget of 256 tokens.

678 Define:

679

680 - Let a be the value of a small button (SB).
681 - Then, 1 SS = a
682 - 1 large button (LB) is worth 3 small buttons, so $LB = 3a$
683 - Alternatively, 1 LB = $3a$
684 - 1 LS can be converted to 1 LB or 3 SS, so $1 LS = 1 LB$ or $3*(1a)$? Wait,
685 small stickers themselves are worth $1a$.

686 Hold on, perhaps clarifying units.

687 Wait, sorry, the problem states:

688 "Lorraine has ... stickers and large buttons..."

689 Wait, maybe it's better to model it with equations.

690 Let me denote:

691 - Value of 1 small sticker = x
692 - Then, 1 medium sticker would be higher, but in our case, 1 large
693 sticker (LS) is worth either 1 large button (LB) or 3 small buttons (SB).

694 But we don't know whether a small button is same as a small sticker? Let's see.

695 The problem says:

702 Table 8: Strategy vs. Critical Token Retention Rate. Cache budgets (MB) = [128, 256, 384, 512]. Models:
 703 Llama-3.1-8B-Instruct (ML), DeepSeek-R1-Distill-Qwen-7B (DQ), Nemotron-Nano-8B-v1 (LN), DeepSeek-
 704 R1-Distill-Llama-8B (DL).
 705

Benchmark	ML				DQ				LN				DL			
	128	256	384	512	128	256	384	512	128	256	384	512	128	256	384	512
GSM8K	full	77.84				77.46%				81.62%				75.59%		
h2o	68.11%	68.11%	69.19%	70.27%	72.30%	72.30%	73.24%	74.18%	68.11%	68.11%	69.19%	70.27%	71.61%	71.61%	72.30%	72.30%
knorm	68.11%	68.11%	69.19%	69.73%	67.60%	67.60%	68.72%	69.27%	65.48%	65.48%	66.67%	67.26%	68.11%	68.11%	69.19%	69.73%
SnapKV	68.11%	68.11%	68.65%	69.73%	72.30%	72.30%	73.24%	74.18%	68.11%	68.11%	69.19%	70.27%	68.11%	68.11%	69.19%	70.27%
streaming_llm	67.03%	67.03%	67.57%	69.73%	71.36%	71.36%	72.30%	73.24%	67.03%	67.03%	68.11%	69.19%	67.03%	67.03%	68.11%	69.19%

710
 711 - A large button is worth three small stickers. So 1 LB = 3 SS.
 712

713 So, 1 SS = 1/3 LB.
 714

715 Or, alternatively, 1 LB = 3 SS.
 716

717 So, yeah, 1 LB = 3 SS.
 718

719 On the flip side, 1 LS can be either 1 LB or 3 SS.
 720

721 But note that 1 LB is equal to 3 SS.
 722

723 So, the question becomes:
 724

725 Traders use sticker-based currencies.
 726

727 Given that, let me

728 A.3 LLM USAGE

729 We used LLMs to stylize tables and figures. We also used LLMs to polish grammar but not to produce
 730 any writing itself.
 731

733 A.4 RETENTION OF CRITICAL TOKENS

735 To understand why reasoning oriented models such as DeepSeek-R1-Distill-Qwen-7B and DeepSeek-
 736 R1-Distill-Llama-8B exhibit different performance patterns, we evaluate GSM8K questions and
 737 measure how well critical tokens (e.g., names, quantities, and key entities) are retained in the KV
 738 cache at the final decoding step. Table 8 reports the critical token retention rate, defined as the
 739 fraction of critical tokens that remain available in the cache at the end of generation. Figure 5 further
 740 visualizes a randomly selected GSM8K example, highlighting which critical tokens persist in the
 741 cache and how they appear in the model’s final answer. From both the quantitative results and the
 742 visualization, we observe that reasoning models consistently retain a larger proportion of critical
 743 tokens than standard instruction tuned baselines. In particular, reasoning models tend to preserve task
 744 relevant concepts such as minutes, day, and similar numerical or semantic anchors, suggesting that
 745 they rely more directly on these tokens throughout the multistep reasoning process.

746 We find that reasoning-trained models consistently retain a larger proportion of critical tokens
 747 compared with standard instruction-tuned models. Both the numerical metrics and qualitative
 748 visualizations indicate that reasoning models rely more heavily on these critical tokens throughout the
 749 problem-solving process. In the example visualization, the reasoning model generates more critical
 750 tokens in its final answer, which helps explain why heavy-hitter or attention-based token-retention
 751 strategies are particularly effective for reasoning models: critical tokens tend to persistently exhibit
 752 high-attention throughout decoding.

753 This observation directly supports our main conclusion that accumulated attention is the most effective
 754 importance heuristic for reasoning eviction. Since RL-distilled reasoning models naturally focus on
 755 preserving critical tokens, methods such as **H2O** and **SnapKV-D** yield larger performance gains by
 ensuring those tokens remain in the KV cache. In contrast, non-reasoning models do not exhibit this

Okay, so I need to figure out how many minutes Kim slept on her bed that day. Let me start by understanding all the information given.

Firstly, normally Kim goes to sleep at 10 p.m. (which is 20:00) and wakes up at 6 a.m. next morning (which would be 6:00). So let's calculate her usual total sleeping time first before considering to day's anomaly because sometimes it helps have some context.

From 10 p.m. to midnight is 2 hours, which is $1\frac{2}{3}$ hours. Then from midnight to 6 a.m. is another 6 hours, which is $3\frac{2}{3}$ hours. So altogether, normal sleeping duration is $1\frac{2}{3} + 3\frac{2}{3} = 5$ hours = 300 minutes each night.

But one day, something different happened. On this particular day:

- She went to sleep as usual? Or maybe not... Wait, no—the problem says "that day" meaning only that specific instance where we're analyzing what happened differently.

Wait—let me read again carefully:

"She installed a camera ... Kim usually goes to sleep at 10 p.m., but that day, she found herself waking up 5 minutes earlier than usual." Hmm...

Meta-Llama3-8B-Instruct

Step 1: Determine normal bedtime and wake-up time

Normal bedtime for Kim is 10 p.m., and normal wake-up time is 6 a.m.

Step 2: Calculate total number of minutes slept normally per night

There are 8 hours between 10 p.m. and 6 a.m. Since there are 60 minutes in an hour, this means that under normal circumstances, Kim sleeps for $(8 * 60) = 480$ minutes every night.

Step 3: Find out how early Kim went to sleep that particular evening.

Since Kim fell asleep while sleepwalking from 2:15 to 2:38 AM but actually started waking up 5 minutes before regular wake-up time, we need to determine what time it would be if she had gone to bed at the normal time instead of falling asleep during the middle of the night.

Step 4: Convert wake-up times into just minutes past midnight

For calculation purposes, let's convert all relevant times into minutes since midnight:

- Normal wake-up time is 6 a.m., which equals $3\frac{2}{3}$ hours past midnight (since $1\frac{2}{3}$ am = 0).
- Sleep period due to being awake early: From 2:15 A.M. till 2:38 A.M. is $(2\frac{3}{4} * 60) + 15 = 180 + 15 = 195$ minutes past midnight

Figure 5: **A generated answer by DeepSeek-R1-Distill-Qwen-7B and Meta-Llama3-8B-Instruct.** Critical tokens rendered in blue and red boxes respectively. Reasoning models outputs contain a higher density of critical tokens. *Prompt: Kim sleepwalks, to monitor her sleeping hours, she installs a camera in her room. Kim usually goes to sleep at 10 p.m. and wakes up at 6 a.m. She is a sleepwalker. One day, after reviewing the cameras, she finds that she woke up that day and was sleepwalking from 2:15 to 2:38 am. Also, that day she woke up 5 minutes earlier than usual to go to the bathroom. How many minutes did she sleep on her bed that day?*

strong reliance on critical tokens, so no single cache-retention strategy is universally optimal across model types.

A.5 MEMORY PERFORMANCE ANALYSIS

In this section, we report the peak memory usage between methods.

Table 9: Peak allocated GPU memory (GB) by press and cache budget on GSM8K.

press	128	256	384	512
Full	14.88	14.88	14.88	14.88
H ₂ O	14.81	14.81	14.82	14.83
KNorm	14.79	14.80	14.81	14.83
R-KV	14.80	14.81	14.83	14.84
StreamingLLM	14.79	14.80	14.81	14.83

A.6 WINDOW SIZE

SnapKV-D using a sliding window of fixed size w to determine critical tokens. In particular, after the budget is filled to capacity, every w decoding steps, SnapKV-D measures the attention scores of all current tokens in the cache against the w most recent tokens (the aggregation is described in (Li et al., 2024)). Those with the lowest scores are dropped to meet the budget again.

In our core experiments, we set $w = 128$, which is recommended by both the authors and the kvpress library. We perform an ablation size over window size in Table 10. The effects are only noticeable at lower budgets, where smaller window size forces more frequent re-assessment of critical tokens in the cache, thus maintaining tokens more relevant to incoming context. However, this costs more overhead: if the total decoded output is length N , we are performing N/w applications of SnapKV-D.

810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
Table 10: Model Performance across Cache Budgets and Window Sizes for SnapKV

Model	Window Size	Cache Budget			
		128	256	384	512
deepseek-ai/DeepSeek-R1-Distill-Llama-8B	16	0.71	0.69	0.80	0.69
	32	0.74	0.71	0.73	0.73
	64	0.72	0.72	0.74	0.72
	128	0.72	0.69	0.75	0.70
Nvidia/Llama-3.1-Nemotron-Nano-8B-v1	16	0.71	0.69	0.66	0.66
	32	0.71	0.66	0.68	0.67
	64	0.65	0.63	0.66	0.66
	128	0.72	0.73	0.72	0.67
deepseek-ai/DeepSeek-R1-Distill-Qwen-7B	16	0.74	0.68	0.65	0.67
	32	0.69	0.71	0.68	0.68
	64	0.67	0.67	0.70	0.71
	128	0.66	0.65	0.70	0.72

Our results illustrate that for larger budgets, wider window sizes should be used since this both improves accuracy and reduces computation.

A.7 PYRAMIDKV ANALYSIS

PyramidKV is a dynamic KV-cache compression method that is built around the idea of pyramidal information funneling: in early Transformer layers, attention is spread broadly over many tokens, while in deeper layers it becomes concentrated on a small subset of salient tokens. Under a fixed overall KV budget, PyramidKV therefore allocates larger cache sizes to lower layers and progressively smaller caches to higher layers, forming a pyramid-shaped retention profile across depth. Within each layer, it uses attention patterns to decide which keys and values to keep (e.g., tokens that are strongly attended to by query/instruction tokens are preferentially retained), so that the cache focuses on the most informative context while still substantially reducing memory usage.

Dataset	Model	Budget			
		128	256	384	512
GSM8K					
gsm8k	DeepSeek-R1-Distill-Llama-8B	0.01	0.01	0.10	0.22
gsm8k	DeepSeek-R1-Distill-Qwen-7B	0.00	0.03	0.09	0.25
gsm8k	Meta-Llama-3.1-8B-Instruct	0.03	0.38	0.72	0.79
gsm8k	Llama-3.1-Nemotron-Nano-8B-v1	0.01	0.01	0.03	0.20
MATH500					
math500	DeepSeek-R1-Distill-Llama-8B	0.00	0.00	0.00	0.00
math500	DeepSeek-R1-Distill-Qwen-7B	0.00	0.01	0.01	0.06
math500	Meta-Llama-3.1-8B-Instruct	0.01	0.05	0.16	0.25
math500	Llama-3.1-Nemotron-Nano-8B-v1	0.02	0.02	0.04	0.04

Table 11: Test accuracy on the GSM8K and MATH500 test sets for each model and KV-cache budget. Budgets (128, 256, 384, 512) index the maximum KV-cache size in tokens, and each cell reports the corresponding accuracy at that budget.

A.8 COMPARISON WITH A SPARSE ATTENTION METHOD

Although our main benchmark evaluates KV Cache pressing methods, we extend our study to compare these results with other architectural categories, such as sparse attention methods. Such methods (Gao et al., 2025; Yuan et al., 2025) train their architectures to enforce sparse attention computations by learning to identify and cluster critical tokens. This is in contrast to eviction methods which are generally training-free. Furthermore, these methods are not memory-bound and host the full KV cache.

We perform a comparative evaluation of sparse decoding modeling with SeerAttention using the SeerAttention-Decode-R1-Distill-Qwen-14B model and present results in Table 12. This analysis is performed on the GSM8K benchmark using a randomly sampled subset of 100 questions. We can see that SnapKV-D and SeerAttention are close in performance with SnapKV-D as slightly better. *Further note that SeerAttention must maintain the full cache which scales with sequence length, while SnapKV-D maintains a fixed size cache.*

Table 12: **SeerAttention-R1-Distill-Qwen14B.** Cache budgets = [128, 256, 384, 512]. We examine the performance of SeerAttention together with H2O and SnapKV-D. Winner per budget in bold.

Method	GSM8K			
	128	256	384	512
H2O	0.33	0.56	0.62	0.64
SnapKV-D	0.80	0.82	0.81	0.78
SeerAttention	0.66	0.80	0.82	0.70

A.9 COMPUTATIONAL OVERHEAD

In this section, we describe the asymptotic computational overhead of each method. More specifically, in Table 13, we report the computational complexity of cache eviction throughout the decoded sequence. Once the cache is evicted down to the budget B , the attention calculation is $\mathcal{O}_d(1)$ (since there are only B tokens of dimension d to compute attention over). Thus, we are interested in comparing the complexity of *evaluating token importance itself*.

Any method relying on accumulated attention scores (H2O & SnapKV-D) incurs a greater cost. StreamingLLM and KNorm are comparatively cheap; the former just keeps a few sink tokens and recent context, while the latter simply evicts the token with the largest key norm. These latencies are reflected accordingly in Figure 2.

Table 13: Decoding computational overhead of importance estimation for our tested KV cache compression methods. B is the cache budget, N is the decoded sequence length, and d the key dimension. For simplicity, we are assuming that the budget is filled after pre-fill and that each attention layer is single-head.

Method	Overhead	Approach
StreamingLLM	$\mathcal{O}(1)$	Maintain sink token + recent tokens.
H2O	$\mathcal{O}(N B d)$	Next token in, token in cache with lowest average accumulated attention score is out.
SnapKV-D	$\mathcal{O}(\frac{N}{w} B d)$	Keep the next window of w tokens, evict tokens in the cache with low accumulated attention score against the window.
R-KV	$\mathcal{O}(\frac{N}{w} B^2 d)$	Measures token redundancy (key cache self-product) and accumulated attention scores against the query every w steps.
Knorm	$\mathcal{O}(N)$	Evicts the token in the cache with the largest ℓ_2 norm.

A.10 THE TRANSFORMER DECODER ARCHITECTURE AND INFERENCE

We visualize decoder-only inference and the role of the cache. Figure 6 (a) exhibits the Q , K , and V vectors along with the self-attention mechanism. Figure 6 (b) demonstrate the decoding KV cache bottleneck on memory.

A.11 HYPERPARAMETER DETAILS

In this section, we describe the hyperparameter details for all eviction methods. We tend towards the default hyper-parameters set by KVPress, which are typically author-recommend selections.

H2O: H_2 tokens kept

PyramidKV: window size 64, kernel size 5, $\beta = 20$

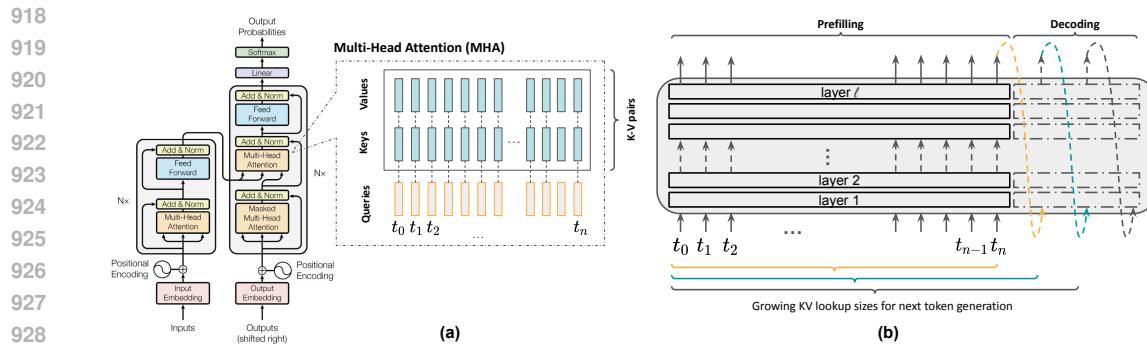


Figure 6: **Overview of the Transformer Decoder Architecture and the Inference Bottleneck.** (a) The standard Transformer decoder architecture (left) and the Multi-Head Attention (MHA) mechanism (right). In MHA, Query vectors representing the current context attend to a sequence of Key-Value (K-V) pairs from all previous tokens. Such K-V pairs form the basis of the KV cache. (b) The two-phase inference process in autoregressive generation. During Prefilling, the tokens in the input context are processed in parallel to populate the initial KV cache across all layers. During Decoding, each new token is generated sequentially. This requires recomputing the entire set of the preceding KV entries at each step, causing the lookup size to grow linearly with the sequence length.

StreamingLLM: Sink retention, first 4 tokens.

SnapKV-D: Observation window size 128.

KNorm: $k = 2$

R-KV: window size 8, buffer interval 128, kernel size 5.

A.12 MAX TOKEN ABLATION

We study the effects of max token limit on performance under a fixed budget of 1024 tokens. Results are presented in Figure 7.



Figure 7: **Performance versus max tokens permitted.** Fixed budget of 1024.

A.13 CONFIDENCE INTERVALS

For each configuration (model, dataset, cache budget, method), we estimate uncertainty by computing a binomial confidence interval over correctness across runs, as shown in Table 14. Specifically, we evaluate 3 independent seeds, each on 100 questions, and treat the resulting 300 binary outcomes (correct/incorrect) as Bernoulli trials with unknown success probability. The sample accuracy for that configuration is the proportion of correct answers over these 300 trials, and we then compute a 95% Wilson score confidence interval for this underlying accuracy parameter (using the normal

approximation with $z = 1.96$. The table cells in this appendix report only these Wilson intervals $[L, U]$, omitting the point estimates, which are shown separately in the main results Tables 1-4.

Table 14: Confidence intervals across cache budgets (Continuous)

Budget	128	256	384	512	128	256	384	512
Llama-3.1-8B-Instruct								
Method	GSM8K					Math500		
Full	[0.838, 0.912]					[0.337, 0.446]		
ShadowKV	[0.270, 0.375]					[0.177, 0.270]		
H2O	[0.574, 0.683]	[0.719, 0.814]	[0.773, 0.859]	[0.783, 0.868]	[0.251, 0.354]	[0.279, 0.385]	[0.279, 0.385]	[0.308, 0.416]
Knorm	[0.031, 0.081]	[0.473, 0.586]	[0.677, 0.777]	[0.773, 0.859]	[0.016, 0.056]	[0.141, 0.227]	[0.177, 0.270]	[0.279, 0.385]
RKV	[0.088, 0.162]	[0.289, 0.395]	[0.444, 0.556]	[0.434, 0.546]	[0.016, 0.056]	[0.071, 0.139]	[0.123, 0.206]	[0.159, 0.249]
SnapKV	[0.473, 0.586]	[0.493, 0.605]	[0.503, 0.615]	[0.473, 0.586]	[0.159, 0.249]	[0.168, 0.260]	[0.150, 0.238]	[0.159, 0.249]
StreamingLLM	[0.214, 0.312]	[0.698, 0.796]	[0.794, 0.877]	[0.827, 0.903]	[0.079, 0.150]	[0.214, 0.312]	[0.270, 0.375]	[0.298, 0.406]
Method	CSQA					OBQA		
Full	[0.719, 0.814]					[0.794, 0.877]		
ShadowKV	[0.159, 0.249]					[0.260, 0.364]		
H2O	[0.688, 0.786]	[0.709, 0.805]	[0.719, 0.814]	[0.719, 0.814]	[0.783, 0.868]	[0.816, 0.895]	[0.816, 0.895]	[0.816, 0.895]
Knorm	[0.289, 0.395]	[0.719, 0.814]	[0.698, 0.796]	[0.709, 0.805]	[0.356, 0.466]	[0.740, 0.832]	[0.794, 0.877]	[0.773, 0.859]
RKV	[0.308, 0.416]	[0.564, 0.673]	[0.709, 0.805]	[0.719, 0.814]	[0.177, 0.270]	[0.605, 0.711]	[0.719, 0.814]	[0.794, 0.877]
SnapKV	[0.646, 0.749]	[0.584, 0.692]	[0.656, 0.758]	[0.667, 0.768]	[0.677, 0.777]	[0.719, 0.814]	[0.667, 0.768]	[0.709, 0.805]
StreamingLLM	[0.159, 0.249]	[0.698, 0.796]	[0.709, 0.805]	[0.719, 0.814]	[0.105, 0.184]	[0.667, 0.768]	[0.794, 0.877]	[0.794, 0.877]
Method	ReClor					DROP		
Full	[0.544, 0.654]					[0.114, 0.195]		
ShadowKV	[0.223, 0.323]					[0.232, 0.333]		
H2O	[0.270, 0.375]	[0.503, 0.615]	[0.544, 0.654]	[0.523, 0.634]	[0.088, 0.162]	[0.105, 0.184]	[0.132, 0.217]	[0.132, 0.217]
Knorm	[0.003, 0.029]	[0.150, 0.238]	[0.404, 0.517]	[0.534, 0.644]	[0.003, 0.029]	[0.054, 0.116]	[0.097, 0.173]	[0.097, 0.173]
RKV	[0.023, 0.069]	[0.168, 0.260]	[0.346, 0.456]	[0.483, 0.596]	[0.038, 0.093]	[0.046, 0.105]	[0.105, 0.184]	[0.079, 0.150]
SnapKV	[0.473, 0.586]	[0.513, 0.625]	[0.523, 0.634]	[0.493, 0.605]	[0.114, 0.195]	[0.088, 0.162]	[0.079, 0.150]	[0.088, 0.162]
StreamingLLM	[0.031, 0.081]	[0.168, 0.260]	[0.534, 0.644]	[0.523, 0.634]	[0.063, 0.128]	[0.079, 0.150]	[0.114, 0.195]	[0.123, 0.206]
Method	StrategyQA					FOLIO		
Full	[0.783, 0.868]					[0.454, 0.566]		
ShadowKV	[0.625, 0.730]					[0.279, 0.385]		
H2O	[0.762, 0.850]	[0.827, 0.903]	[0.838, 0.912]	[0.850, 0.921]	[0.177, 0.270]	[0.375, 0.487]	[0.356, 0.466]	[0.375, 0.487]
Knorm	[0.414, 0.527]	[0.805, 0.886]	[0.838, 0.912]	[0.827, 0.903]	[0.009, 0.043]	[0.232, 0.333]	[0.337, 0.446]	[0.327, 0.436]
RKV	[0.544, 0.654]	[0.740, 0.832]	[0.719, 0.814]	[0.740, 0.832]	[0.046, 0.105]	[0.308, 0.416]	[0.385, 0.497]	[0.289, 0.395]
SnapKV	[0.730, 0.823]	[0.730, 0.823]	[0.762, 0.850]	[0.709, 0.805]	[0.385, 0.497]	[0.346, 0.456]	[0.395, 0.507]	[0.404, 0.517]
StreamingLLM	[0.079, 0.150]	[0.709, 0.805]	[0.850, 0.921]	[0.805, 0.886]	[0.016, 0.056]	[0.063, 0.128]	[0.204, 0.302]	[0.298, 0.406]
Deepseek-R1-Distill-Qwen-7B								
Method	GSM8K					Math500		
Full	[0.646, 0.749]					[0.414, 0.527]		
ShadowKV	[0.414, 0.527]					[0.279, 0.385]		
H2O	[0.168, 0.260]	[0.385, 0.497]	[0.454, 0.566]	[0.464, 0.576]	[0.105, 0.184]	[0.168, 0.260]	[0.242, 0.344]	[0.260, 0.364]
Knorm	[0.000, 0.013]	[0.000, 0.013]	[0.054, 0.116]	[0.123, 0.206]	[0.000, 0.013]	[0.003, 0.029]	[0.016, 0.056]	[0.031, 0.081]
RKV	[0.023, 0.069]	[0.046, 0.105]	[0.141, 0.227]	[0.251, 0.354]	[0.023, 0.069]	[0.023, 0.069]	[0.031, 0.081]	[0.132, 0.217]
SnapKV	[0.615, 0.721]	[0.615, 0.721]	[0.646, 0.749]	[0.656, 0.758]	[0.327, 0.436]	[0.308, 0.416]	[0.308, 0.416]	[0.270, 0.375]
StreamingLLM	[0.009, 0.043]	[0.150, 0.238]	[0.270, 0.375]	[0.385, 0.497]	[0.016, 0.056]	[0.088, 0.162]	[0.150, 0.238]	[0.214, 0.312]
Method	CSQA					OBQA		
Full	[0.615, 0.721]					[0.730, 0.823]		
ShadowKV	[0.159, 0.249]					[0.260, 0.364]		
H2O	[0.385, 0.497]	[0.554, 0.663]	[0.544, 0.654]	[0.584, 0.692]	[0.366, 0.477]	[0.584, 0.692]	[0.636, 0.740]	[0.615, 0.721]
Knorm	[0.031, 0.081]	[0.097, 0.173]	[0.251, 0.354]	[0.366, 0.477]	[0.016, 0.056]	[0.031, 0.081]	[0.186, 0.281]	[0.327, 0.436]
RKV	[0.071, 0.139]	[0.063, 0.128]	[0.223, 0.323]	[0.289, 0.395]	[0.071, 0.139]	[0.071, 0.139]	[0.168, 0.260]	[0.214, 0.312]
SnapKV	[0.594, 0.702]	[0.564, 0.673]	[0.534, 0.644]	[0.554, 0.663]	[0.656, 0.758]	[0.698, 0.796]	[0.625, 0.730]	[0.709, 0.805]
StreamingLLM	[0.054, 0.116]	[0.105, 0.184]	[0.260, 0.364]	[0.424, 0.536]	[0.009, 0.043]	[0.079, 0.150]	[0.232, 0.333]	[0.317, 0.426]
Method	ReClor					DROP		
Full	[0.395, 0.507]					[0.123, 0.206]		
ShadowKV	[0.223, 0.323]					[0.105, 0.184]		
H2O	[0.003, 0.029]	[0.023, 0.069]	[0.141, 0.227]	[0.232, 0.333]	[0.023, 0.069]	[0.046, 0.105]	[0.071, 0.139]	[0.071, 0.139]
Knorm	[0.000, 0.013]	[0.000, 0.013]	[0.003, 0.029]	[0.003, 0.029]	[0.000, 0.013]	[0.003, 0.029]	[0.003, 0.029]	[0.016, 0.056]
RKV	[0.023, 0.069]	[0.016, 0.056]	[0.009, 0.043]	[0.003, 0.029]	[0.023, 0.069]	[0.023, 0.069]	[0.016, 0.056]	[0.023, 0.069]
SnapKV	[0.395, 0.507]	[0.337, 0.446]	[0.346, 0.456]	[0.375, 0.487]	[0.097, 0.173]	[0.079, 0.150]	[0.088, 0.162]	[0.123, 0.206]
StreamingLLM	[0.000, 0.013]	[0.003, 0.029]	[0.003, 0.029]	[0.023, 0.069]	[0.023, 0.069]	[0.031, 0.081]	[0.054, 0.116]	[0.097, 0.173]

Continued on next page...

Table 14: Confidence intervals (continued)...

Budget	128	256	384	512	128	256	384	512
Method	StrategyQA				FOLIO			
Full	[0.615, 0.721]				[0.308, 0.416]			
ShadowKV	[0.544, 0.654]				[0.279, 0.385]			
H2O	[0.279, 0.385]	[0.584, 0.692]	[0.688, 0.786]	[0.667, 0.768]	[0.016, 0.056]	[0.168, 0.260]	[0.186, 0.281]	[0.186, 0.281]
Knorm	[0.000, 0.013]	[0.088, 0.162]	[0.385, 0.497]	[0.534, 0.644]	[0.000, 0.013]	[0.003, 0.029]	[0.016, 0.056]	[0.031, 0.081]
RKV	[0.031, 0.081]	[0.105, 0.184]	[0.289, 0.395]	[0.366, 0.477]	[0.023, 0.069]	[0.016, 0.056]	[0.009, 0.043]	[0.038, 0.093]
SnapKV	[0.544, 0.654]	[0.534, 0.644]	[0.513, 0.625]	[0.574, 0.683]	[0.251, 0.354]	[0.204, 0.302]	[0.260, 0.364]	[0.242, 0.344]
StreamingLLM	[0.000, 0.013]	[0.031, 0.081]	[0.177, 0.270]	[0.366, 0.477]	[0.000, 0.013]	[0.003, 0.029]	[0.009, 0.043]	[0.016, 0.056]
Nemotron-Nano-8B								
Method	GSM8K				Math500			
Full	[0.584, 0.692]				[0.395, 0.507]			
ShadowKV	[0.385, 0.497]				[0.232, 0.333]			
H2O	[0.177, 0.270]	[0.395, 0.507]	[0.464, 0.576]	[0.513, 0.625]	[0.123, 0.206]	[0.195, 0.291]	[0.260, 0.364]	[0.279, 0.385]
Knorm	[0.003, 0.029]	[0.009, 0.043]	[0.063, 0.128]	[0.141, 0.227]	[0.003, 0.029]	[0.003, 0.029]	[0.016, 0.056]	[0.038, 0.093]
RKV	[0.023, 0.069]	[0.016, 0.056]	[0.063, 0.128]	[0.114, 0.195]	[0.009, 0.043]	[0.023, 0.069]	[0.016, 0.056]	[0.038, 0.093]
SnapKV	[0.594, 0.702]	[0.574, 0.683]	[0.605, 0.711]	[0.605, 0.711]	[0.356, 0.466]	[0.385, 0.497]	[0.395, 0.507]	[0.375, 0.487]
StreamingLLM	[0.016, 0.056]	[0.159, 0.249]	[0.346, 0.456]	[0.473, 0.586]	[0.009, 0.043]	[0.097, 0.173]	[0.177, 0.270]	[0.289, 0.395]
Method	CSQA				OBQA			
Full	[0.454, 0.566]				[0.584, 0.692]			
ShadowKV	[0.159, 0.249]				[0.260, 0.364]			
H2O	[0.414, 0.527]	[0.434, 0.546]	[0.464, 0.576]	[0.454, 0.566]	[0.534, 0.644]	[0.534, 0.644]	[0.523, 0.634]	[0.564, 0.673]
Knorm	[0.308, 0.416]	[0.346, 0.456]	[0.385, 0.497]	[0.404, 0.517]	[0.270, 0.375]	[0.385, 0.497]	[0.424, 0.536]	[0.513, 0.625]
RKV	[0.232, 0.333]	[0.251, 0.354]	[0.366, 0.477]	[0.356, 0.466]	[0.298, 0.406]	[0.385, 0.497]	[0.454, 0.566]	[0.454, 0.566]
SnapKV	[0.434, 0.546]	[0.444, 0.556]	[0.454, 0.566]	[0.473, 0.586]	[0.625, 0.730]	[0.574, 0.683]	[0.605, 0.711]	[0.605, 0.711]
StreamingLLM	[0.308, 0.416]	[0.385, 0.497]	[0.404, 0.517]	[0.444, 0.556]	[0.308, 0.416]	[0.404, 0.517]	[0.464, 0.576]	[0.564, 0.673]
Method	ReClor				DROP			
Full	[0.424, 0.536]				[0.079, 0.150]			
ShadowKV	[0.223, 0.323]				[0.079, 0.150]			
H2O	[0.159, 0.249]	[0.177, 0.270]	[0.298, 0.406]	[0.346, 0.456]	[0.031, 0.081]	[0.038, 0.093]	[0.071, 0.139]	[0.063, 0.128]
Knorm	[0.003, 0.029]	[0.016, 0.056]	[0.046, 0.105]	[0.046, 0.105]	[0.003, 0.029]	[0.003, 0.029]	[0.009, 0.043]	[0.016, 0.056]
RKV	[0.016, 0.056]	[0.054, 0.116]	[0.054, 0.116]	[0.046, 0.105]	[0.009, 0.043]	[0.038, 0.093]	[0.031, 0.081]	[0.016, 0.056]
SnapKV	[0.366, 0.477]	[0.366, 0.477]	[0.366, 0.477]	[0.317, 0.426]	[0.079, 0.150]	[0.079, 0.150]	[0.088, 0.162]	[0.071, 0.139]
StreamingLLM	[0.016, 0.056]	[0.038, 0.093]	[0.063, 0.128]	[0.105, 0.184]	[0.016, 0.056]	[0.009, 0.043]	[0.038, 0.093]	[0.054, 0.116]
Method	StrategyQA				FOLIO			
Full	[0.850, 0.921]				[0.308, 0.416]			
ShadowKV	[0.594, 0.702]				[0.279, 0.385]			
H2O	[0.709, 0.805]	[0.794, 0.877]	[0.805, 0.886]	[0.783, 0.868]	[0.177, 0.270]	[0.308, 0.416]	[0.298, 0.406]	[0.317, 0.426]
Knorm	[0.327, 0.436]	[0.493, 0.605]	[0.625, 0.730]	[0.709, 0.805]	[0.016, 0.056]	[0.023, 0.069]	[0.046, 0.105]	[0.097, 0.173]
RKV	[0.366, 0.477]	[0.395, 0.507]	[0.584, 0.692]	[0.656, 0.758]	[0.038, 0.093]	[0.054, 0.116]	[0.079, 0.150]	[0.105, 0.184]
SnapKV	[0.783, 0.868]	[0.805, 0.886]	[0.794, 0.877]	[0.794, 0.877]	[0.327, 0.436]	[0.366, 0.477]	[0.356, 0.466]	[0.356, 0.466]
StreamingLLM	[0.195, 0.291]	[0.337, 0.446]	[0.464, 0.576]	[0.636, 0.740]	[0.016, 0.056]	[0.016, 0.056]	[0.038, 0.093]	[0.114, 0.195]
DeepSeek-R1-Distill-Llama-8B								
Method	GSM8K				Math500			
Full	[0.646, 0.749]				[0.404, 0.517]			
ShadowKV	[0.454, 0.566]				[0.289, 0.395]			
H2O	[0.317, 0.426]	[0.473, 0.586]	[0.564, 0.673]	[0.554, 0.663]	[0.159, 0.249]	[0.260, 0.364]	[0.308, 0.416]	[0.308, 0.416]
Knorm	[0.000, 0.013]	[0.063, 0.128]	[0.150, 0.238]	[0.232, 0.333]	[0.000, 0.013]	[0.003, 0.029]	[0.009, 0.043]	[0.038, 0.093]
RKV	[0.031, 0.081]	[0.023, 0.069]	[0.105, 0.184]	[0.132, 0.217]	[0.016, 0.056]	[0.031, 0.081]	[0.009, 0.043]	[0.009, 0.043]
SnapKV	[0.667, 0.768]	[0.667, 0.768]	[0.688, 0.786]	[0.667, 0.768]	[0.366, 0.477]	[0.385, 0.497]	[0.356, 0.466]	[0.356, 0.466]
StreamingLLM	[0.038, 0.093]	[0.204, 0.302]	[0.337, 0.446]	[0.503, 0.615]	[0.016, 0.056]	[0.063, 0.128]	[0.168, 0.260]	[0.242, 0.344]
Method	CSQA				OBQA			
Full	[0.698, 0.796]				[0.794, 0.877]			
ShadowKV	[0.159, 0.249]				[0.260, 0.364]			
H2O	[0.424, 0.536]	[0.667, 0.768]	[0.677, 0.777]	[0.677, 0.777]	[0.424, 0.536]	[0.730, 0.823]	[0.783, 0.868]	[0.794, 0.877]
Knorm	[0.031, 0.081]	[0.232, 0.333]	[0.483, 0.596]	[0.605, 0.711]	[0.016, 0.056]	[0.223, 0.323]	[0.513, 0.625]	[0.646, 0.749]
RKV	[0.046, 0.105]	[0.079, 0.150]	[0.123, 0.206]	[0.298, 0.406]	[0.046, 0.105]	[0.046, 0.105]	[0.150, 0.238]	[0.270, 0.375]
SnapKV	[0.688, 0.786]	[0.677, 0.777]	[0.688, 0.786]	[0.677, 0.777]	[0.773, 0.859]	[0.783, 0.868]	[0.783, 0.868]	[0.762, 0.850]
StreamingLLM	[0.023, 0.069]	[0.105, 0.184]	[0.298, 0.406]	[0.444, 0.556]	[0.046, 0.105]	[0.114, 0.195]	[0.270, 0.375]	[0.464, 0.576]
Method	ReClor				DROP			
Full	[0.454, 0.566]				[0.105, 0.184]			
ShadowKV	[0.223, 0.323]				[0.063, 0.128]			

Continued on next page...

Table 14: Confidence intervals (continued)...

Budget	128	256	384	512	128	256	384	512
H2O	[0.016, 0.056]	[0.054, 0.116]	[0.186, 0.281]	[0.327, 0.436]	[0.038, 0.093]	[0.046, 0.105]	[0.071, 0.139]	[0.079, 0.150]
Knorm	[0.000, 0.013]	[0.000, 0.013]	[0.009, 0.043]	[0.071, 0.139]	[0.000, 0.013]	[0.003, 0.029]	[0.003, 0.029]	[0.031, 0.081]
RKV	[0.023, 0.069]	[0.016, 0.056]	[0.016, 0.056]	[0.079, 0.150]	[0.016, 0.056]	[0.016, 0.056]	[0.031, 0.081]	[0.046, 0.105]
SnapKV	[0.444, 0.556]	[0.054, 0.116]	[0.031, 0.081]	[0.031, 0.081]	[0.132, 0.217]	[0.114, 0.195]	[0.114, 0.195]	[0.123, 0.206]
StreamingLLM	[0.000, 0.013]	[0.000, 0.013]	[0.003, 0.029]	[0.038, 0.093]	[0.009, 0.043]	[0.009, 0.043]	[0.063, 0.128]	[0.097, 0.173]
Method	StrategyQA				FOLIO			
Full	[0.688, 0.786]				[0.414, 0.527]			
ShadowKV	[0.751, 0.841]				[0.279, 0.385]			
H2O	[0.204, 0.302]	[0.636, 0.740]	[0.719, 0.814]	[0.740, 0.832]	[0.046, 0.105]	[0.317, 0.426]	[0.356, 0.466]	[0.404, 0.517]
Knorm	[0.038, 0.093]	[0.308, 0.416]	[0.513, 0.625]	[0.646, 0.749]	[0.000, 0.013]	[0.016, 0.056]	[0.079, 0.150]	[0.168, 0.260]
RKV	[0.054, 0.116]	[0.298, 0.406]	[0.444, 0.556]	[0.574, 0.683]	[0.016, 0.056]	[0.054, 0.116]	[0.097, 0.173]	[0.214, 0.312]
SnapKV	[0.625, 0.730]	[0.605, 0.711]	[0.584, 0.692]	[0.625, 0.730]	[0.404, 0.517]	[0.395, 0.507]	[0.434, 0.546]	[0.404, 0.517]
StreamingLLM	[0.016, 0.056]	[0.079, 0.150]	[0.308, 0.416]	[0.503, 0.615]	[0.000, 0.013]	[0.003, 0.029]	[0.023, 0.069]	[0.063, 0.128]

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133