A PROBABILISTIC INTERPRETATION OF KV CACHE EVICTION

Anonymous authors
Paper under double-blind review

ABSTRACT

The premise and promise of KV (cache) eviction is simple: higher throughput can be achieved by evicting some entries from the KV cache, at a negligible cost to quality. This holds empirically for many existing methods, though most rely on creative heuristics for selecting which entries to drop. Despite recent advances, the problem of KV eviction has remained informal in the literature. This paper aims to properly formalize this problem through the lens of probabilistic reasoning and reveal what can be learned from this perspective. Concretely, we (1) formalize the problem of KV eviction and, unfortunately, prove that it is computationally hard, (2) show that by framing it probabilistically, KV eviction reduces to the problem of expectation estimation, which can be approximated through sampling, (3) show that through this probabilistic interpretation, correcting for evicted entries during decoding—a previously ignored problem—becomes feasible, and (4) reveal that existing methods in the literature are zero-variance biased estimators that can be easily adapted in order to enable decode time correction. In practice, we show that this probabilistic version of KV eviction coupled with decode time correction is more robust to different tasks compared to existing eviction methods and achieves competitive performance at the same compression budget.

1 INTRODUCTION

Key-Value (KV) cache compression poses a natural proposition: increase throughput and decrease memory footprint by sacrificing some precision in the KV cache. Popular methods for this utilize quantization (Hooper et al., 2024; Zandieh et al., 2026), compaction (Zweiger et al., 2026; Eyuboglu et al., 2025), eviction (Xiao et al., 2024; Li et al., 2024; Oren et al., 2024; Zhang et al., 2023), *inter alia*, to shrink down the size of the cache and speed up inference. Among these techniques, KV (cache) eviction presents the simplest solution: given a prompt and the KV cache associated with it, evict some entries of the cache at a (hopefully) negligible cost to generation quality. Despite this simplicity, the problem itself is quite challenging, and existing KV eviction strategies elaborate sophisticated heuristics in order to identify which entries to evict, performing surprisingly well in practice. Although these recent advances are indeed quite impressive, there is a distinct lack of formality in the literature. This paper aims to address this, and in its wake expose key insights on the structure of KV eviction.

As a first result, by formalizing KV eviction we reveal that the problem of KV eviction is unfortunately computationally hard. However, through a probabilistic interpretation of the attention mechanism, eviction can be viewed as a problem of expectation estimation, which is well studied in the statistical methods literature (Owen, 2013). We then reveal that the usual KV eviction setup ignores the distributional shift that occurs after eviction: because of missing entries, the distribution over all entries is distorted, and therefore so is the expectation. Thus, within this framing an expectation estimator (and KV eviction strategy) aims to achieve low error at decode time by either reducing its bias, variance, or both. Interestingly, most existing KV eviction methods are trivially subsumed within this framework.

The common approach in KV eviction is to score entries by using the attention weights and values, and then take the top-k entries according to these scores (Li et al., 2024; Zhang et al., 2023; Oren et al., 2024); this estimator is zero-variance as it is deterministic. However, it can have an arbitrarily large bias. We instead suggest that these scores should be viewed as unnormalized distributions on entries

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

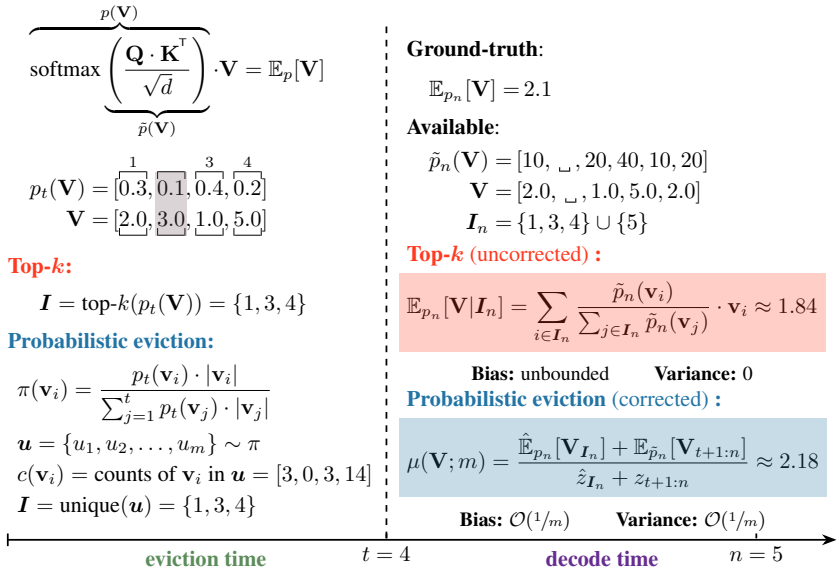


Figure 1: **Correction can attenuate the effects of eviction at decode time.** Evicting entries deforms the distribution and biases the expectation, as the normalizing constant is altered by the evicted entries. Existing eviction methods deterministically evict the top-k scored entries, which although zero-variance, may have arbitrarily large bias. Probabilistic eviction addresses this unintended deformation at an upper bounded error of $\mathcal{O}(m^{-1})$, where m is the number of samples.

which to evict, allowing us to later reuse them as a proposal distribution to correct for the distortion caused by eviction through self-normalized importance sampling. This is an attractive option because self-normalized importance sampling has reasonable upper bounds on bias and variance (Owen, 2013; Agapiou et al., 2017). Figure 1 summarizes our approach.

In summary, our contributions are as follows:

1. **We show that KV eviction is** NP-complete (Section 3);
2. **We formalize KV eviction as an expectation estimation problem** and show an asymptotically correct estimator for the decode time expectation with bounded error (Sections 4 and 5);
3. **We show the interplay of bias versus variance** in this estimator and how existing eviction methods are subsumed by this framework (Section 6 and Appendix D);
4. **We empirically validate our findings** by showing that coupling probabilistic eviction and decode time correction is not only competitive at the same compression budget but is more robust across different tasks (Section 7).

2 Preliminaries

We start by briefly introducing notation and reviewing the attention mechanism of Vaswani et al. (2017) as well as the concepts of KV caching and eviction. Throughout this paper, we denote random variables (RVs) as single upper case letters (e.g. X, Y), their values or scalars as single lower case (e.g. x, y), sets as bold (e.g. $\boldsymbol{X}, \boldsymbol{Y}$ for RVs or $\boldsymbol{x}, \boldsymbol{y}$ for values), and sequences or matrices as bold upright lower and upper case respectively (e.g. $\mathbf{x}, \mathbf{y}$ for sequences and vectors, and $\mathbf{X}, \mathbf{Y}$ for matrices).

Attention. Let $\mathbf{K}, \mathbf{Q}, \mathbf{V} \in \mathbb{R}^{n\times d}$ be the key, query and value embedding matrices for a particular attention head, where n is the token sequence length and d is the embedding dimension. Attention is computed as

$$\mathbf{O} := \text{softmax}\left(\frac{\mathbf{Q}\cdot\mathbf{K}^\intercal}{\sqrt{d}}\right)\cdot\mathbf{V}, \tag{1}$$

where the denominator inside the softmax is often omitted as it is mostly used as a regularizer during training (and here we shall do so as well). This computation ultimately yields a matrix $\mathbf{O} \in \mathbb{R}^{n\times d}$

corresponding to the attention values for each of the n entries. The key, query and value embeddings for the next layer are then computed as a function of $\mathbf{O}$.

KV caching. The intermediate softmax computation is $n \times n$ which can be prohibitively large at longer sequences. Because generation depends only on the next token attention values, it is therefore useful to, instead of fully computing the softmax matrix in Equation (1), compute only the last row of the softmax matrix and cache $\mathbf{K}$ and $\mathbf{V}$ at each attention head

$$\underset{1\times d}{\mathbf{o}} := \overbrace{\text{softmax}\Big(\underset{1\times d}{\mathbf{Q}_{-1}} \cdot \underset{d\times n}{\mathbf{K}^\intercal}\Big)}^{1\times n} \cdot \overbrace{\mathbf{V}}^{n\times d} = \text{softmax}(\overbrace{[\underset{1\times d}{\mathbf{k}_i} \cdot \underset{d\times 1}{\mathbf{q}_n}]_{i=1}^{n}}^{1\times n}) \cdot \overbrace{\mathbf{V}}^{n\times d}; \tag{2}$$

where here we annotate matrices and vectors with their dimensions for clarity. The cached keys and values $\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_n$ and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ are then reused in the next $n+1$ forward pass.

KV cache eviction. One popular strategy to further speed up computation is to further reduce the softmax computation in Equation (2) to a subset of entries. Let $\boldsymbol{I} \subseteq [1..n]$ be the set of indices of which entries to keep, where here we denote $[i..j]$ as the set of positive integers in $[i, j]$. The KV cache evicted attention of an attention head with kept entries $\boldsymbol{I}$ is given by

$$\mathbf{o}_{\boldsymbol{I}} := \text{softmax}\left([\mathbf{k}_i \cdot \mathbf{q}_n]_{i\in\boldsymbol{I}}\right) \cdot \mathbf{V}_{\boldsymbol{I}}, \tag{3}$$

where $\mathbf{V}_{\boldsymbol{I}}$ indicates indexing the rows of $\mathbf{V}$ with $\boldsymbol{I}$. The remaining entries not in $\boldsymbol{I}$ are evicted from the cache. This is KV cache eviction. The goal in eviction is to thus select an appropriate $\boldsymbol{I}$ such that downstream performance is (ideally) unaffected or only slightly impacted. We refer the reader to Appendix A for a discussion on related work.

Now that we understand the setting and the goal of KV (cache) eviction, we are ready to formalize and study KV eviction. We start by showing its hardness.

3 KV Cache Eviction is Hard

First and foremost we state the problem, its input and output.

Problem 1 (KVEviction)**.**
Input. Matrices $\mathbf{K}, \mathbf{Q}, \mathbf{V} \in \mathbb{Q}^{n\times d}$, error $\varepsilon \in \mathbb{Q}^d$, and compression ratio $r \in (0, 1)$.
Output. Whether there exists a set $\boldsymbol{I} \subset [1..n]$ such that $|\boldsymbol{I}| = \lfloor n \cdot r \rfloor$ and

$$\left|\text{softmax}\left([\mathbf{k}_i \cdot \mathbf{q}_n]_{i\in\boldsymbol{I}}\right) \cdot \mathbf{V}_{\boldsymbol{I}} - \text{softmax}\left([\mathbf{k}_i \cdot \mathbf{q}_n]_{i=1}^{n}\right) \cdot \mathbf{V}\right| \leq \varepsilon. \tag{4}$$

Our reduction is from the Partition problem.

Problem 2 (Partition)**.**
Input. A sequence of n integers $\mathbf{v} = (v_1, v_2, \ldots, v_n)$.
Output. Whether there exists a set $\boldsymbol{I} \subset [1..n]$ such that $|\boldsymbol{I}| = \lfloor \frac{n}{2} \rfloor$ and

$$\sum_{v\in\mathbf{v}_{\boldsymbol{I}}} v = \sum_{u\in\mathbf{v}_{\overline{\boldsymbol{I}}}} u,$$

where we here denote $\mathbf{v}_{\overline{\boldsymbol{I}}}$ to mean the values in $\mathbf{v}$ *not* indexed by $\boldsymbol{I}$.

Theorem 1. KVEviction is NP-complete.

The reduction and membership proof is stated in Appendix B. The attentive reader might recognize that KVEviction is in fact an *easier* problem compared to real-world KV eviction. In KV eviction, the goal is to preserve the attention head values at *decode* time—i.e. the *future* time steps *after* entries have been evicted and are no longer available to compute. Our results show that even preserving the attention head values at *eviction* time—when we have access to all the ground-truth values—is hard.

With this in mind, our goal now is to exploit the structure of KVEviction in order to find principled KV eviction approximations. We do so by a probabilistic interpretation of KV eviction, showing that through this lens we obtain a richer space of eviction strategies.

4 A PROBABILISTIC INTERPRETATION OF KV CACHE EVICTION

A natural yet crucial observation is that the softmax matrix in Equation (1) induces n distributions (one for each row of the matrix) over the n entries represented by each column. The resulting attention value for an attention head is thus a matrix containing the dot product of these probabilities with the values $\mathbf{V}$. This amounts to computing the expectation of $\mathbf{V}$ under a distribution $p_i(\mathbf{V})$ parameterized by $\mathbf{q}_i$ and $\mathbf{K}$. It is important to note that this distribution is not over the space of values $\mathbb{R}^{n\times d}$, but rather is a categorical distribution over the n entries (i.e. indices) that map to $\mathbf{V}$.

$$\text{softmax}\left(\mathbf{Q}\cdot\mathbf{K}^\top\right)\cdot\mathbf{V} = [\overbrace{\text{softmax}\left(\mathbf{q}_i\cdot\mathbf{K}^\top\right)}^{p_i(\mathbf{V})}]_{i=1}^{n}\cdot\mathbf{V} = \left[\sum_{j=1}^{n} p_i(\mathbf{V}=\mathbf{v}_j)\cdot\mathbf{v}_j\right]_{i=1}^{n} = [\mathbb{E}_{p_i}[\mathbf{V}]]_{i=1}^{n} \tag{5}$$

In short, attention is nothing more than computing expectations. When using a KV cache, this further reduces to simply computing a single expectation of a d-dimensional vector on the last row. This provides an interesting framework for KV eviction: Each entry now has a clear distribution associated to it and can—within the setting of KV eviction—be interpreted as the probability distribution of keeping that entry. The goal of KV eviction then becomes to find a subset $\boldsymbol{I}$ that estimates future expectations with least bias and variance.

Interestingly, existing eviction methods usually base their eviction decision on computing some score, either from $p_n(\mathbf{V})$ or $\mathbf{V}$, and then taking the top-k highest scored entries to keep (Devoto et al., 2024; Li et al., 2024; Zhang et al., 2023). This is a zero-variance yet clearly biased estimator (and in fact unboundedly biased), skewing the expectation even at eviction time.

A simple and easy solution to achieve unbiasedness at eviction time is to select $\boldsymbol{I}$ according to samples from $p_n(\mathbf{V})$. For a given number of samples m taken with replacement from $p_n(\mathbf{V})$, $\boldsymbol{I}$ can be chosen as the collection of unique entries in this set of samples. This provides an unbiased eviction estimator at eviction time. In fact, one can use any proposal distribution π in order to probabilistically evict from the cache and later adjust the expectation computation with importance sampling.

The choice of which proposal to use is open to the user as long as its support covers all entries. In fact, existing top-k score KV eviction methods are easily subsumed in this framework by interpreting these scores as an unnormalized proposal distribution $\tilde{\pi}$ (Appendix D). Indeed, given a score eviction strategy, there exists an infinite number of proposal distributions whose top-k also correspond to the same top-k of that strategy. Top-k eviction corresponds to computing the expectation conditioned on the k-maximum a posteriori (MAP) states as a proxy for the eviction time expectation, while probabilistic eviction—i.e. sampling from π and choosing all entries which have been sampled at least once—estimates the expectation through importance sampling.

So far we have seen that top-k is biased even at eviction time, while probabilistic eviction emerges as an unbiased eviction time alternative. However, this discussion ignores the future; the distribution at later steps will have missing probabilities which can potentially cause further error to future expectations. Thus, a natural question to ask ourselves is:

Goal. If we evict at time step t, can we correct Eq. (3) for the missing entries at time steps $> t$?

As we shall see in the next sections, we can make use of the proposal distribution and its samples used in probabilistic eviction to correct decoding.

5 CORRECTING FOR EVICTION

Given a sequence of length t, the task of KV eviction is usually to perform eviction on this sequence—i.e. at time step t—and then generate new tokens at future time steps $n > t$. We shall call time step t *eviction time* and any time step after that *decode time*. While at eviction time we have access to all previous distributions $[p_i(\mathbf{V})]_{i=1}^{t}$, at decode time $n > t$ we only have access to the kept entries of $p_n(\mathbf{V})$ (as well as cached keys $\mathbf{K}_{\boldsymbol{I}}$ and values $\mathbf{V}_{\boldsymbol{I}}$). The usual treatment in the literature is to ignore missing entries and compute attention as usual. However, this severely changes the distribution, as

this amounts to conditioning the distribution to restricting its support to only kept entries

$$\text{softmax}\left([\mathbf{k}_i \cdot \mathbf{q}_n]_{i \in \boldsymbol{I}_t^n}\right) \cdot \mathbf{V}_{\boldsymbol{I}_t^n} = \left[\frac{p_n(\mathbf{v}_i) \cdot [\![i \in \boldsymbol{I}_t^n]\!]}{\sum_{j \in \boldsymbol{I}_t^n} p_n(\mathbf{v}_j)}\right]_{i=1}^{n} \cdot \mathbf{V}_{\boldsymbol{I}_t^n} = p_n\left(\mathbf{V} | \boldsymbol{I}_t^n\right) \cdot \mathbf{V}_{\boldsymbol{I}_t^n} = \mathbb{E}_{p_n}\left[\mathbf{V} | \boldsymbol{I}_t^n\right], \tag{6}$$

where $\boldsymbol{I}_t^n := \boldsymbol{I} \cup [t+1..n]$—i.e. all the kept entries and all the entries generated after eviction time—, and $[\![\cdot]\!]$ denotes the Iverson bracket. This means that the contribution of each entry will be excessively boosted or diminished by the normalizing constant proportionally to the probability mass evicted. Furthermore, because $\mathbf{K}$, $\mathbf{V}$ and $\mathbf{Q}$ are all computed as a function of the attention values of the previous layer, any error—even if small—propagates and is amplified to all following layers and future generations.

To correct for this, we utilize self-normalized importance sampling to adjust the probabilities at decode time (Kloek & van Dijk, 1978; Geweke, 1989; Owen, 2013). This requires we exploit the probabilistic interpretation introduced in Section 4. In summary, given a proposal distribution π at eviction time t, we sample entries from π, record their counts, and evict any entries that have not been sampled. At decode time, we then use π and the sampled counts to correct for eviction through self-normalized importance sampling. We now describe this process in detail and show that the corrected expectation estimate is asymptotically correct. To do so, let us first properly define the tools we are going to use.

Definition 1 (Self-normalized importance sampling)**.** Let $\tilde{p}(X)$ be an unnormalized probability distribution over an RV X that can take n values, p its normalized distribution, and $\pi(X)$ a proposal distribution also over X. Given m samples from π, the function $c(x)$ maps the value x to the number of times x has been sampled from π. The self-normalized importance sampling estimator for the expectation $\mathbb{E}_p[X]$ is

$$\hat{\mu}(X; \tilde{p}, \pi, m) := \frac{1}{m} \sum_{i=1}^{n} \frac{c(x_i) \frac{\tilde{p}(x_i)}{\pi(x_i)}}{\frac{1}{m} \sum_{j=1}^{n} c(x_j) \frac{\tilde{p}(x_j)}{\pi(x_j)}} x_i \overset{m \to \infty}{=} \mathbb{E}_\pi\left[\frac{p(X)}{\pi(X)} \cdot X\right] = \mathbb{E}_p[X]. \tag{7}$$

Our goal is to estimate the expectation at decode time n. In order to do so, we slightly massage the expectation into a more appealing expression

$$\begin{aligned}
\mathbb{E}_{p_n}[\mathbf{V}] &= \sum_{i=1}^{n} p_n(\mathbf{v}_i) \cdot \mathbf{v}_i = \frac{\sum_{i=1}^{t} \tilde{p}_n(\mathbf{v}_i) \cdot \mathbf{v}_i + \sum_{i=t+1}^{n} \tilde{p}_n(\mathbf{v}_i) \cdot \mathbf{v}_i}{\sum_{j=1}^{n} \tilde{p}_n(\mathbf{v}_i)} \\
&= \frac{\mathbb{E}_{\tilde{p}_n}[\mathbf{V}_{1:t}] + \mathbb{E}_{\tilde{p}_n}[\mathbf{V}_{t+1:n}]}{z} = \mathbb{E}_{p_n^{(1:t)}}[\mathbf{V}] \cdot \frac{z_{1:t}}{z_{1:t} + z_{t+1:n}} + \mathbb{E}_{p_n^{(t+1:n)}}[\mathbf{V}] \cdot \frac{z_{t+1:n}}{z_{1:t} + z_{t+1:n}} \\
&= \boxed{\mathbb{E}_{p_n^{(1:t)}}[\mathbf{V}] \cdot p(T \leq t) + \mathbb{E}_{p_n^{(t+1:n)}}[\mathbf{V}] \cdot p(T > t).}
\end{aligned} \tag{8}$$

We use the superscript to denote the distribution restricted to the support specified by the interval and then renormalized accordingly $p^{(i:k)}(X) := \left[p(x_j) / \sum_{l=i}^{k} p(x_l)\right]_{j=i}^{k}$. Similarly, $\mathbf{V}_{i:k}$ is used to denote a restriction on the values of $\mathbf{V}$ to $\{\mathbf{v}_j\}_{j=i}^{k}$. Lastly, we use z to denote the normalizing constant and correspondingly $z_{i:k} := \sum_{j=i}^{k} \tilde{p}_n(\mathbf{v}_j)$ to mean the normalizing constant for that interval.

The final expression in Equation (8) tells us that we may decompose the expectation into two local expectations: *before* and *after* eviction, as long as we weigh them appropriately according to $p(T \leq t)$ and $p(T > t)$. The local expectation after eviction $\mathbb{E}_{p_n^{(t+1:n)}}[\mathbf{V}]$ can be computed exactly: all entries are available. However, the local expectation before eviction contains missing entries and thus needs to be approximated. Indeed, both $p(T \leq t)$ and $p(T > t)$ also need to be approximated as their denominator is the normalizing constant z. To do so, we define the following estimator.

Definition 2 (Attention estimator)**.** The estimator for corrected attention is given by

$$\mu_n(\mathbf{V}; m) := \hat{\mu}(\mathbf{V}; \tilde{p}_n^{(1:t)}, \pi, m) \cdot p(T \leq t) + \mathbb{E}_{p_n^{(t+1:n)}}[\mathbf{V}] \cdot p(T > t), \tag{9}$$

where $\hat{\mu}$ is a self-normalized importance sampling estimator that uses the unnormalized distribution $\tilde{p}_n^{(1:t)}$ as the target distribution and π as the proposal. The probabilities $p(T \leq t)$ and $p(T > t)$ are approximated by aggregating the importance weights from μ_n: $\hat{z}_{1:t} = \frac{1}{m} \sum_{j=1}^{t} c(\mathbf{v}_j) \frac{\tilde{p}_n(\mathbf{v}_j)}{\pi(\mathbf{v}_j)}$.

Algorithm 1 IMPORTANCEATTENTION

Input KV cache $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{n \times d}$, sample counts $[c(\mathbf{v}_i)]_{i \in \boldsymbol{I}}$, proposal π, current query $\mathbf{q}_n \in \mathbb{R}^d$.
Output Corrected attention $\mathbf{o} \in \mathbb{R}^d$

1: $\tilde{p}_n \leftarrow \exp\left(\mathbf{q}_n \cdot \mathbf{K}^\intercal\right)$ ▷ Unnormalized probabilities of non-evicted entries
2: $w_i \leftarrow c(\mathbf{v}_i)/(m \cdot \pi(\mathbf{v}_i)), \forall i \in [1..t]$ ▷ Weights to be applied to the unnormalized probabilities
3: $\hat{z}_{1:t} \leftarrow \sum_{i=1}^{t} w_i \cdot \tilde{p}_n(\mathbf{v}_i)$ ▷ Normalizing constant for the eviction time entries
4: $z_{t+1:n} \leftarrow \sum_{i=t+1}^{n} \tilde{p}_n(\mathbf{v}_i)$ ▷ Normalizing constant for the decode time entries
5: $\hat{\mu}_{1:t} \leftarrow \sum_{i=1}^{t} w_i \cdot \tilde{p}_n(\mathbf{v}_i) \cdot \mathbf{v}_i$ ▷ Estimated unnormalized expectation for eviction time
6: $\mu_{t+1:n} \leftarrow \sum_{i=t+1}^{n} \tilde{p}_n(\mathbf{v}_i) \cdot \mathbf{v}_i$ ▷ Exact unnormalized expectation for decode time
7: $\mathbf{o} \leftarrow \frac{\hat{\mu}_{1:t} + \mu_{t+1:n}}{\hat{z}_{1:t} + z_{t+1:n}}$ ▷ Corrected attention value after normalization
8: **return** $\mathbf{o}$

This estimator is asymptotically correct w.r.t. the desired expectation estimand (see Appendix C).

Proposition 1. If $\text{supp}(\pi) \supseteq \text{supp}(p_n)$, then $\lim_{m \to \infty} \mu_n(\mathbf{V}; m) = \mathbb{E}_{p_n}[\mathbf{V}]$.

Note that although $\mu_n(\mathbf{V}; m)$ is still biased for a fixed number of samples m, both bias and (mean squared) error are upper bounded by $O(1/m)$ (Agapiou et al., 2017, Theorem 2.1), meaning more samples reduce this error significantly. This bias can be further reduced by applying any bias-reducing method for self-normalized importance sampling to $\mu_n(\mathbf{V}; m)$ (Cardoso et al., 2022). In contrast to this bound, existing KV eviction methods deterministically keep the top-k entries according to some score on either $p_t(\mathbf{V})$ or $\mathbf{V}$ and thus are zero-variance yet unboundedly biased (Li et al., 2024; Zhang et al., 2023; Oren et al., 2024).

Equipped with Definition 2 and Proposition 1, we can now explicitly construct the correction algorithm for decode time. Algorithm 1 shows how to compute the corrected attention value for a given attention head. Note that here we simplify the algorithm and notation due to space constraints, but one must be careful when computing the corrected probabilities: an entry i is evicted iff $c(\mathbf{v}_i) = 0$, meaning that $\tilde{p}_n(\mathbf{v}_i)$ will be undefined. Any entry which is missing due to eviction should be properly ignored in the computation.

Ultimately, the pipeline is as follows: at eviction time t, (1) construct a proposal distribution $\pi^{(h)}$ for each attention head h, and (2) sample counts $[c^{(h)}(\mathbf{v}_i)]_{i=1}^{t}$. At decode time n, (3) compute attention following Algorithm 1.

So far, our contributions have either been theoretical or philosophical. In the next section, we empirically validate probabilistic eviction and correction, studying the role of bias and variance in KV eviction.

6 BIAS VS VARIANCE

Although bias and variance share the same asymptotic upper bounds in self-normalized importance sampling, in practice they may present distinct behaviors (Agapiou et al., 2017). In this section, we show that by decreasing the variance at the cost of more bias, we are able to achieve lower error under certain conditions. To do so, we introduce a temperature scaling parameter $\tau \in [0, \infty)$ to the importance weights. We then replace Line 2 in Algorithm 1 with

$$w_i \leftarrow \exp\left(\frac{1}{\tau} \cdot \log\left(\frac{c(\mathbf{v}_i)}{m \cdot \pi(\mathbf{v}_i)}\right)\right), \forall i \in [1..t]. \tag{10}$$

When $\tau = 1$, Equation (10) reduces to the same expression as Line 2; when $\tau \to \infty$, it is equivalent to not applying correction since w_i approaches 1. This reduces variance, since the stochasticity comes from the samples of the proposal distribution, which can be amplified by the correction. By suppressing correction, variance is reduced.

Our goal is to show the impact of bias and variance in terms of the mean absolute error (MAE) between the estimated expectation and the ground-truth expectation. Given a prompt $\mathbf{x}$ and a first response token y, we compute the KV cache $(\mathbf{K}, \mathbf{V})$ on $\mathbf{x}$ and apply probabilistic eviction using

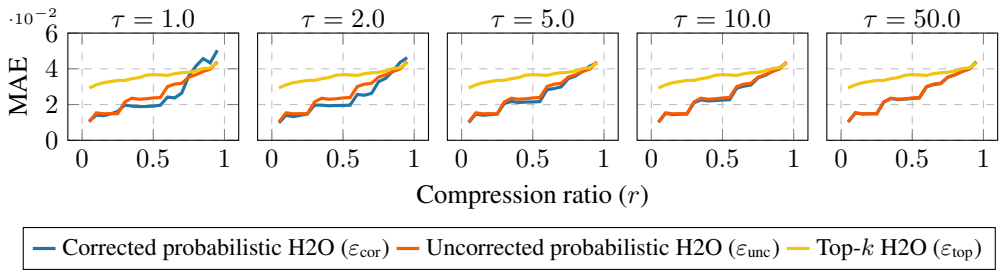


Figure 2: **KV eviction is a game of trading bias for variance and vice versa.** The plot shows mean absolute error (y-axis) of the attention values for the first token after an evicted prompt on `Llama3` averaged across all heads (see Appendix E for details) for different compression ratios (x-axis). Bias and variance are low at lower r, since m (number of samples) is high. As r increases, m decreases, causing variance to increase more than bias. Higher importance weight temperatures τ decrease the variance, causing the error to decrease at extremely high r's.

a proposal π constructed from the scores of the H2O eviction method (Zhang et al., 2023) (see Appendix D for details). This results in an evicted KV cache $(\mathbf{K}_{\boldsymbol{I}}, \mathbf{V}_{\boldsymbol{I}})$. We then compute the average error between the uncorrected $\mathbf{o}_{\boldsymbol{I}}$ and ground-truth $\mathbf{o}$ attention values ε_{unc}, and between the corrected $\mu_n(\mathbf{V}; m)$ and ground-truth attention values ε_{cor} across all h attention heads

$$\varepsilon_{\text{unc}} := \frac{1}{h} \cdot \sum_{i=1}^{h} \left| \mathbf{o}^{(i)} - \mathbf{o}_{\boldsymbol{I}}^{(i)} \right|, \quad \varepsilon_{\text{cor}} := \frac{1}{h} \cdot \sum_{i=1}^{h} \left| \mathbf{o}^{(i)} - \mu_n^{(i)}(\mathbf{V}; m) \right|. \tag{11}$$

Finally, we compute the error ε_{top} between the ground-truth and H2O's estimation of the attention.

In order to provide a fair comparison of ε_{cor} and ε_{unc} against ε_{top}, we constrain the compression ratio to be the same for every attention head. To do so, for each attention head, we select a number of samples that achieves, on average, the desired compression ratio. In Appendix F, we discuss how this problem relates to the classical coupon collector problem and provide a tractable approximation with negligible error in practice.

Figure 2 shows the average errors across compression ratios $r \in [0, 1]$ and temperatures $\tau \in \{1, 2, 5, 10, 50\}$ for `Llama3.2-3B` (Grattafiori et al., 2024); prompt $\mathbf{x}$ and first token y are described in Appendix E. Lower compression ratio r implies a higher number of samples m. When m is high, bias and variance are low; when m is extremely low, variance is very high, causing ε_{cor} to be much higher than ε_{unc}. This is where temperature scaling provides an improvement by reducing the variance at a cost to bias. As the middle ranges of r show, increasing bias when variance is already low harms estimation quality. The ε_{unc} curve does not match with ε_{top} because eviction is done stochastically, while H2O greedily evicts the top-k. This shows that even probabilistic eviction by itself allows for less error. As r approaches one, the probabilistic choices all collapse to the top-k, which is when ε_{unc} and ε_{top} meet.

In this section, we showed how correction allows for a trade-off between bias and variance. However, in practice bias plays a much larger role in attention error. This motivates the shift from low-variance high-bias to higher-variance low-bias estimators. As we shall see in the next section, corrected probabilistic eviction tends to produce more robust estimators compared to existing zero-variance unbounded bias KV eviction strategies due to reduced bias.

7 Importance Attention in Practice

In this section, we show that zero-variance estimators suffer from a heuristic misalignment, where the fixed bias introduced by these heuristics can cause catastrophic failures in certain tasks. To do so, we compare probabilistic eviction against four popular existing KV eviction methods: StreamingLLM (Xiao et al., 2024), SnapKV (Li et al., 2024), TOVA (Oren et al., 2024) and H2O (Zhang et al., 2023). All of these methods are implemented in KVPress, which we use for evaluation and our own implementation (Devoto et al., 2025). We evaluate `Llama3.2-3B` (Grattafiori et al., 2024) and

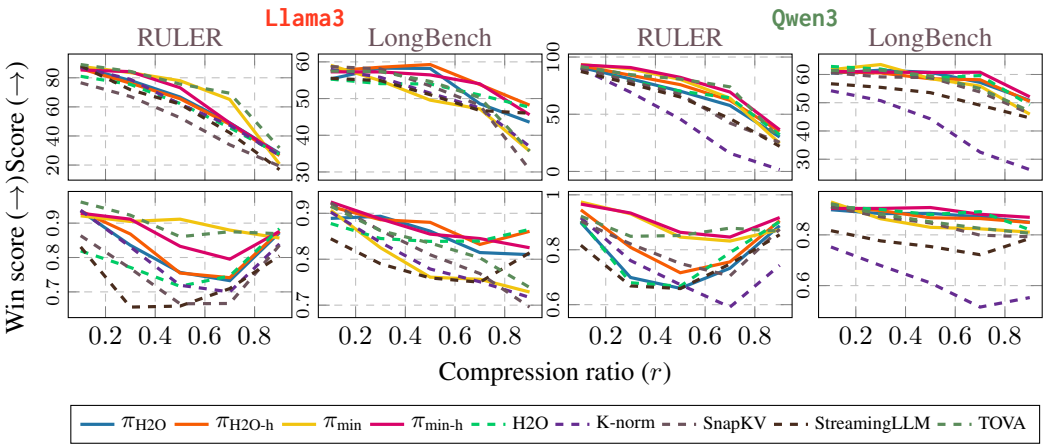


Figure 3: **Probabilistic eviction achieves competitive performance (top) and are more robust for different tasks.** Probabilistic eviction with correction performs best at lower to middle ranges, when there are enough samples to achieve a lower variance and bias. Solid lines show our approach, dashed lines show existing eviction methods. Proposal $\pi_{\text{min-h}}$ achieves best ranking across datasets and models, showing more robustness when minimizing variance and lowering bias.

Qwen3-4B (Yang et al., 2025) on two standard KV eviction benchmark datasets: LongBench (Bai et al., 2024) and RULER (Hsieh et al., 2024). Due to compute constraints, we evaluate RULER on a subset of 130 examples and on all instances of the HotpotQA, QASPER and TriviaQA splits of LongBench whose context prompt is shorter than 3,000 tokens.

To evaluate probabilistic eviction, we propose four different proposal distributions: π_{min}, $\pi_{\text{min-h}}$, π_{H2O} and $\pi_{\text{H2O-h}}$ which we formally define in Appendix H and provide proper probabilistic semantics. Intuitively, π_{min} computes the minimum variance proposal distribution for the last query at eviction time t, π_{H2O} computes a probabilistic version of H2O where scores are normalized and used as the proposal distribution, $\pi_{\text{min-h}}$ computes the minimum-variance proposal for a distribution where all previous time steps are marginalized and weighted by a harmonic prior h_j, and $\pi_{\text{H2O-h}}$ computes H2O with a harmonic prior h_j.

In these experiments, we constrain the compression ratio at the global level: given a compression ratio r, we choose a number of samples m such that, on average, the aggregated compression ratio of all KV caches is r. This is achieved by a similar mechanism to the one for locally constraining each attention head to a specific compression ratio. Section F.1 shows how this can be done and Section F.2 provides empirical evidence that the error between the achieved compression ratio and target compression ratio is, in practice, insignificant. Although the overall compression ratio is fixed, every attention head will adaptively choose a suitable compression ratio depending on its distribution. This allows for attention heads that require a lower compression ratio to automatically consume the budget of heads that do not. Interestingly, this naturally results in compression ratios that follow a pattern of lower compression at lower layers, and higher at higher layers (see Section I.3 and Figure 8). This is in agreement with the findings of Cai et al. (2025), which suggest that allocating more budget to lower layers and less to higher is preferable.

Figure 3 (top) shows the average total score in LongBench and RULER for the four previously mentioned proposals and five top-k existing eviction methods. This is computed by averaging the score at each split, weighing them by the fraction of examples in that split and then adding all of these weighted scores together. The average score of each split is given by a scorer function suited to that split $f_{\text{split}} : \mathcal{L}^* \to [0, 1]$ that takes in the text generated by the (KV evicted) LLM and attributes a number in $[0, 1]$. For example, HotpotQA uses F1 as the scorer function, while the NIAH (needle-in-a-haystack) split of RULER uses exact string matching. Results show that probabilistic eviction and correction achieve better performance or are at least as competitive as the state-of-the-art.

Because these average total scores aggregate over many tasks, they can hide the behavior of eviction methods for certain tasks. For example, on Llama3 StreamingLLM ranks first on the QuestionAnswer split, yet dead last in MultiKey-NIAH (see Figure 4). To account for these catastrophic failures, we

measure the pairwise win scores of each method. Specifically, we assign a rank from 0 to $n-1$ to each method based on how well it performs compared to the other $n-1$ methods. The method with the lowest score gets 0 points, while the method with the highest score gets $n-1$ points. If multiple methods achieve the same score, they receive the same number of points as if they won against tied opponents. Appendix J justifies this design choice; in short, the goal of this choice is to explicitly punish catastrophic failures. Ranking is repeated for all the examples in the task, after which the points are normalized by the number of examples in the task to ensure all tasks contribute equally. We refer to the normalized average rank across tasks as the *win score*.

Figure 3 (bottom) shows the win score for each proposal and eviction method. Notably, $\pi_{\text{min-h}}$ consistently ranks at the top, showing that it is most robust compared to other methods. Probabilistic eviction tends to do better at low to middle ranges, as there are enough samples to decrease the bias and variance. However, even at higher compression it performs competitively against other eviction methods. Note that when the compression ratio is extremely high (i.e. close to 1.0), all methods do equally poorly, as most of the prompt has been evicted. This is shown as most, if not all, curves increasing in Figure 3 (bottom), as ties are also rewarded in this scoring. Appendix K shows more quantitative results on robustness: we show that our eviction method tends to be much more performant compared to other eviction methods even when not rewarding ties (see Table 3).

To better underscore how much our probabilistic eviction method outperforms existing heuristics, we measure the overall ELO rating—together with 95% confidence bounds—across all compression ratios and methods on LongBench using `Llama3` (again, normalizing by the number of examples in each task to ensure equal contribution across tasks). Table 1 shows that $\pi_{\text{H2O-h}}$ outperforms competitors and that crucially their lower confidence bound is significantly above average scores of all the non-probabilistic methods. In fact, its lower bound is above the upper confidence bounds of all but one of the non-probabilistic competitors. The same can be said of the lower bounds of all proposals but π_{min}: these lower bounds are above all but one of the non-probabilistic lower bounds. See Table 4 in Appendix K for the non-aggregated ELO scores (and their confidence bounds) across each of the compression ratios.

Table 1: **Aggregated ELO ranking on LongBench with `Llama3`.** Proposal $\pi_{\text{H2O-h}}$ dominates over all other methods (especially non-probabilistic eviction strategies).

Method	ELO	[LCB, UCB]
$\pi_{\text{min-h}}$	1016.7	[1009.4, 1023.8]
$\pi_{\text{H2O-h}}$	**1026.2**	**[1017.2, 1033.5]**
π_{H2O}	1014.7	[1005.8, 1023.0]
π_{min}	977.7	[969.6, 986.2]
H2O	1015.6	[1006.9, 1023.7]
StreamingLLM	994.9	[984.4, 1004.8]
TOVA	989.1	[981.3, 996.5]
SnapKV	983.5	[975.9, 991.7]
K-norm	981.7	[972.4, 990.5]

Besides evaluating performance (degradation), it is also useful to measure the overhead necessary to (1) compute the extra information needed for correction and (2) execute Algorithm 1. Appendix L shows how our method requires negligible amount of time to probabilistically select which entries to evict and a very small amount of overhead to compute IMPORTANCEATTENTION.

8 CONCLUSION

The goal of this paper was to formalize KV cache eviction and in doing so find principled KV eviction estimators. As a result, we showed that KV eviction is NP-complete, and revealed that through a probabilistic interpretation KV eviction is nothing more than an expectation estimation problem. In our quest to characterize this, we found that existing KV eviction methods are zero-variance yet unbounded bias estimators and that a good KV eviction strategy should reduce bias and variance in order to best approximate future expectations. To this end, we proposed a self-normalized importance sampling estimator with reasonable upper bounds on both bias and variance. We then empirically showed that this class of estimators achieves competitive performance against state-of-the-art eviction methods yet are more robust at different tasks. This paper opens up an exciting and promising avenue to developing new KV eviction estimators. We hope that framing KV eviction as an estimation problem and connecting the field of statistical methods with KV cache eviction generates fruitful new research not only within KV eviction but also across other KV cache compression techniques.

REFERENCES

S. Agapiou, O. Papaspiliopoulos, D. Sanz-Alonso, and A. M. Stuart. Importance Sampling: Intrinsic Dimension and Computational Cost. *Statistical Science*, 32(3):405 – 431, 2017. doi: 10.1214/17-STS611. URL https://doi.org/10.1214/17-STS611.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. Longbench: A bilingual, multitask benchmark for long context understanding, 2024. URL https://arxiv.org/abs/2308.14508.

Arthur S. Bianchessi, Yasmin C. Aguirre, Rodrigo C. Barros, and Lucas S. Kupssinskü. Bayesian attention mechanism: A probabilistic framework for positional encoding and context length extrapolation. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=dXJB9O8fLd.

Richard P. Brent and Paul Zimmermann. *Modern Computer Arithmetic*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2010.

Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Yucheng Li, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Junjie Hu, and Wen Xiao. PyramidKV: Dynamic KV cache compression based on pyramidal information funneling. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=ayi7qezU87.

Gabriel Cardoso, Sergey Samsonov, Achille Thin, Eric Moulines, and Jimmy Olsson. Br-snis: Bias reduced self-normalized importance sampling. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 716–729. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/04bd683d5428d91c5fbb5a7d2c27064d-Paper-Conference.pdf.

Alex Chen, Renato Geh, Aditya Grover, Guy Van den Broeck, and Daniel Israel. The pitfalls of kv cache compression. *arXiv preprint arXiv:2510.00231*, 2025.

Alessio Devoto, Yu Zhao, Simone Scardapane, and Pasquale Minervini. A simple and effective L_2 norm-based strategy for KV cache compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18476–18499, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1027. URL https://aclanthology.org/2024.emnlp-main.1027/.

Alessio Devoto, Maximilian Jeblick, and Simon Jégou. Expected attention: Kv cache compression by estimating attention from future queries distribution. *arXiv preprint arXiv:2510.00636*, 2025. URL https://arxiv.org/abs/2510.00636.

Paul Erdős and Alfréd Rényi. On a classical problem of probability theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézetének Kőzleményei*, 1961. URL https://www.renyi.hu/~p_erdos/1961-09.pdf.

Sabri Eyuboglu, Ryan Ehrlich, Simran Arora, Neel Guha, Dylan Zinsley, Emily Liu, Will Tennien, Atri Rudra, James Zou, Azalia Mirhoseini, and Christopher Re. Cartridges: Lightweight and general-purpose long context representations via self-study, 2025. URL https://arxiv.org/abs/2506.06266.

Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S Kevin Zhou. Ada-KV: Optimizing KV cache eviction by adaptive budget allocation for efficient LLM inference. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026. URL https://openreview.net/forum?id=tcisuhGsQZ.

Philippe Flajolet, Danièle Gardy, and Loÿs Thimonier. Birthday paradox, coupon collectors, caching algorithms and self-organizing search. *Discrete Applied Mathematics*, 39(3):207–229, 1992. ISSN 0166-218X. doi: https://doi.org/10.1016/0166-218X(92)90177-C. URL https://www.sciencedirect.com/science/article/pii/0166218X9290177C.

John Geweke. Bayesian inference in econometric models using monte carlo integration. *Econometrica*, 57(6):1317–1339, 1989. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913710.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee,

Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 1270–1303. Curran Associates, Inc., 2024. doi: 10.52202/079017-0040. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/028fcbcf85435d39a40c4d61b42c99a4-Paper-Conference.pdf.

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

T. Kloek and H. K. van Dijk. Bayesian estimates of equation system parameters: An application of integration by monte carlo. *Econometrica*, 46(1):1–19, 1978. ISSN 00129682, 14680262. URL http://www.jstor.org/stable/1913641.

Yuhong Li, Yingbing Huang, Bowen Yang, Bharat Venkitesh, Acyr Locatelli, Hanchen Ye, Tianle Cai, Patrick Lewis, and Deming Chen. Snapkv: Llm knows what you are looking for before generation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 22947–22970. Curran Associates, Inc., 2024. doi: 10.52202/079017-0722. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/28ab418242603e0f7323e54185d19bde-Paper-Conference.pdf.

Matanel Oren, Michael Hassid, Nir Yarden, Yossi Adi, and Roy Schwartz. Transformers are multi-state RNNs. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 18724–18741, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.1043. URL https://aclanthology.org/2024.emnlp-main.1043/.

Art B. Owen. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

Ryan Singh and Christopher L. Buckley. Attention: Marginal probability is all you need?, 2023. URL https://arxiv.org/abs/2304.04556.

Roberto Tauraso. Expected number of non-empty boxes. Mathematics Stack Exchange, 2016. URL https://math.stackexchange.com/q/1950447. Version 2016-10-02 and visited on 2026-09-25. Author profile: https://math.stackexchange.com/users/299698/robert-z.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 21875–21895, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5e5fd18f863cbe6d8ae392a93fd271c9-Paper-Conference.pdf.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tO3ASKZlok.

Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 34661–34710. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/6ceefa7b15572587b78ecfcebb2827f8-Paper-Conference.pdf.

Adam Zweiger, Xinghong Fu, Han Guo, and Yoon Kim. Fast kv compaction via attention matching, 2026. URL https://arxiv.org/abs/2602.16284.

A Related Work

KV eviction has received significant interest for the purpose of speeding up inference in large language models. Following the taxonomy given by Chen et al. (2025), we categorize KV eviction methods as position-based, attention-based, embedding-based, and hybrid. The most notable examples of each respectively are StreamingLLM (Xiao et al., 2024), TOVA (Oren et al., 2024), K-norm (Devoto

et al., 2024), and SnapKV (Li et al., 2024). All these methods can be framed as top-k score eviction methods with zero variance and unbounded bias. The problem of correcting for the distortion caused by eviction has not been, as far as we know, properly addressed, although it has been recognized in the literature: Zhang et al. (2023) acknowledge the issue in passing, while in contemporary work, Zweiger et al. (2026) identify this problem but only as a motivation for KV cache compaction, a more compute-intensive strategy for KV cache compression. The probabilistic interpretation of attention as expectation has previously appeared in the literature as a Bayesian perspective to attention (Singh & Buckley, 2023; Bianchessi et al., 2026), although with no connection to KV eviction.

B HARDNESS

Theorem 1. KVEVICTION is NP-complete.

Proof. We first show hardness. We shall reduce from the PARTITION problem. Assume n is even (without loss of generality); set $d = 1$, $r = \frac{1}{2}$, $\varepsilon = 0$ and $\mathbf{K} = \mathbf{Q} = \mathbf{1}_{n\times d}$, where here we denote $\mathbf{1}_{m\times n}$ as the all-ones matrix in $\mathbb{Q}^{m\times n}$, and similarly $\mathbf{1}_n$ as the all-one vector in $\mathbb{Q}^n$. Thus, the problem boils down to deciding whether

$$\text{softmax}\left(\mathbf{1}_{|\boldsymbol{I}|}\right) \cdot \mathbf{V}_{\boldsymbol{I}} = \text{softmax}\left(\mathbf{1}_n\right) \cdot \mathbf{V}.$$

Note that the softmax terms on the left and on the right are equivalent to uniform distributions over $|\boldsymbol{I}|$ and n elements respectively. Thus, we may rewrite the above equation (after expanding the matrix multiplications) as

$$\frac{1}{\lfloor \frac{n}{2} \rfloor} \cdot \sum_{v \in \mathbf{V}_{\boldsymbol{I}}} v = \frac{1}{n} \cdot \sum_{u \in \mathbf{V}} u.$$

From the previous equation, it follows immediately that

$$\sum_{v \in \mathbf{V}_{\boldsymbol{I}}} v = \sum_{u \in \mathbf{V}_{\overline{\boldsymbol{I}}}} u.$$

We have thus shown that any instance of KVEVICTION is at least as hard as any instance of PARTITION. Since PARTITION is NP-hard, KVEVICTION is also NP-hard.

We now show membership in NP. The main issue is computing the softmax function in a Turing machine. We invoke Brent & Zimmermann (2010), where the authors show how to compute exponents and division up to arbitrary precision in time polynomial in the number of bits. In our case, the number of bits needed is directly specified by ε, meaning that one only needs to compute the softmax up to the precision of parameter ε. Thus, it suffices to define the verifier to be Equation (4), which can be done in time polynomial in the precision of ε, n and d. Therefore KVEVICTION is in NP and so is NP-complete. □

C CORRECTNESS

Proposition 1. If $\text{supp}(\pi) \supseteq \text{supp}(p_n)$, then $\lim_{m\to\infty} \mu_n(\mathbf{V}; m) = \mathbb{E}_{p_n}[\mathbf{V}]$.

Proof. We want to show that

$$\lim_{m\to\infty} \hat{\mu}\left(\mathbf{V}; \tilde{p}_n, \pi, m\right) \cdot \frac{\hat{z}_{1:t}}{\hat{z}_{1:t} + z_{t+1:n}} + \mathbb{E}_{p_n^{(t+1:n)}}\left[\mathbf{V}\right] \cdot \frac{z_{t+1:n}}{\hat{z}_{1:t} + z_{t+1:n}} = \mathbb{E}_{p_n}\left[\mathbf{V}\right]. \tag{12}$$

The first thing to note is that both $\mathbb{E}_{p_n^{(t+1:n)}}\left[\mathbf{V}\right]$ and $z_{t+1:n}$ can be computed exactly as we have access to $[\tilde{p}_n(\mathbf{v}_i)]_{i=t+1}^n$. Thus, we need to show that both (i) $\hat{\mu}\left(\mathbf{V}; \tilde{p}_n, \pi, m\right)$ and (ii) $\hat{z}_{1:t}$ correctly compute the desired estimands in expectation.

We first show (i)

$$\lim_{m\to\infty}\hat{\mu}\left(\mathbf{V};\tilde{p}_n,\pi,m\right)=\lim_{m\to\infty}\frac{\frac{1}{m}\cdot\sum_{i=1}^{m}\frac{\tilde{p}_n(\mathbf{v}^{(i)})}{\pi(\mathbf{v}^{(i)})}\cdot\mathbf{v}^{(i)}}{\frac{1}{m}\cdot\sum_{i=1}^{m}\frac{\tilde{p}_n(\mathbf{v}^{(i)})}{\pi(\mathbf{v}^{(i)})}}=\lim_{m\to\infty}\frac{\frac{1}{m}\cdot\sum_{i=1}^{t}c(\mathbf{v}_i)\cdot\frac{\tilde{p}_n(\mathbf{v}_i)}{\pi(\mathbf{v}_i)}\cdot\mathbf{v}_i}{\frac{1}{m}\cdot\sum_{i=1}^{t}c(\mathbf{v}_i)\cdot\frac{\tilde{p}_n(\mathbf{v}_i)}{\pi(\mathbf{v}_i)}} \tag{13}$$

$$=\frac{\sum_{i=1}^{t}\pi(\mathbf{v}_i)\cdot\frac{\tilde{p}_n(\mathbf{v}_i)}{\pi(\mathbf{v}_i)}\cdot\mathbf{v}_i}{\sum_{i=1}^{t}\pi(\mathbf{v}_i)\cdot\frac{\tilde{p}_n(\mathbf{v}_i)}{\pi(\mathbf{v}_i)}}=\frac{\mathbb{E}_{p_n^{(1:t)}}\left[\mathbf{V}\right]\cdot z_{1:t}}{z_{1:t}}=\mathbb{E}_{p_n^{(1:t)}}\left[\mathbf{V}\right], \tag{14}$$

where the superscript $\mathbf{v}^{(i)}$ indicates the i-th sampled entry from π and the step from (13) to (14) is due to our assumption of $\text{supp}(\pi)\supseteq\text{supp}(p_n)$.

The same applies to (ii)

$$\lim_{m\to\infty}\hat{z}_{1:t}=\lim_{m\to\infty}\frac{1}{m}\cdot\sum_{i=1}^{m}\frac{\tilde{p}_n(\mathbf{v}^{(i)})}{\pi(\mathbf{v}^{(i)})}=\lim_{m\to\infty}\frac{1}{m}\cdot\sum_{i=1}^{t}c(\mathbf{v}_i)\cdot\frac{\tilde{p}_n(\mathbf{v}_i)}{\pi(\mathbf{v}_i)}=\sum_{i=1}^{t}\pi(\mathbf{v}_i)\cdot\frac{\tilde{p}_n(\mathbf{v}_i)}{\pi(\mathbf{v}_i)}=z_{1:t}. \tag{15}$$

Now that we have shown convergence of these estimators, equality follows directly

$$\lim_{m\to\infty}\mu_n(\mathbf{V};m)=\mathbb{E}_{p_n^{(1:t)}}\left[\mathbf{V}\right]\cdot\frac{z_{1:t}}{z}+\mathbb{E}_{p_n^{(t+1:n)}}\left[\mathbf{V}\right]\cdot\frac{z_{t+1:n}}{z}=\frac{\mathbb{E}_{\tilde{p}_n^{(1:t)}}\left[\mathbf{V}\right]+\mathbb{E}_{\tilde{p}_n^{(t+1:n)}}\left[\mathbf{V}\right]}{z}=\mathbb{E}_{p_n}\left[\mathbf{V}\right]. \tag{16}$$

□

D Score-based Eviction Methods as Probabilistic Eviction Policies

Any eviction method based on scores can easily be subsumed by probabilistic eviction. In fact, there exists an infinite number of proposal distributions that share the same top-k as the score strategy. In this section we show a few examples and how they can be turned into proposal distributions.

StreamingLLM. StreamingLLM selects the first four entries and a rolling window of the last k KV cache entries to keep, and evicts the rest. Any proposal distributions that have the first four and last k entries as modes are obviously equivalent in terms of their top-k modes; for example, a mixture of any monotonically increasing distribution with a distribution whose support consists of the first four entries.

H2O. H2O scores are based on the probabilities $p(\mathbf{V})$ and computed as

$$\text{score}_{\text{H2O}}(\mathbf{v}_i):=\sum_{j=1}^{n}p_j(\mathbf{v}_i). \tag{17}$$

Intuitively, each column in the softmax matrix is summed out. A proposal distribution can be achieved by simply normalizing these scores

$$\pi_{\text{H2O}}(\mathbf{v}_i)=\frac{\text{score}_{\text{H2O}}(\mathbf{v}_i)}{\sum_{j=1}^{n}\text{score}_{\text{H2O}}(\mathbf{v}_j)}. \tag{18}$$

K-norm. The K-norm scores are computed from the negative L^2 norm of the $\mathbf{V}$ matrix. This gives an n-dimensional vector, with a score for each entry

$$\text{score}_{\text{knorm}}(\mathbf{v}_i)=\left\|\mathbf{v}_i\right\|_2=\sqrt{\sum_{j=1}^{d}\mathbf{v}_{ij}^2}.$$

Because $\mathbf{V}$ may contain negative numbers, we apply a softmax instead of the usual normalization. Thus, the proposal distribution for K-norm is defined as

$$\pi_{\text{knorm}}(\mathbf{v}_i)=\frac{\exp\left(\text{score}_{\text{knorm}}(\mathbf{v}_i)\right)}{\sum_{j=1}^{t}\exp\left(\text{score}_{\text{knorm}}(\mathbf{v}_j)\right)}. \tag{19}$$

TOVA. Scores are given by the average $p_t(\mathbf{V})$ (i.e. the last query's distribution) across all heads in a layer. Let $\boldsymbol{h}_{\mathcal{L}} = \{h_1, h_2, \ldots, h_l\}$ be the set of all attention heads in a given layer $\mathcal{L}$. The score for each attention head $h_x \in \boldsymbol{h}_{\mathcal{L}}$ is given by

$$\text{score}_{\text{TOVA}}\left(\mathbf{v}_i^{(h_x)}\right) := \frac{1}{l} \cdot \sum_{i=1}^{l} p_t(\mathbf{v}_i). \tag{20}$$

The proposal distribution for TOVA is

$$\pi_{\text{TOVA}}(\mathbf{v}_i) = \frac{\text{score}_{\text{TOVA}}(\mathbf{v}_i)}{\sum_{j=1}^{n} \text{score}_{\text{TOVA}}(\mathbf{v}_j)}. \tag{21}$$

SnapKV. The process of obtaining the SnapKV scores is more involved and requires computing an average pooling of $p(\mathbf{V})$ for a given kernel and window size. In short, SnapKV computes a column average of $p(\mathbf{V})$ similar to H2O, applies a 1-dimensional pooling on this tensor, and then averages this again at the same dimension. We define the proposal distribution for SnapKV to be the normalized scores across entries, similar to the previous proposals.

D.1 Adjusting the Proposal Distribution

Any of these proposals can be equipped with other mechanisms for adjusting these probabilities. As an example, it is possible to apply a temperature scaling factor τ on these proposal distributions

$$\pi^{\tau}(\mathbf{v}_i) = \frac{\exp\left(\frac{1}{\tau} \cdot \text{score}(\mathbf{v}_i)\right)}{\sum_{j=1}^{t} \exp\left(\frac{1}{\tau} \cdot \text{score}(\mathbf{v}_j)\right)}$$

in order to decrease variance, or mix the proposal with a uniform in order to decrease bias

$$\pi'(\mathbf{v}_i) = \alpha \cdot \pi(\mathbf{v}_i) + (1 - \alpha) \cdot \frac{1}{t}, \text{ where } \alpha \in [0, 1].$$

E Prompt and Question for MAE Error

The prompt $\mathbf{x}$ consists of the tokenization of the following text:

> The Roman Empire was one of the largest and most enduring political entities in ancient history, reaching its greatest extent under Emperor Trajan in 117 CE when it spanned from Britain to Mesopotamia. Rome evolved from a monarchy to a republic before Julius Caesar's assassination in 44 BCE precipitated civil wars. Augustus emerged victorious and became the first emperor in 27 BCE, initiating a period of relative peace known as the Pax Romana. Roman law, architecture, and administrative practices shaped European civilization for centuries. Latin, the language of the Romans, evolved into the Romance languages including French, Spanish, Italian, Portuguese, and Romanian. The western empire collapsed in 476 CE, while the eastern half survived as the Byzantine Empire for nearly a thousand more years.

While the first token y is given by the first element in the tokenized sequence of the following string:

> Who was the first Roman emperor?

F On the Relationship Between Number of Samples and Compression Ratio

It is often useful, at a practical level, to control how much of the KV cache is evicted. For example, a practitioner might want to preserve as much generation quality given their GPU memory budget. This materializes as a so-called *compression ratio* r that describes the percentage of compression of a KV cache: when $r = 0$, no entries are evicted, $r = 0.5$ indicates that half of the KV cache is evicted, and

$r = 1$ means all of the entries are evicted. The probabilistic eviction procedure described in Section 4 makes no such distinction: given a number of samples m, it automatically sets a compression ratio based on the number of unique entries it has sampled. This problem is probabilistic in nature: in order to set an r, one would require finding an m s.t. a compression ratio of r is achieved on average.

Problem 3 (EXPECTEDSAMPLES)**.**
Input. Sequence length n, proposal π, target compression ratio $r^* \in [0, 1)$, error tolerance $\varepsilon \geq 0$.
Output. An $m \in \mathbb{N}$ such that

$$|\mathbb{E}_\pi[J(m)] - j^*| \leq \varepsilon, \tag{22}$$

where $j^* = (1 - r^*) \cdot n$, and $J(m)$ is a random variable denoting the number of unique values sampled after drawing m samples with replacement from π.

This can be reduced from the classical Erdős & Rényi coupon collector problem (Erdős & Rényi, 1961).

Problem 4 (Coupon Collector's Problem—CCP (Erdős & Rényi, 1961))**.**
Input. Set of coupons $\boldsymbol{a} = \{a_1, a_2, \ldots, a_n\}$, each with probability $p(A = a_i)$ of being issued, and a target number of unique coupons $j^* \in \mathbb{N}$.
Output. The expectation of the number of coupons that need to be drawn from $\boldsymbol{a}$ with replacement until at least j^* unique coupons are sampled.

In the KV eviction setting, the n sequence positions correspond to coupons, sampling from π corresponds to drawing coupons with probabilities $p(A = a_i)$, the number of samples m corresponds to the number of draws, and j^* is the number of distinct sampled positions/values. As these probabilities are discrete, the solution (Flajolet et al., 1992) is

$$\mathbb{E}[m] = \sum_{q=0}^{j^*-1} \left((-1)^{j^*-1-q} \binom{n-q-1}{n-j^*} \sum_{|\boldsymbol{s}|=q} \frac{1}{1-\rho_{\boldsymbol{s}}} \right), \tag{23}$$

where the inner sum is over all subsets $\boldsymbol{s} \subseteq \{1, \ldots, n\}$ with $|\boldsymbol{s}| = q$, and $\rho_{\boldsymbol{s}} = \sum_{i \in \boldsymbol{s}} p(A = a_i)$.

Complexity. Evaluating Equation (23) naively requires exponential time.

Let $\alpha = (-1)^{j^*-1-q}\binom{n-q-1}{n-j^*}$ and $\beta = \sum_{|\boldsymbol{s}|=q} \frac{1}{1-\rho_{\boldsymbol{s}}}$. Equation (23) can now be written as

$$\mathbb{E}[m] = \sum_{q=0}^{j^*-1} \alpha\beta. \tag{24}$$

The amortized complexity of evaluating α is $O(1)$ if we use a lookup table. For β, there are $\binom{n}{q}$ subsets $\boldsymbol{s}$ such that $|\boldsymbol{s}| = q$, and each subset sums over q probabilities. As n increases and q varies, evaluating β scales exponentially. Therefore, evaluating Equation (23) naïvely requires *at least* exponential time.

Although Equation (23) gives us the exact solution we need, it is computationally infeasible to compute it in the naïve way. In fact, as far as we know, this problem has not been shown to either be in P *or* NP-hard. Our conjecture is that Problem 4 is in fact intractable, and thus cannot be computed exactly in polynomial time.

To solve Problem 3 efficiently, we consider a different approach. Intuitively, we do not need to know the **exact** number of samples $m \in \mathbb{R}_{\geq 0}$ to draw j^* unique values. For the purposes of KVEVICT, m is a non-negative integer, which reduces the search space massively, especially given a suitable upper bound. Therefore, we can utilize a trial and error approach optimized by binary search. Instead of obtaining m **directly** given n, π, and r^*, we repeatedly **guess** what m should be and (quickly) verify if/how our guess $\hat{m}$ needs to be updated for the next guess by comparing the expected number of unique values sampled $\mathbb{E}_\pi[J(\hat{m})]$ to the target j^*.

Theorem 2. Algorithm 2 solves Problem 3 (EXPECTED SAMPLES) with error $|\mathbb{E}_\pi[J(m)] - j^*| \leq \varepsilon$, where $\varepsilon = 0.5$; assuming full support on π.

Algorithm 2 EXPECTED SAMPLES

Input Sequence length n, proposal distribution $\pi(\mathbf{v}) = [\pi(\mathbf{v}_1), \pi(\mathbf{v}_2), ..., \pi(\mathbf{v}_n)]$, target compression ratio $r^* \in [0, 1]$, and error tolerance $\varepsilon = 0.5$.

Output A non-negative integer m such that $|\mathbb{E}_\pi[J(m)] - j^*| \leq \varepsilon$, where $j^* = (1 - r^*) \cdot n$, and $J(m)$ is the random variable denoting the number of unique values sampled after drawing m samples with replacement from π.

1: $j^* \leftarrow (1 - r^*) \cdot n$ ▷ Target number of unique values sampled
2: $\alpha \leftarrow 10^{12}$ ▷ Large constant; existence of a valid α is guaranteed
3: $\texttt{low} \leftarrow 0, \texttt{high} \leftarrow \alpha$ ▷ by Part 2 of Theorem 2
4: **while** $\texttt{low} \leq \texttt{high}$ **do** ▷ Binary search
5: $\hat{m} \leftarrow \lfloor (\texttt{low} + \texttt{high}) \div 2 \rfloor$
6: $\mathbb{E}_\pi[J(\hat{m})] \leftarrow \sum_{i=1}^{n} \left(1 - (1 - \pi(v_i))^{\hat{m}}\right)$ ▷ Expected number of unique values sampled
7: **if** $|\mathbb{E}_\pi[J(\hat{m})] - j^*| \leq \varepsilon$ **then** ▷ Target j^* reached
8: $m \leftarrow \hat{m}$
9: **break**
10: **else if** $\mathbb{E}_\pi[J(\hat{m})] < j^*$ **then**
11: $\texttt{low} \leftarrow \hat{m} + 1$
12: **else**
13: $\texttt{high} \leftarrow \hat{m} - 1$
14: **return** m

Proof.

1. We first show that line 6 in Algorithm 2 gives us the expected number of unique values sampled (Tauraso, 2016)

$$\mathbb{E}_\pi[J(m)] = \sum_{i=1}^{n} \left(1 - (1 - \pi(v_i))^m\right). \tag{25}$$

Let J_i be the indicator random variable for position i: $J_i = 1$ if v_i is sampled, 0 otherwise. For the i-th position,

$$\mathbb{E}_\pi[J_i(m)] = p(J_i = 1) = 1 - (1 - \pi(v_i))^m. \tag{26}$$

We want the total expected number of unique values sampled $J(m) = \sum_{i=1}^{n} J_i$. By linearity of expectation,

$$\mathbb{E}_\pi[J(m)] = \mathbb{E}_\pi[J_1(m) + J_2(m) + \cdots + J_n(m)] = \sum_{i=1}^{n} \mathbb{E}_\pi[J_i(m)] = \sum_{i=1}^{n} \left(1 - (1 - \pi(v_i))^m\right). \tag{27}$$

2. We show that i) $\mathbb{E}_\pi[J(m)]$ is non-decreasing in m and ii) that there exists a finite upper bound α such that $\mathbb{E}_\pi[J(\alpha)] \geq j^*$.

   i) Each term $(1-(1-\pi(v_i))^m)$ is non-decreasing in m as $\pi(v_i) \in (0, 1]$ implies $(1-\pi(v_i))^m \in [0, 1)$, making $(1 - \pi(v_i))^m$ non-increasing. Hence, $\mathbb{E}_\pi[J(m)]$ is non-decreasing in m.

   ii) Assuming π has full support ($\pi(v_i) > 0$ for all i), since each $(1 - \pi(v_i)) < 1$, we have $(1 - \pi(v_i))^m \to 0$ as $m \to \infty$, so $\mathbb{E}_\pi[J(m)] \to n \geq j^*$. Hence, a finite α with $\mathbb{E}_\pi[J(\alpha)] \geq j^*$ always exists. In practice, we simply set $\alpha = 10^{12}$ rather than computing α from π, which suffices in all realistic settings.

3. Finally, we show $|\mathbb{E}_\pi[J(m)] - j^*| \leq \varepsilon$, where $\varepsilon = 0.5$. The difference between consecutive number of samples $m \in \mathbb{Z}$ is given by

$$\mathbb{E}_\pi[J(m+1)] - \mathbb{E}_\pi[J(m)] = \sum_{i=1}^{n} \left(1 - (1 - \pi(v_i))^{m+1}\right) - \sum_{i=1}^{n} \left(1 - (1 - \pi(v_i))^m\right) \tag{28}$$

$$= \sum_{i=1}^{n} \left((1 - (1 - \pi(v_i))^{m+1}) - (1 - (1 - \pi(v_i))^{m}) \right) \tag{29}$$

$$= \sum_{i=1}^{n} \left((1 - \pi(v_i))^{m} - (1 - \pi(v_i))^{m+1} \right) \tag{30}$$

$$= \sum_{i=1}^{n} (1 - \pi(v_i))^{m} (1 - (1 - \pi(v_i))) = \sum_{i=1}^{n} (1 - \pi(v_i))^{m} \pi(v_i). \tag{31}$$

Because $0 < \pi(v_i) \leq 1 \; \forall \; i = 1...n$, $(1 - \pi(v_i))^m \leq 1$. Therefore,

$$\mathbb{E}_\pi[J(m+1)] - \mathbb{E}_\pi[J(m)] = \sum_{i=1}^{n} (1 - \pi(v_i))^{m} \pi(v_i) \leq \sum_{i=1}^{n} \pi(v_i) = 1. \tag{32}$$

We now prove by strong induction that for every $j^* \in [0, \, \mathbb{E}_\pi[J(m)]]$, there exists $m' \leq m$ with $|\mathbb{E}_\pi[J(m')] - j^*| \leq 0.5$.

Base case ($m = 0$). $\mathbb{E}_\pi[J(0)] = 0$, so the only j^* in $[0, 0]$ is $j^* = 0$, and $|\mathbb{E}_\pi[J(0)] - 0| = 0 \leq 0.5$.

Inductive step. Assume that the claim holds for all $m' < m$. Let $j^* \in [0, \, \mathbb{E}_\pi[J(m)]]$.

- If $j^* \leq \mathbb{E}_\pi[J(m-1)]$, there must be some $m' \leq m - 1$ with $|\mathbb{E}_\pi[J(m')] - j^*| \leq 0.5$ following the assumption.
- If $j^* \in \big(\mathbb{E}_\pi[J(m-1)], \, \mathbb{E}_\pi[J(m)]\big]$, then

$$\underbrace{(\mathbb{E}_\pi[J(m)] - j^*)}_{\geq 0} + \underbrace{(j^* - \mathbb{E}_\pi[J(m-1)])}_{\geq 0} = \mathbb{E}_\pi[J(m)] - \mathbb{E}_\pi[J(m-1)] \leq 1, \tag{33}$$

  so at least one term is ≤ 0.5, meaning either m or $m - 1$ satisfies the bound.

Conclusion. By strong induction, for every $j^* \in [0, \, \mathbb{E}_\pi[J(m)]]$, there exists $m' \leq m$ with $|\mathbb{E}_\pi[J(m')] - j^*| \leq 0.5$.

□

Remark 1. Theorem 2 assumes π has full support, which holds in practice as softmax assigns some probability to every token. However, if attention is highly peaked, there can be many $\mathbf{v}_i$ where $\pi(\mathbf{v}_i) \approx 0$, causing these tokens to be rarely sampled. As such, achieving low compression ratios (large j^*) may require an impractically large m. While this is theoretically an issue, empirically it does not arise at reasonable compression ratios ($r^* \geq 0.1$), where tokens with near-zero attention weight are inconsequential anyway.

Proposition 2. Algorithm 2 is $O(n \log(\alpha))$.

Proof. Evaluating $\mathbb{E}_\pi[J(\hat{m})]$ in Equation (25) requires $O(n)$ time as we sum over $i = 1...n$. The binary search is bounded by α, resulting in $O(\log(\alpha))$ complexity. For each trial in binary search, we evaluate $\mathbb{E}_\pi[J(\hat{m})]$, so the total complexity is $O(n \log(\alpha))$. □

F.1 Globally soft constraining the compression ratio

We now consider the case of constraining the compression ratio at a global level. Before, we set the number of samples m to be such that the compression ratio of an attention head would be on average a target compression ratio r^*. To do this, we needed to be able to compute the expectation $\mathbb{E}_\pi[J(m)]$. This can be easily done as we have access to $\pi(v_i)$, as we saw in the previous subsection. However, this is not optimal. By constraining each attention head, we are effectively forcing each attention head to select a possibly unnatural choice of compression ratio; for example, consider the case where the

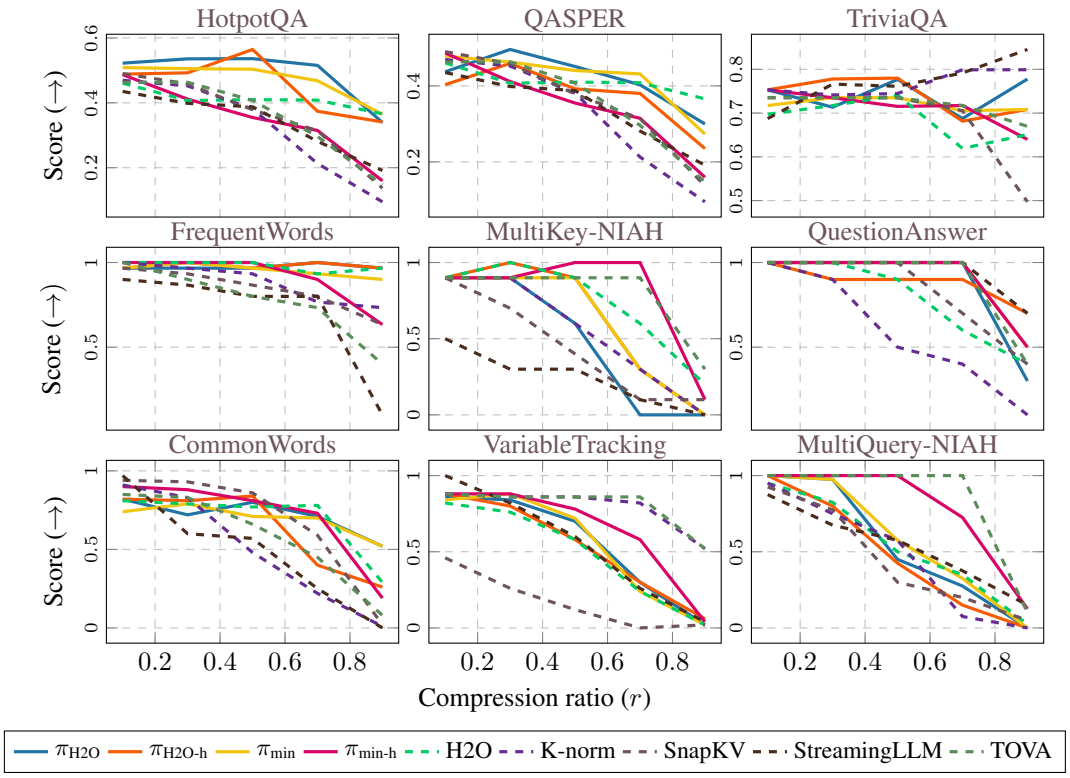


Figure 4: **Scores for each individual dataset split on Llama3.** Solid lines show the score for the four proposals defined in Section 7. Dashed lines show the five top-k eviction methods.

sequence length $n = 4$, $\pi = (0.8, 0.0998, 0.0001, 0.0001)$ and $m = 100$: the expected compression ratio for this m is

$$1 - \frac{\mathbb{E}_\pi[J(m)]}{n} \approx 1 - \frac{2}{4} = 0.5.$$

But if we set a target compression ratio $r^* = 0.75$, then we require $m \approx 7000$ samples in order to achieve this without unbiasing the attention head output (on average). A similar argument can be made to lowering the compression ratio: setting $r^* = 0.25$ requires us to go as low as $m = 2$ samples, which can introduce a lot of variance.

Thus, an attractive alternative is to (soft) constraint the compression ratio at a global level: instead of per-head, we set a fixed number of samples for all attention heads such that the average compression ratio across all heads[1] is (on average) a target compression ratio r^*. Doing so is easy; we are interested in the expression $\mathbb{E}\left[\frac{1}{h} \cdot \sum_{i=1}^{h} J^{(i)}(m) \,\middle|\, \boldsymbol{x}\right]$, where $J^{(i)}(m)$ is the number of unique sampled indices for attention head i after m draws and $\boldsymbol{x}$ is the prompt. By linearity of expectation,

$$\mathbb{E}\left[\frac{1}{h} \cdot \sum_{i=1}^{h} J^{(i)}(m) \,\middle|\, \boldsymbol{x}\right] = \frac{1}{h} \cdot \sum_{i=1}^{h} \mathbb{E}\left[J^{(i)}(m) \,\middle|\, \boldsymbol{x}\right] = \frac{1}{h} \cdot \sum_{i=1}^{h} \mathbb{E}_{\pi^{(i)}}\left[J^{(i)}(m) \,\middle|\, \boldsymbol{x}\right]. \tag{34}$$

Note that the last equality is exactly the same expression we previously computed (we omitted the conditioning on the prompt earlier). Thus, Algorithm 2 can be run at the global level in order to select an adequate number of samples for a target compression ratio by only replacing $\mathbb{E}_\pi[J(m) \mid \boldsymbol{x}]$ with Equation (34).

[1]This is reasonable since all attention heads have the same size.

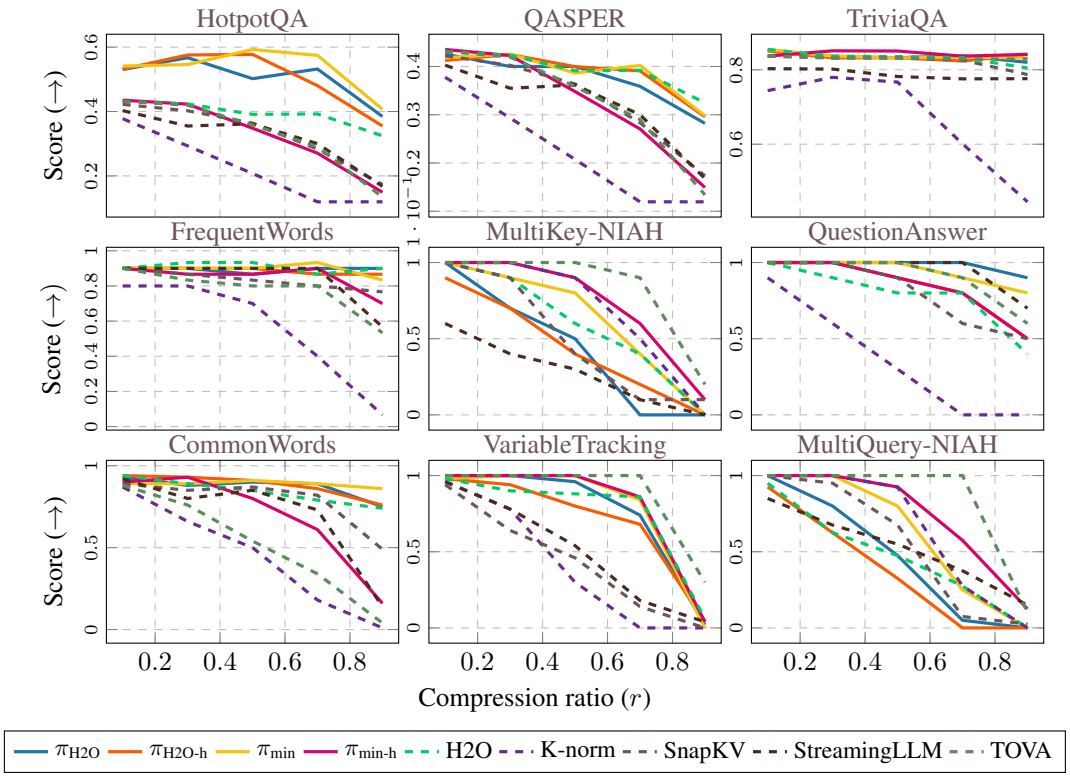


Figure 5: **Scores for each individual dataset split on `Qwen3`.** Solid lines show the score for the four proposals defined in Section 7. Dashed lines show the five top-k eviction methods.

F.2 Target vs Effective Compression Ratio

Table 2 empirically shows that when we set the target compression ratio in Algorithm 2, we obtain an effective compression ratio with minimal error. In fact, most if not all of the error can be explained by the fact that the original number of entries t might not be divisible by the number of kept entries k.

G Experimental Details

We use the evaluation pipeline and implementations of SnapKV, TOVA, H2O, StreamingLLM, and K-norm available in KVPress v0.5.3 (Devoto et al., 2025). All experiments are run on two NVIDIA RTX A6000 (48GB) and eight NVIDIA RTX A5000 (24GB). All experiments can be run on a 24GB GPU, taking approximately 3 to 5 seconds to completely run each example in either LongBench or RULER. The full research project required more compute than what is reported in the paper

Table 2: **Difference between target and effective compression ratio is insignificant** for `Llama3` on the RULER dataset. Results are similar for `Qwen3` and LongBench.

Target r	Mean effective r ($\pm$ std)			
	π_{H2O}	$\pi_{H2O\text{-}h}$	π_{min}	$\pi_{min\text{-}h}$
0.1	0.0999 ± 0.0003	0.1000 ± 0.0003	0.1000 ± 0.0002	0.0999 ± 0.0003
0.3	0.2999 ± 0.0004	0.3001 ± 0.0004	0.3000 ± 0.0004	0.2999 ± 0.0005
0.5	0.4999 ± 0.0004	0.5000 ± 0.0004	0.5000 ± 0.0004	0.5001 ± 0.0004
0.7	0.6999 ± 0.0004	0.7000 ± 0.0004	0.7000 ± 0.0004	0.7001 ± 0.0004
0.9	0.9000 ± 0.0003	0.9000 ± 0.0002	0.8999 ± 0.0002	0.9000 ± 0.0002

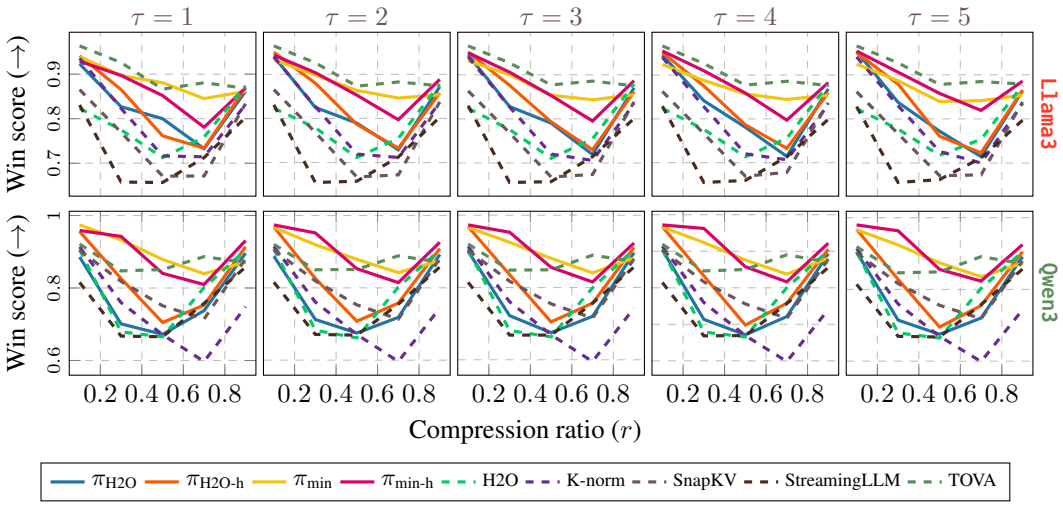


Figure 6: **Importance sampling temperature scaling pairwise win scores.** `Llama3` (top) and `Qwen3` (bottom) pairwise win scores at different importance sampling temperature scale values τ.

due to preliminary experiments as well as initial experimentation that either contained errors in implementation or were not as competitive.

H Minimum Variance Proposals and Priors

We define the following four proposal distributions

$$\pi_{\text{min}}(\mathbf{v}_i) := \frac{p_t(\mathbf{v}_i) \cdot \|\mathbf{v}_i\|}{\sum_{j=1}^{t} p_t(\mathbf{v}_j) \cdot \|\mathbf{v}_j\|}, \qquad \pi_{\text{min-h}}(\mathbf{v}_i) := \frac{\sum_{j=1}^{t} p_j(\mathbf{v}_i) \cdot h_j \cdot \|\mathbf{v}_i\|}{\sum_{l=1}^{t}\sum_{j=1}^{t} p_j(\mathbf{v}_l) \cdot h_j \cdot \|\mathbf{v}_l\|}, \tag{35}$$

$$\pi_{\text{H2O}}(\mathbf{v}_i) := \frac{\sum_{j=1}^{t} p_j(\mathbf{v}_i)}{\sum_{l=1}^{t}\sum_{j=1}^{t} p_j(\mathbf{v}_l)}, \qquad \pi_{\text{H2O-h}}(\mathbf{v}_i) := \frac{\sum_{j=1}^{t} p_j(\mathbf{v}_i) \cdot h_j}{\sum_{l=1}^{t}\sum_{j=1}^{t} p_j(\mathbf{v}_l) \cdot h_j}; \tag{36}$$

where $h_j := [(t-j+1) \cdot \mathcal{H}_t]^{-1}$ and $\mathcal{H}_t$ is the t-th harmonic number.

In this section, we discuss what exactly are the semantics of these distributions. Proposal π_{min} is well known folklore result in the statistical methods literature (Owen, 2013). It is simply the proposal distribution that minimizes the variance for the expectation estimator for the last query attention value, i.e. $\mathbb{E}_{p_t}[\mathbf{V}]$: the expectation at eviction time. Notably, it is a proposal that uses the values of $\mathbf{V}$ in order to choose which entries to evict. None of the top-k eviction methods we compare and study make use of both $p(\mathbf{V})$ *and* $\mathbf{V}$; see Appendix D for a description on how these eviction strategies compute their score.

Proposal $\pi_{\text{min-h}}$ extends this to all distributions $\{p_i(\mathbf{V})\}_{i=1}^{t}$. To be precise, recall that the softmax matrix in Equation (5) $p(\mathbf{V})$ at eviction time t (i.e. during the prefilling step) is $t \times t$. The first proposal π_{min} ignores all of the first $[1..t-1]$ rows, even though they may contain useful information to decide on which entries to evict. We interpret these rows as a random variable T that, when used to condition the probability of eviction, tells you which entries are best to evict. That is,

$$\text{softmax}\left(\mathbf{Q} \cdot \mathbf{K}^{\intercal}\right) = [p(\mathbf{V}|T=i)]_{i=1}^{t}. \tag{37}$$

This assumes that there exists a distribution on T. We interpret this distribution $p(T)$ as some prior knowledge on how meaningful are each $p(\mathbf{V}|T=i)$ as i distances from t. In both $\pi_{\text{min-h}}$ and π, we set this to

$$h_j := p(T=j) = \frac{1}{(t-j+1) \cdot \mathcal{H}_t}. \tag{38}$$

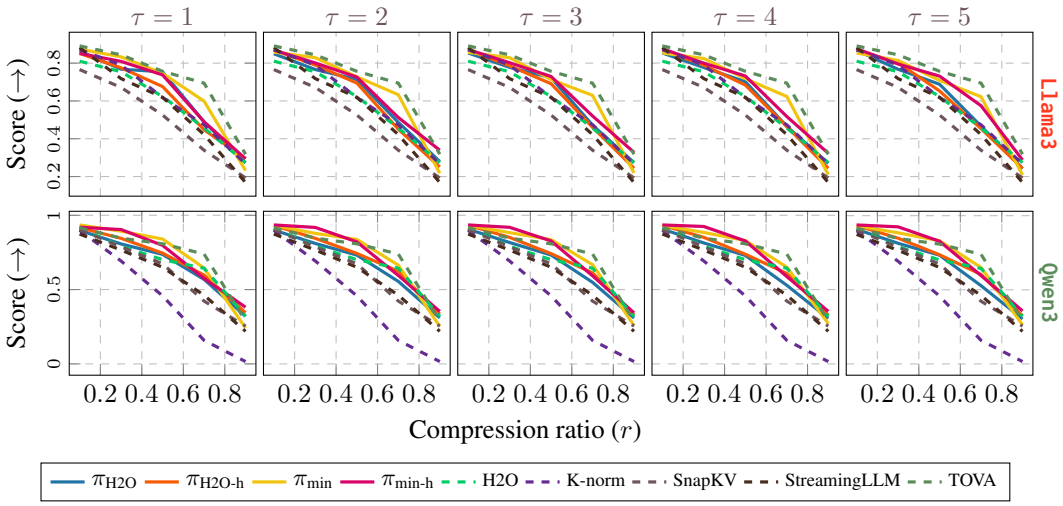


Figure 7: **Importance sampling temperature scaling average total scores.** Llama3 (top) and Qwen3 (bottom) average total scores at different importance sampling temperature scale values τ.

The semantics of this prior is simple: we assume that each row has exponentially diminishing importance: we weigh each row according to the following sequence $\mathbf{h} = \left[\frac{1}{t}, \frac{1}{t-1}, \ldots, \frac{1}{3}, \frac{1}{2}, 1\right]$ and then normalize to get a probability distribution. The normalizing constant of sequence $\mathbf{h}$ is the harmonic number $\mathcal{H}_t = \sum_{k=1}^{t} \frac{1}{k}$, and the probability of each row is $p(T = i) = [(t - j + 1) \cdot \mathcal{H}_h]^{-1}$

To then retrieve the probability of each entry $p(\mathbf{v}_i)$, we marginalize over all possible time steps

$$p(\mathbf{v}_i) = \sum_{j=1}^{t} p(\mathbf{v}_i, T = j) = \sum_{j=1}^{t} p(\mathbf{v}_i | T = j) \cdot p(T = j), \tag{39}$$

which when using $p(T = j) = h_j$ for either the minimum variance proposal or the H2O proposal, yields $\pi_{\text{min-h}}$ and $\pi_{\text{H2O-h}}$ respectively. Note that any distribution for $p(T)$ is valid and results in a different proposal. In fact, any proposal that ignores the first $[1..t-1]$ rows contains a prior that is degenerated at $T = t$, with zero probability everywhere else.

I Additional Experiments

In this section, we show additional experiments that supplement the claims in Section 7.

I.1 Per Dataset Split Scores

We show the eviction scores for each of the four proposals defined in Section 7 and the five top-k eviction methods—namely, Streaming LLM, SnapKV, TOVA, H2O and K-norm—across all of the splits of LongBench and some of RULER's. Due to space and time constraints, we only show six out of the 13 different splits in RULER.

Figure 4 shows the scores for Llama3 and Figure 5 shows the scores for Qwen3. In both cases, probabilistic eviction with correction tends to achieve better score performance. Notably, some eviction methods that do well in certain tasks. For example, on Llama3 StreamingLLM ranks first on the QuestionAnswer split, yet dead last in MultiKey-NIAH. Similarly, on Qwen3 K-norm is consistently in the top-3 on the MultiKey-NIAH split, yet last in CommonWords.

I.2 Importance Sampling Temperature

To further investigate the trade-off between bias and variance, we apply importance sampling temperature scaling to the same experimental setting as described in Section 7. Figures 6 and 7 show

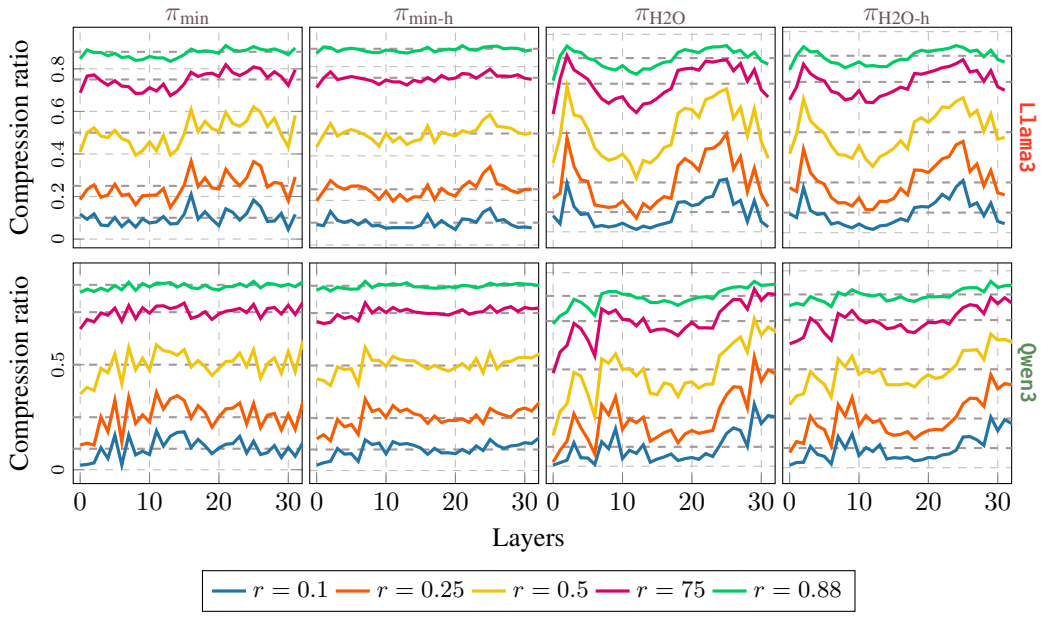


Figure 8: **Pattern of average compression ratio per layer when probabilistically evicting.** Each solid curve shows the average effective compression ratio when given a target compression ratio r. The target compression ratio is shown as a dashed dark gray line. The average of all points in a solid curve equals to the target r. This pattern shows that lower layers (closer to input) tend to evict less, while higher layers (closer to output) evict more.

the scores and pairwise win scores of RULER on `Llama3` and `Qwen3` across temperature values of $\tau \in \{1, 2, 3, 4, 5\}$. The plots show that, in most cases, the decrease in variance was not worth the increase in bias. This suggests that bias is the more important issue to resolve in KV eviction.

I.3 On the Behavior of Attention Head Compression Ratio

Given a fixed number of samples m, probabilistic eviction will automatically set the appropriate compression ratio of each head in accordance with that head's proposal distribution it samples from. This means that different attention heads might have very distinct compression ratios then others. Our findings show that this causes a distinct pattern in the distribution of attention head compression ratios. We find that lower (closer to the input) layers tend to be allocated more of the cache budget, while at higher (closer to the output) layers, less budget. This is a behavior that is reminiscent of Cai et al. (2025)'s PyramidKV, where lower layers are given more of the KV cache budget.

Figure 8 shows this pattern for different proposal distributions. Notably, $\pi_{\text{min-h}}$ tends to act more conservatively, while π_{H2O} spikes rapidly at certain layers.

J On Rewarding Ties

In Section 7, we argue for a rewarding scheme where a ranking is constructed from pairwise comparisons of methods across different tasks where both wins and ties are rewarded. This is to punish catastrophic failures: if one were not to reward ties, any method that achieves a high ranking has some leeway to fail horrendously. In other words, by not rewarding ties methods that achieve a high rank in one task can leverage this fact to do poorly in some other task. If one reward ties, then any method that performs robustly (but is not necessarily the best in all tasks) is given points for not only winning against worse ones, but also doing as well as just as good methods. This ultimately means that the best model so far has to continue to do well (as good as the lower ranked tying models) in order to maintain its top position. For example, given methods $\boldsymbol{M} = \{A, B, C, D\}$, consider the

Table 3: **Ranking scores for Llama3 on LongBench when not rewarding ties (a) and when rewarding ties (b).** When not rewarding ties, $\pi_{\text{H2O-h}}$ dominates the benchmark; for each proposal $\pi \in \{\pi_{\text{min-h}}, \pi_{\text{H2O}}, \pi_{\text{H2O-h}}\}$, π tends to outperform the other non-probabilistic methods. The same can be said when rewarding ties: $\{\pi_{\text{min-h}}, \pi_{\text{H2O}}, \pi_{\text{H2O-h}}\}$ tends to dominate non-probabilistic eviction strategies.

(a) No reward for tying

	Compression ratio (r)				
Method	0.1	0.3	0.5	0.7	0.9
$\pi_{\text{min-h}}$	0.1039	0.1385	0.1931	0.2361	0.2670
π_{H2O}	0.0847	0.1729	0.2089	0.2304	0.2779
$\pi_{\text{H2O-h}}$	0.0950	**0.1790**	**0.2189**	**0.2711**	**0.2940**
π_{min}	0.1022	0.1322	0.1290	0.1563	0.1351
H2O	0.1030	0.1465	0.2087	0.2441	0.2871
K-norm	0.0920	0.1202	0.1447	0.1659	0.1889
SnapKV	0.1005	0.1319	0.1533	0.1575	0.1271
StreamingLLM	**0.1390**	0.1756	0.2015	0.1972	0.2431
TOVA	0.0828	0.1143	0.1633	0.1812	0.1252

(b) Reward for tying

	Compression ratio (r)				
Method	0.1	0.3	0.5	0.7	0.9
$\pi_{\text{min-h}}$	**0.9246**	0.8877	0.8546	**0.8450**	0.8250
π_{H2O}	0.8889	**0.8935**	0.8611	0.8145	0.8100
$\pi_{\text{H2O-h}}$	0.9147	0.8860	**0.8803**	0.8322	0.8607
π_{min}	0.9065	0.8250	0.7609	0.7563	0.7286
H2O	0.8777	0.8451	0.8388	0.8396	**0.8650**
K-norm	0.9025	0.8368	0.7787	0.7509	0.7169
SnapKV	0.9225	0.8625	0.8097	0.7688	0.6958
StreamingLLM	0.8445	0.7901	0.7592	0.7500	0.8140
TOVA	0.9152	0.8623	0.8352	0.8030	0.7386

Table 4: **ELO rankings and scores for Llama3 on LongBench.** Not only do most of our probabilistic proposals dominate the non-probabilistic methods, but their lower confidence bound tend to be higher than other method's lower confidence bounds, showing the robustness of our methods.

	Compression ratio									
Method	$r = 0.1$		$r = 0.3$		$r = 0.5$		$r = 0.7$		$r = 0.9$	
$\pi_{\text{min-h}}$	**1008.6**	[997.0, 1020.0]	1007.9	[994.2, 1023.3]	1014.4	[997.7, 1031.8]	1024.5	[1008.2, 1043.3]	1028.1	[1006.9, 1046.9]
π_{H2O}	992.1	[976.8, 1006.5]	**1020.0**	**[1003.3, 1036.6]**	1021.1	[1002.5, 1040.7]	1013.5	[992.4, 1037.3]	1026.8	[1004.5, 1052.0]
$\pi_{\text{H2O-h}}$	1002.9	[988.4, 1016.4]	1019.6	[1000.3, 1037.5]	**1030.0**	**[1012.0, 1045.1]**	**1031.2**	**[1009.5, 1052.2]**	**1047.3**	**[1026.1, 1066.3]**
π_{min}	1002.6	[988.7, 1017.8]	987.1	[968.9, 1005.4]	966.7	[948.4, 988.6]	973.6	[954.8, 994.2]	958.4	[938.9, 977.2]
H2O	994.2	[976.6, 1013.2]	997.5	[978.4, 1014.3]	1014.3	[993.4, 1036.1]	1025.3	[1007.6, 1042.9]	1046.5	[1023.6, 1068.8]
K-norm	998.3	[984.3, 1012.1]	987.1	[971.7, 1005.4]	976.9	[956.9, 996.2]	974.9	[955.5, 995.8]	971.3	[950.0, 992.8]
SnapKV	1006.9	[993.4, 1019.8]	998.3	[983.6, 1013.9]	988.9	[970.0, 1008.8]	977.7	[959.3, 994.8]	945.8	[925.7, 966.0]
StreamingLLM	995.0	[974.2, 1015.1]	989.7	[966.4, 1013.7]	988.2	[962.4, 1012.1]	984.1	[959.3, 1007.1]	1017.4	[1000.9, 1039.2]
TOVA	999.4	[985.1, 1011.7]	992.9	[978.5, 1010.1]	999.6	[982.0, 1015.0]	995.2	[976.9, 1013.8]	958.4	[940.3, 976.7]

two following scenarios:

Rank	**First task**	**Second task**
1st:	$\{A\}$	$\{D\}$
2nd:	$\{B, C\}$	$\{B, C\}$
3rd:	$\{D\}$	$\{A\}$

When ties are *not* rewarded, the resulting scores are $\text{scores}_0(\boldsymbol{M}) = \{A : 3, B : 2, C : 2, D : 3\}$; a method gets a point if it beats an opponents, no points against tied opponents. A and D are ranked first *even though they failed catastrophically* in the second and first tasks respectively. However, when ties *are* rewarded, the scores then become $\text{scores}_1(\boldsymbol{M}) = \{A : 3, B : 4, C : 4, D : 3\}$, as now B and C get an additional point in both tasks due to ties. This bumps B and C to first place precisely because they are stable and have come out as robust performers.

K Quantitative Robustness Evaluation

In this section, we provide quantitative values on robustness evaluation for Llama3 on the LongBench dataset. As mentioned in Appendix J, rewarding ties should provide a clearer picture of robustness, however to further complement evaluation, we also evaluate on the same metric but without tie rewarding, and provide ELO ratings together with a 95% confidence bound.

Table 3 shows ranking scores without rewarding ties (Table 3a) and with tie rewarding (Table 3b). The values show that our probabilistic method (across proposals $\{\pi_{\text{min-h}}, \pi_{\text{H2O}}, \pi_{\text{H2O-h}}\}$) tends to outperform non-probabilistic ones (namely H2O, K-norm, SnapKV, StreamingLLM and TOVA). Table 4 shows the same experiment but with an ELO ranking (with 95% confidence bounds) instead of pairwise win-tie-lose comparisons, with a similar conclusion. Crucially, the lower confidence bound in the probabilistic methods tend to be higher than non-probabilistic ones, demonstrating the robustness of our method.

Table 5: **Eviction runtime (in seconds) for Llama3 on LongBench together with absolute and relative differences.** The overhead for probabilistic eviction is negligible, at most 8.3% slower compared to the existing deterministic heuristics.

(a) Raw eviction runtime

Compression	$\pi_{\text{min-h}}$ (ours)	SnapKV	H2O	TOVA
0.1	1.939 ± 0.295	1.805 ± 0.269	1.851 ± 0.280	1.807 ± 0.271
0.3	1.945 ± 0.297	1.807 ± 0.269	1.852 ± 0.280	1.805 ± 0.270
0.5	1.943 ± 0.300	1.806 ± 0.267	1.850 ± 0.278	1.805 ± 0.268
0.7	1.948 ± 0.298	1.805 ± 0.270	1.848 ± 0.279	1.804 ± 0.269
0.9	1.944 ± 0.298	1.805 ± 0.266	1.849 ± 0.275	1.795 ± 0.266

(b) Absolute eviction difference

Compression	SnapKV	H2O	TOVA
0.1	0.133	0.088	0.132
0.3	0.138	0.093	0.140
0.5	0.137	0.093	0.138
0.7	0.143	0.100	0.144
0.9	0.139	0.095	0.148

(c) Relative eviction difference

Compression	SnapKV	H2O	TOVA
0.1	1.074	1.048	1.073
0.3	1.076	1.050	1.077
0.5	1.076	1.050	1.077
0.7	1.079	1.054	1.080
0.9	1.077	1.051	1.083

L The Overhead of Probabilistic Eviction and Correction

In terms of memory overhead, ImportanceAttention requires storing a single k-sized vector containing the importance weights $\mathbf{w} = \left[\frac{c(\mathbf{v}_i)}{m \cdot \pi(\mathbf{v}_i)}\right]_{i=1}^{k}$ for each attention head, where k is the size of non-evicted entries at compression time. Note that this vector does not grow, and so the overhead is constant with respect to the number of entries generated after eviction. In fact, k is dominated by the number of generated tokens *after* eviction, and thus the overhead size of $\mathbf{w}$ is negligible compared to the memory saved in the embedding size for the generated tokens.

Runtime-wise, at eviction time one requires only m sampling runs (where m can be determined from Algorithm 2 as a function of the chosen compression ratio or set as a constant) and computing $\mathbf{w}$, meaning the overall overhead for eviction is $\mathcal{O}(h \cdot (m+k))$, where h is the number of atttention heads. Table 5 shows the eviction runtime, as well as absolute and relative difference between probabilistic eviction and the three most competitive heuristics: SnapKV, H2O and TOVA.

Correction requires a single Hadamard product operation between the correction factor (materialized as the weights $\mathbf{w}$) and the current time step distribution. Time differences are shown in Table 6, with overhead below a millisecond and a relative overhead difference of $3.749\times$.

It is important to note that our runtime experiments show an upper bound on the runtime. Our implementation uses a naïve implementation where we recompute $\mathbf{w}$ every decoding step for each to-be-generated token. Due to restrictions on KVPress (the KV eviction framework we implement our method on), we also recompute many entries that would have been evicted in a sparse tensor implementation. One could improve upon this through a bespoke implementation that uses a similar mechanism as Feng et al. (2026). The focus of the paper is to introduce a new class of KV cache eviction estimators, make the connection with the statistical methods literature, and validate the improvement in performance and robustness compared to existing heuristic methods. Our implementation is mostly to provide a simple and correct example of how one would use probabilistic eviction and correction in practice. Even then, we show that overhead is very small to negligible.

M Limitations

The ideal KV eviction method is not only zero-variance but also zero-bias. However, it is unrealistic to achieve this, as to do so, correction would have to be perfect, meaning that it can perfectly recreate the same KV cache from less memory than what was evicted. Our approach is also not perfect: self-normalized importance sampling is biased and has variance greater than zero. However, this

Table 6: **Correction runtime (in seconds) for Llama3 on LongBench together with absolute and relative differences.** The relative difference between correction and no correction is at most 3.749×; a higher relative difference is expected since the absolute difference between correction and no correction is less than a millisecond.

Compression	Correction	No correction	Absolute diff	Relative diff
0.1	1.899e-04 ± 2.204e-05	5.072e-05 ± 2.984e-06	1.391e-04	3.473
0.3	1.863e-04 ± 1.798e-05	5.083e-05 ± 3.141e-06	1.354e-04	3.664
0.5	1.868e-04 ± 1.949e-05	5.082e-05 ± 3.021e-06	1.360e-04	3.677
0.7	1.917e-04 ± 2.186e-05	5.145e-05 ± 2.927e-06	1.403e-04	3.726
0.9	1.920e-04 ± 2.273e-05	5.123e-05 ± 3.523e-06	1.408e-04	3.749

estimator has reasonable upper bounds on bias and variance. This is in contrast to deterministic KV eviction methods that do not apply correction: these are zero-variance estimators but their bias is unbounded.

The bias of self-normalized importance sampling is upper bounded by $\mathcal{O}(m^{-1})$, where m is the number of samples. This upper bound indicates that if there is less compression, then it will be more accurate—which is usually true for other eviction methods. Note that the inverse is not true: at a reasonable compression ratio, if the distributions at the attention heads are sparse, then the effective number of samples required to achieve a certain precision will be lower, meaning that in those cases even a higher compression ratio can achieve lower error.

In terms of implementation, we show that we are able to achieve distinct compression ratios at different heads. This requires a properly optimized KV eviction implementation to compute correction using an efficient sparse tensors implementation. This is addressed by Feng et al. (2026), and can be extended to our case. Further, correction requires an additional memory overhead of $\mathcal{O}(k \cdot h \cdot b)$ bytes, where k is the number of kept tokens, h is the total number of attention heads in the LLM, and b is the floating point precision in bytes. Asymptotically, this is dwarfed by the KV cache size itself and not a real concern, as the cache size is $\mathcal{O}(d \cdot k \cdot h \cdot b)$, where d is the embedding dimension.

N AI DISCLOSURE

In this work, we have **not** used generative AI tools for coding, writing, ideation, data generation, proofs, analysis, interpretation, generation of figure, or editing.