Streaming Attention Approximation via Discrepancy Theory

Ekaterina Kochetkova
EPFL
ekaterina.kochetkova@epfl.ch

Kshiteej Sheth
EPFL
kshiteej.sheth@epfl.ch

Insu Han
KAIST
insu.han@kaist.ac.kr

Amir Zandieh
Google Research
zandieh@google.com

Michael Kapralov
EPFL
michael.kapralov@epfl.ch

Abstract

Large language models (LLMs) have achieved impressive success, but their high memory requirements present challenges for long-context token generation. In this paper we study the streaming complexity of attention approximation, a key computational primitive underlying token generation.

Our main contribution is BalanceKV, a streaming algorithm for ϵ-approximating attention computations based on geometric process for selecting a balanced collection of Key and Value tokens as per Banaszczyk's vector balancing theory. We complement our algorithm with space lower bounds for streaming attention computation. Besides strong theoretical guarantees, BalanceKV exhibits empirically validated performance improvements over existing methods, both for attention approximation and end-to-end performance on various long context benchmarks.

1 Introduction

Transformer-based models are the foundation of ongoing artificial intelligence revolution. Their applications span a wide range of domains, from leading-edge language models (LLM) [1, 65] to text-to-image [58, 66, 69], text-to-video synthesis [70], coding assistance [68] and even in multimodal domains across text, audio, image, and video [53]. At the core of these models is the Transformer architecture, powered by the self-attention mechanism [73], which enables effective capture of pairwise correlations across tokens in an input sequence. As these models scale in size and context length [41], they face significant computational challenges, particularly in terms of memory usage. Efficiency and accuracy are essential to unlock the full potential of LLMs in generating long sequences.

Space bottlenecks in transformer models. Most large language models, along with multimodal and video models, adopt an autoregressive, decoder-only architecture. This architecture generates tokens sequentially, applying attention dynamically to each newly generated token. To avoid redundant attention score computations during the generation phase, these models explicitly store the key and value embeddings of previously generated tokens in a cache in each attention layer. Thus, a major challenge is the fact that the memory complexity of storing previously generated key value embeddings scales with both the model size (i.e., the number of layers and attention heads) and, critically, the context size. Additionally, each model session typically requires its own dedicated cache for storing key value embeddings, further exacerbating memory usage. This growing demand has become a significant bottleneck, affecting both memory consumption and computational speed, particularly for models handling long context lengths.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

Streaming attention computation. The main reason for the need of storing the past key and value embeddings is for the attention computation happening inside each self attention layer during token generation after processing a context – to generate the next token, each self attention layer computes the attention between the query embedding of the current token and the key and value embeddings of all the tokens that were previously generated or part of the context. In this paper we study the *streaming attention approximation* problem – the problem of approximately computing attention using a small amount of space, i.e. without storing all previously seen key and value embeddings. Our main contribution is BALANCEKV, a novel provably correct algorithm for streaming attention approximation based on discrepancy theory. The core of our approach is a vector balancing algorithm from discrepancy theory that exploits the geometry of key and value tokens to deduce a small subset of them that well approximates the operations happening inside a self-attention layer. We complement our algorithm with a lower bound on the streaming complexity of approximating attention.

An algorithm for streaming attention approximation can directly be used for compressing the key value cache which stores the past key value embeddings in each layer in an LLM, thus improving the efficiency of LLM token generation. We empirically evaluate BALANCEKV both on the problem of approximating attention and on end-to-end generation tasks, showing performance gains.

1.1 Related Work

For discrepancy theory, Banaszczyk's seminal works [6, 7] establishing theoretical guarantees for vector set discrepancy have sparked research in the vector balancing problem [17]. This led to algorithmic developments in both offline [8] and online [9, 3, 43] settings. The vector balancing problem has particular relevance to streaming and sublinear algorithms, as minimizing a dataset's discrepancy yields small subsets that effectively preserve the original dataset's properties. Recently [55, 15] extend these discrepancy theory ideas for *kernel density estimation* using sublinear memory.

A simple yet effective approach is quantizing previously generated key value embeddings with fewer bits [80, 78, 26, 40, 49, 35, 84, 81]. Another line of work focuses on token-level pruning, where redundant or less important tokens get evicted from the set of all previously generated key value embeddings [10, 85, 48, 76, 83, 46]. Many of the works in this line have used accumulated attention scores to select important previously generated tokens [85, 46, 76]. Recent works extend those methods to an adaptive way of budget allocation across layer [14] and head [30].

1.2 Overview of Our Contributions

In this work we take the token subset selection approach to reduce the memory complexity of LLM token generation: store and maintain only a subset of previously generated key and value embeddings corresponding to a few "important" tokens in the sequence. Of course, the central question is how to define "importance" of tokens. Our approach here is to apply discrepancy theory, which, at a high level, considers a token important if it is crucial to preserving the projection of the total collection of tokens onto some direction in the token space. This leads to the idea of selecting a subset of tokens that is "balanced" simultaneously in every direction. Inspired by the recent breakthrough result of [3] on online discrepancy minimization, we design a method for balancing key-value pairs online using small space, namely our BALANCEKV algorithm. Interestingly, this algorithm is *online*, i.e. the importance of a token is determined only by preceding tokens – in sharp contrast with state of the art heuristics for token selection such as PyramidKV [14] and SnapKV [46], whose performance, as we show, our algorithm matches or improves upon. Our contributions are:

1. In Section 3 we propose BALANCEKV, an algorithm for recursively compressing the set of previously generated tokens using a geometric correlated sampling process based on discrepancy theory. We show that BALANCEKV gives provable guarantees for streaming attention approximation under the bounded ℓ_2 norm assumption (Theorem 3.1). Using tools from communication complexity, we also show a lower bound on the memory complexity of any algorithm for streaming attention approximation in Section 3. Section 2 contains the formal problem formulation of streaming attention approximation, its applicability to key value cache compression, as well as a technical overview of the main results and techniques of Section 3.
2. In Section 4 we empirically evaluate our algorithm in various settings. In Section 4.1 we show our approach leads to a lower relative error for single layer attention approximation for open-source LLMs including `Llama-3.1-8B-Instruct` [27] and `Ministral-8B-Instruct-`

2410 [52] as compared to uniformly sampling keys and values in the cache. Section 4.1 we also perform ablation studies to show how various parameters in our algorithm affect the relative error for single layer attention approximation. In Sections 4.2 and 4.3 we perform end to end experiments on various benchmarks such as LongBench [5] using models of various sizes such as `Llama-3.1-8B-Instruct`,`Qwen-2.5-14B-Instruct` and `Qwen-2.5-32B-Instruct` [77, 71], and Needle in a Haystack [39]. We show that our provable method for attention approximation when applied to key value cache compression performs better compared to previous existing token subset selection heuristics on end to end tasks. Finally in Section 4.4 we present system efficiency metrics regarding our implementation.

2 Technical Overview

In this section, we first set up the formal problem formulation that we tackle, followed by an overview of our techniques and our main results.

2.1 Streaming Attention Approximation: Formulation and Motivation

Autoregressive Transformers generate tokens one by one and each depends on the previously generated tokens. When Transformers process a sequence of tokens, the *attention mechanism* operates by computing three types of embeddings for each token at every layer: query, key and value. The query and key capture how different tokens interact, while the value is the actual content to be aggregated. Such interactions are quantified by so-called *attention scores*, obtained by applying the softmax to the inner product between the query of a given token and the keys of all others. These scores determine how much each previous token's value contributes to the final output. Once the keys and values are computed for a given token, they do not need to be recomputed when generating subsequent tokens.

Formally, suppose that we have a stream of query, key and value embeddings $(q_1, k_1, v_1), \ldots, (q_n, k_n, v_n)$, that is the j-th token is represented as a triplet of (q_j, k_j, v_j) where $q_j, k_j, v_j \in \mathbb{R}^d$ for all $j \in [n]$. Let $K_j, V_j \in \mathbb{R}^{j \times d}$ be matrices defined by stacking those keys and values in their respective rows. To compute the following at every step j to generate $j+1$ token, is called the *streaming attention problem*:

$$\mathrm{Attn}(q_j, K_j, V_j) := \mathrm{softmax}\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right)^T \cdot V_j. \tag{1}$$

Keeping all of the key-value pairs in the cache is prohibitively expensive, especially for long sequences. Instead, we opt for approximate computation by sampling a few key-value pairs. Specifically, our goal is to construct an algorithm that at every time step j computes an estimator z_j for $\mathrm{Attn}(q_j, K_j, V_j)$ in sublinear in n time and memory. In particular for given precision $\varepsilon > 0$, z_j should satisfy the following error constraint:

$$\|z_j - \mathrm{Attn}(q_j, K_j, V_j)\|_2 \leq \varepsilon \left\|\mathrm{softmax}\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right)\right\|_2 \|V_j\|_F. \tag{2}$$

A sublinear in n time and memory algorithm to compute z_j will require knowledge of significantly less key-value pairs than K_j, V_j, thus reducing the size of the key value cache needed to store them. This motivates the study of streaming attention approximation, as an algorithm for this can directly be used for key value cache compression during LLM token generation. In the next section we discuss how we will construct such an estimator z_j at a high level.

2.2 SoftmaxBalance: Attention Approximation via Discrepancy Theory

We now start with presenting the main ideas of our approach. By the definition of softmax, Equation (1) can be written as

$$\mathrm{Attn}(q_j, K_j, V_j) = \frac{1}{Z_j} \exp\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right)^T \cdot V_j,$$

where for a matrix A we write $\exp(A)$ to denote entry-wise exponential function to A and $Z_j := \sum_{i \in [j]} \exp(\langle k_i, q_j \rangle / \sqrt{d})$. Our approach to approximate $\mathrm{Attn}(q_j, K_j, V_j)$ consists of two subroutines which approximate:

1. Softmax normalization $Z_j = \sum_{i\in[j]} \exp(\langle k_i, q_j\rangle/\sqrt{d})$,
2. Matrix-vector product between V_j and $\exp(K_j \cdot q_j/\sqrt{d})$.

To understand our main idea, suppose we are at the end of the stream (i.e., $j = n$) and we store all key-value pairs $(k_1, v_1), \ldots, (k_n, v_n)$. Then for an arbitrary query q_n we aim to approximate the matrix-vector product $\exp(K_n \cdot q_n/\sqrt{d})^T \cdot V_n = \sum_{i\in[n]} \exp(\langle k_i, q_n\rangle/\sqrt{d})v_i$ by choosing a subset of the rows of K_n and V_n of size at most $n/2$ which corresponds to a compression rate of 0.5. Suppose we can design an algorithm which splits the set C of all keys and values into two groups C' and $C\backslash C'$ so that the matrix-vector product function for any query vector q_n is roughly equal over C' and $C\backslash C'$ that is informally,

$$\sum_{\{k,v\}\in C'} \exp\left(\frac{\langle k, q_n\rangle}{\sqrt{d}}\right) v \approx \sum_{\{k,v\}\in C\backslash C'} \exp\left(\frac{\langle k, q_n\rangle}{\sqrt{d}}\right) v.$$

Then, we are able to approximate the matrix-vector product function with either one of the sums above since informally:

$$\sum_{\{k,v\}\in C} \exp\left(\frac{\langle k, q_n\rangle}{\sqrt{d}}\right) v \approx 2 \sum_{\{k,v\}\in C'} \exp\left(\frac{\langle k, q_n\rangle}{\sqrt{d}}\right) v.$$

Therefore, it would suffice to keep the smaller subset of C' and $C\backslash C'$ as the desired subset of key value embeddings and discard the rest. If we wanted to compress the key value cache to a smaller size by a factor 2^T for some T, we would recursively compress the selected subset using the same procedure $T - 1$ more times.

A similar goal is captured by the *vector balancing problem* studied extensively in discrepancy theory; given a set of vectors $C = \{k_1, \ldots, k_n\} \subset \mathbb{R}^d$ with $\|k_j\|_2 \leq 1$ for all j, partition them into two groups $C', C \setminus C'$ such that for any $q \in \mathbb{R}^d$ it holds $\sum_{k\in C'} \langle k, q\rangle \approx \sum_{k\in C\setminus C'} \langle k, q\rangle$ with high probability. The Self-Balancing Walk algorithm [3] is a breakthrough result for the above vector balancing problem. However we need to develop an algorithm for the vector balancing problem with respect to function $\exp(\langle k, \cdot\rangle/\sqrt{d})v$ instead of the inner product function $\langle k, \cdot\rangle$.

Our first contribution is to develop an algorithm for our task, building upon the result from the self-balancing walk [3], which essentially randomly partitions the set of keys and values C into C' and $C \setminus C'$ such that the following holds with high probability under the assumptions that the norms of the query and key embeddings are bounded,

$$\left\| \sum_{\{k,v\}\in C'} \exp\left(\frac{\langle k, q_n\rangle}{\sqrt{d}}\right) v - \sum_{\{k,v\}\notin C'} \exp\left(\frac{\langle k, q_n\rangle}{\sqrt{d}}\right) v \right\|_2 \leq O\left(\log(nd)\right) \cdot \max_{j\in[n]} \|v_i\|_2.$$

We refer to this algorithm as SOFTMAXBALANCE, its formal guarantee is presented in Theorem 3.3 and its pseudocode is presented in Algorithm 2. Theorem 3.3 shows that SOFTMAXBALANCE succeeds to divide C into subsets C' and $C\backslash C'$ which are balanced with respect to function $\exp(\langle k, \cdot\rangle/\sqrt{d})v$ up to an error which only has logarithmic dependence on the size of C. In addition, SOFTMAXBALANCE can accept as input value vectors of arbitrary dimension s. Therefore, if instead of the value vectors $v_1, \ldots, v_n \in \mathbb{R}^d$ we input the set of scalars $v_1 = \cdots = v_n = 1$, we will get an algorithm for the vector balancing problem with respect to function $\exp(\langle k, \cdot\rangle/\sqrt{d})$. This implies that we can use SOFTMAXBALANCE to compress the key value cache to even approximate the softmax normalization $\sum_{i\in[n]} \exp(\langle k_i, q_n\rangle/\sqrt{d})$. We now discuss how to use SOFTMAXBALANCE for streaming attention approximation, i.e. to use it to compute an estimator z_j satisfying Equation (2).

2.3 BALANCEKV: Implementing SOFTMAXBALANCE in Streaming

For a sequence of n tokens and a given memory budget of $t \ll n$, we aim to design a procedure which applies SOFTMAXBALANCE to select from n key-value embeddings a set of at most t in the streaming setting and can compute an estimator z_j satisfying Equation (2) for all steps j in the stream. In the streaming setting one needs to consider the following aspects. As described in the previous section, one iteration of SOFTMAXBALANCE only allows one to select a $n/2$ sized subset of

n key-value embeddings, which is higher than the desired budget of t embeddings. This can be easily mitigated by recursively applying SOFTMAXBALANCE $2^{\log(n/t)}$ times, each time halving the set of key-value embeddings. However, this cannot be implemented in the streaming as we have a limited memory budget of t which prohibits us from storing all key-value embeddings during recursion.

To deal with this, we use the classical merge and reduce technique used in the design of streaming algorithms [13, 51, 33]. MERGEANDREDUCE algorithm is a recursive binary tree-based approach that allows one to implement SOFTMAXBALANCE recursively in a streaming setting with the total memory not exceeding $\widetilde{O}(dt)$, where $\widetilde{O}(\cdot)$ supresses polynomial in $\log n$ factors, under the assumption that the norms of queries and keys are bounded. The guarantees of MERGEANDREDUCE are presented in Theorem 3.4, its pseudocode in Algorithm 4 and a visual representation in Figure 2. If the norms of all value embeddings in the stream are the same up to constant factors, that is for all $i, j \in [n]$ $0.5 \leq \|v_i\|_2/\|v_j\|_2 \leq 2$, then the outputs of MERGEANDREDUCE can be used to construct an estimator z_j satisfying our attention approximation guarantee of equation Equation (2) with precision ε for $t = \widetilde{O}(\sqrt{d}/\varepsilon)$. However, the value embeddings may have very different norms.

Our main algorithm BALANCEKV (pseudocode in Algorithm 1) deals with this issue by grouping the key-value embeddings in the stream according to the norms of the value embeddings, running a separate instance of MERGEANDREDUCE on each group, and combining the outputs of each instance of MERGEANDREDUCE. BALANCEKV constructs a final estimator z_j satisfying Equation (2) with precision ε only using $\widetilde{O}(d\sqrt{d}/\varepsilon)$ memory and $\widetilde{O}(d^2/\varepsilon^2)$ runtime per every step j of the stream, assuming the norms of query and key embeddings are bounded. Existing methods [83] subsample keys and values independently in the cache, and thus have a $1/\varepsilon^2$ dependence on ε in total memory. The guarantees of BALANCEKV are presented in Theorem 3.1.

Finally using the lower bound on the communication complexity of INDEX, we show a lower bound on the memory complexity of any algorithm for streaming attention approximation in Theorem 3.2.

3 Main Theoretical Results

Our main algorithm for streaming attention approximation is BALANCEKV. It takes in as input a stream of n tokens $(q_1, k_1, v_1), (q_2, k_2, v_2), \ldots, (q_n, k_n, v_n)$ and at every step of the stream outputs an estimate z_j to $\text{Attn}(q_j, K_j, V_j)$ (see Equation (1) for the definition of Attn(.)) satisfying Equation (2) with precision ε. Assuming that the ℓ_2 norms of q_j, k_j are at most r for all j, BALANCEKV uses total space $\widetilde{O}(d\sqrt{d}e^{2r^2/\sqrt{d}} \cdot 1/\varepsilon)$ and uses $\widetilde{O}(d^2 e^{4r^2/\sqrt{d}} \cdot 1/\varepsilon^2)$ runtime at each step j of the stream to output z_j. Our main theorem is as follows.

Theorem 3.1. *For any $r, \varepsilon > 0$, any positive integers n, d, any set of tokens $(q_1, k_1, v_1), (q_2, k_2, v_2), \ldots, (q_n, k_n, v_n)$ where $q_j, k_j, v_j \in \mathbb{R}^d$ satisfy $\|q_j\|_2, \|k_j\|_2 \leq r$ for all j, consider an invocation of BALANCEKV with*

$$\textit{batch size } t = \widetilde{O}\left(\sqrt{d}e^{2r^2/\sqrt{d}}/\varepsilon\right) \textit{ and compression rate } 2^{-T} \textit{ with } T = \log(n/t).$$

Then BALANCEKV outputs a vector z_j satisfying Equation (2) with probability at least $1 - 1/poly(n)$ at every step j of the stream. It uses total memory $\widetilde{O}\left(d\sqrt{d}e^{2r^2/\sqrt{d}}/\varepsilon\right)$ across all steps of the stream and runtime of $\widetilde{O}\left(d^2 e^{4r^2/\sqrt{d}}/\varepsilon^2\right)$ per step of the stream.

A pseudocode of BALANCEKV is described in Algorithm 1. At its core BALANCEKV relies on our main discrepancy based algorithm, namely SOFTMAXBALANCE– see Section 3.1 for details on SOFTMAXBALANCE. BALANCEKV uses the output of SOFTMAXBALANCE to compute estimates of the numerator and denominator of $\text{Attn}(q_j, K_j, V_j)$ and returns the desired attention approximation z_j for each streamed index j. There are two subtleties, however. First, it is important to bucket tokens in the stream according to the norm of the value vectors – see lines 5 and 6. Second, a direct application of SOFTMAXBALANCE would require too much memory space. To ensure small space usage, we apply a classical streaming technique, namely the MERGEANDREDUCE algorithm on top of SOFTMAXBALANCE to reduce the space consumption. The space reduction achieved by MERGEANDREDUCE is by running a logarithmic number of copies of SOFTMAXBALANCE in a tree-like fashion. More details are introduced in Section 3.2.

Algorithm 1 BALANCEKV$((q_j, k_j, v_j)_{j=1}^n, r, t, T, \varepsilon)$

input: stream of n tokens (q_j, k_j, v_j), diameter r, batch size t, compression rate 2^{-T}, precision parameter ε.
// Bucket the stream and maintain $\log(n)$ instances of MERGEANDREDUCE, MR-NUMERATOR$_i$, for each bucket to approximate the numerator of Attn(q_j, K_j, V_j); and one instance, MR-DENOMINATOR, to approximate its denominator.
$v_{\max} \leftarrow 0$
repeat
  Find an index i such that $2^i \geq \|v_j\|_2 \geq 2^{i-1}$
  Send (k_j, v_j) as input to MR-NUMERATOR$_i$ // Bucket the stream by $\|v\|_2$
  $v_{\max} \leftarrow \max\{\|v_j\|_2, v_{\max}\}$
  Erase all MR-NUMERATOR$_i$ with $2^i \leq \frac{\varepsilon}{2n} e^{-\frac{r^2}{\sqrt{d}}} v_{\max}$ // Erase small norm buckets
  $C_i^0, \ldots, C_i^T \leftarrow$ the output of MR-NUMERATOR$_i$
  $V^l \leftarrow \cup_i C_i^l$ for $l = 0, \ldots, T$ // Combine the outputs of MR-NUMERATOR$_i$
  Send $(k_j, 1)$ as input to MR-DENOMINATOR
  $K^0, \ldots K^T \leftarrow$ MR-DENOMINATOR
  output: $z_j = \frac{\sum_{l=0}^T 2^l \sum_{\{k,v\} \in V^l} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right) v}{\sum_{l=0}^T 2^l \sum_{\{k,v\} \in K^l} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right)}$
  $j \leftarrow j + 1$
until token stream ends

To summarize, BALANCEKV groups tokens in the stream according to the norms of the corresponding value embeddings, runs a separate instance of MERGEANDREDUCE on each group, and combines the outputs of each instance to construct the final estimate for Attn(q_j, K_j, V_j) at each step $j \in [n]$. Next we present SOFTMAXBALANCE and MERGEANDREDUCE. The full proof of Theorem 3.1 is given in appendix Section A.1. Finally we state the theorem which provides a lower bound on the memory complexity of any algorithm for streaming attention approximation below, its full proof is provided in appendix Section C.

Theorem 3.2. *Suppose that $r^2 \leq d$. Any streaming algorithm which on input $(\{k_1, v_1\}, \ldots, \{k_n, v_n\}, q)$, $\|q\|_2, \|k_i\|_2 \leq r$, outputs z_q satisfying Equation (2) with probability 0.999 has space complexity $\Omega\left(\min\{\frac{1}{\varepsilon^2}, d\exp(2r^2/\sqrt{d})\}\right)$.*

3.1 SOFTMAXBALANCE

We now present our main discrepancy based compression algorithm, SOFTMAXBALANCE. Given a sequence of key and value embeddings $C = \{(k_1, v_1), \ldots (k_n, v_n)\}$ (with key and value embeddings having possibly different dimensions), the goal of SOFTMAXBALANCE is to produce a partition of C into subsets $C', C \setminus C'$ such that for any query $q \in \mathbb{R}^d$ we have that $\sum_{(k,v)\in C'} \exp(\langle k, q\rangle/\sqrt{d})v \approx \sum_{(k,v)\in C\setminus C'} \exp(\langle k, q\rangle/\sqrt{d})v$ with high probability. Without loss of generality assume that $|C'| \leq |C|/2$, we can then output $2\sum_{(k,v)\in C'} \exp(\langle k, q\rangle/\sqrt{d})v$ as an approximation to $\sum_{(k,v)\in C} \exp(\langle k, q\rangle/\sqrt{d})v$, thus achieving a factor 2 compression. Its description is presented in Algorithm 2 below. We note that while SOFTMAXBALANCE takes as input a sequence of key and value embeddings, it can nevertheless be used to compute the softmax normalization: we simply run it on the keys, with the corresponding value vector one-dimensional and all equal to 1 – see line 11 in BALANCEKV, where SOFTMAXBALANCE is called within the corresponding invocation of MERGEANDREDUCE with value vectors as 1s. It's guarantees are as follows.

Theorem 3.3. *Given sets $K = \{k_1, \ldots, k_n\} \subset \mathbb{R}^d, V = \{v_1, \ldots, v_n\} \subset \mathbb{R}^s$, and failure probability $\delta > 0$, define C to be the dataset of pairs $C = \{(k_1, v_1), \ldots, (k_n, v_n)\}$. There exists a randomized algorithm, SOFTMAXBALANCE, which outputs a subset $C' \subset C$, $|C'| \leq |C|/2$, such that, for any*

Algorithm 2 SOFTMAXBALANCE$((k_j, v_j)_j, r_{\text{key}}, r_{\text{value}}, \delta)$

input: stream of $\leq n$ key-value embeddings (k_j, v_j), radii r_{key}, r_{value}: $\max_j \|k_j\|_2 \leq r_{\text{key}}$, $\max_j \|v_j\|_2 \leq r_{\text{value}}$, probability of failure δ.
$R \leftarrow \exp(r_{\text{key}}^2/2\sqrt{d}) \cdot r_{\text{value}}$
$c \leftarrow 30 \log(n/\delta)$
Initialize zero vector $\eta \leftarrow \{0\}$
for j from 1 and until the end of the stream **do**
  $y \leftarrow \left(\exp(\langle k_i, k_j\rangle/\sqrt{d})\langle v_i, v_j\rangle\right)_{i\in[j]}$
  if $\left|y^T\eta\right| > c \cdot R^2$ **then** FAIL
  $p_j \leftarrow \frac{1}{2} - \frac{y^T\eta}{2c\cdot R^2}$
  $\eta_j \leftarrow \begin{cases} +1 & \text{with probability } p_j \\ -1 & \text{o.w.} \end{cases}$
  Add a new zero coordinate $\eta_{j+1} \leftarrow 0$
end for
if $|\{(k_i, v_i) : \eta_i = 1\}| \leq |\{(k_i, v_i) : \eta_i = -1\}|$ **then**
  output: $\{(k_i, v_i) : \eta_i = 1\}$
else
  output: $\{(k_i, v_i) : \eta_i = -1\}$
end if

vector $q \in \mathbb{R}^d$, with probability at least $1 - \delta$,

$$\left\| \sum_{\{k,v\}\in C'} \exp\left(\frac{\langle k, q\rangle}{\sqrt{d}}\right) v - \sum_{\{k,v\}\notin C'} \exp\left(\frac{\langle k, q\rangle}{\sqrt{d}}\right) v \right\|_2 \leq$$
$$O\left(\sqrt{s} \cdot \log(ns/\delta) \cdot \exp\left(\frac{\|q\|_2^2}{2\sqrt{d}}\right) \cdot \exp\left(\max_{j\in[n]} \frac{\|k_j\|_2^2}{2\sqrt{d}}\right) \cdot \max_{j\in[n]} \|v_j\|_2\right).$$

The runtime of SOFTMAXBALANCE *is $O((d + s)n^2)$ and memory is $O((d + s)n)$.*

The proof of the above theorem uses the breakthrough result of [3] for the vector balancing problem, one of the main problems in discrepancy theory. Given a set of vectors $k_1, \ldots, k_n$ the result of [3] produces a subset C of these vectors of at most half the size such that for any vector q we have that $\sum_{k\in C}\langle k, q\rangle \approx \sum_{k\in[n]\setminus C}\langle k, q\rangle$ with high probability. Our main contribution is an algorithm for the vector balancing problem with respect to the function $\exp(\langle k, \cdot\rangle/\sqrt{d})v$ as compared to $\langle k, \cdot\rangle$ in the case of [3]. We defer the proof of Theorem 3.3 to Appendix A.2.

3.2 MERGEANDREDUCE

As briefly mentioned above in Section 3, MERGEANDREDUCE is a streaming version of SOFTMAXBALANCE. The idea is to partition the stream of tokens into batches of size t, apply SOFTMAXBALANCE to the batches to reduce the size of each batch by a constant factor, and then repeat recursively – see Fig. 2 in the appendix.

If we set batch size t to be about $1/\varepsilon$ (see Theorem 3.4 below for the more precise setting), we obtain a streaming algorithm that approximates $\sum_{i=1}^{j} \exp(\langle k_i, q_j\rangle/\sqrt{d})v_i$ at any point j in the stream using total space $\widetilde{O}(d\sqrt{d}e^{2r^2/\sqrt{d}}/\varepsilon)$ and runtime $\widetilde{O}(d^2e^{4r^2/\sqrt{d}}/\varepsilon^2)$ per step, where r is an upper bound on the norms of key and query embeddings.

As before, an important aspect is that MERGEANDREDUCE can handle value embeddings of dimension not necessarily equal to that of key and query embeddings. Thus, when run on scalars $v_i = 1$ for all i, it can also be used to approximate softmax normalization at any point j in the stream. This is the main subroutine used in BALANCEKV to approximate $\text{Attn}(q_j, K_j, V_j)$. Its pseudocode description is presented in Appendix A.3.1, and its proof is in Appendix A.3.2. The formal guarantees are

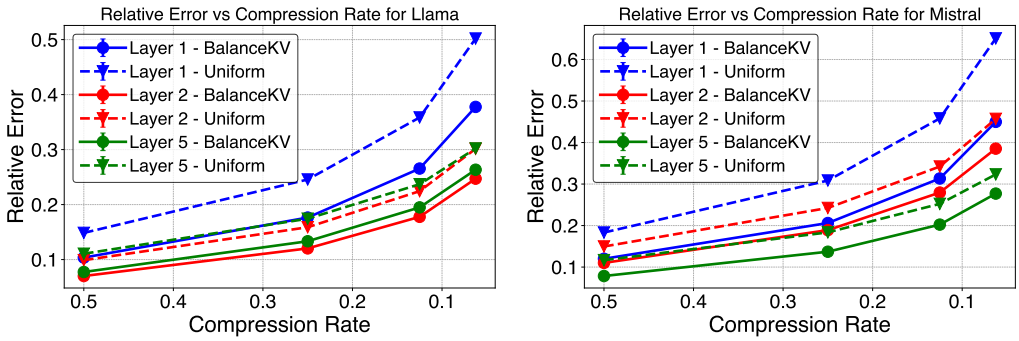

Figure 1: Comparison of relative errors across different layers of `Llama-3.1-8B-Instruct` (left) and `Ministral-8B-Instruct-2410` (right) on TriviaQA dataset.

Theorem 3.4. *For any $r, \varepsilon > 0$, any set of tokens $(q_1, k_1, v_1), \ldots, (q_n, k_n, v_n)$ where $q_j, k_j \in \mathbb{R}^d$ satisfy $\|q_j\|_2, \|k_j\|_2 \leq r$, $v_j \in \mathbb{R}^s$ for $s \leq d$ suppose,*

$$\textit{batch size } t = \widetilde{O}(\sqrt{s}e^{2r^2/\sqrt{d}}/\varepsilon) \textit{ and compression rate } 2^{-T} \textit{ with } T = \log(n/t).$$

Then MERGEANDREDUCE *on input parameters t, r, d, s, ε, outputs at every step j of the stream subsets of key-value embedding pairs $C^0, \ldots, C^T \subset C := \{(k_1, v_1), \ldots, (k_n, v_n)\}$ such that, $z_j := \sum_{i=0}^{T} 2^i \sum_{\{k,v\} \in C^i} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right) v$, satisfies with probability at least $1 - 1/poly(n)$,*

$$\left\| \sum_{i=1}^{j} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right) v_i - z_j \right\|_2 \leq \varepsilon j \cdot e^{-r^2/\sqrt{d}} \cdot \max_{i \in [n]} \|v_i\|_2.$$

Total memory of the algorithm is $\widetilde{O}(d\sqrt{s}e^{2r^2/\sqrt{d}}/\varepsilon)$, its j-th iteration runtime is $\widetilde{O}(dse^{4r^2/\sqrt{d}}/\varepsilon^2)$.

4 Experiments

In this section we now present our experimental results. The full details of all sections as well as the experimental setup and implementation can be found in Appendix B.

4.1 Ablation Studies on Single Layer Attention Approximation

We evaluate the effectiveness of BALANCEKV for approximating attention in individual layers of `Llama-3.1-8B-Instruct` [27] and `Ministral-8B-Instruct-2410` [52] on the TriviaQA dataset from LongBench [5]. Specifically, we examine layers 1, 2, and 5 and compare against independent uniform sampling key and value embeddings.

Due to space limitations, we provide the full experimental details in Appendix B.1. For each layer, we approximate attention for recent tokens using a compressed cache that retains a fixed number of initial and recent embeddings, alongside intermediate ones selected via BALANCEKV, and measure its relative error against exact attention. We vary the compression rate $2^{-T} \in \{1/2, 1/4, 1/8, 1/16\}$. As shown in Fig. 1, BALANCEKV consistently yields lower relative approximation error than approximating attention by uniform sampling past key value pairs across all settings, empirically validating its advantage as predicted by Theorem 3.1.

For a fixed dataset and layer, we also analyzed how the performance and runtime of BALANCEKV depend on the batch size and compression rate. More precisely, we repeat the single-layer attention approximation experiment TriviaQA and layers 1 and 15 of `Llama-3.1-8B-Instruct`, for batch size $\in [64, 128, 256]$ and compression rate $2^{-T} \in [1/2, 1/4, 1/8]$. The results are presented in Figure 3. As this experiment suggests, the quality of attention approximation increases as the size of the block doubles, while the runtime becomes slower as also proven theoretically.

Method	qasper	multi	hotpotqa	2wiki	gov	multinews	trec	triviaqa	samsum	p.count	p.ret	lcc	repo-p	average
Qwen2.5-32B-Instruct														
Exact (Baseline)	44.56	50.65	69.14	60.39	21.3	19.52	75.33	81.14	43.17	22.0	99.67	50.9	35.22	51.77
StreamingLLM	20.12	34.35	51.84	48.23	19.09	17.10	61.00	51.14	28.52	23.33	41.33	39.19	26.54	35.52
PyramidKV	34.47	46.33	67.78	55.92	15.16	15.39	69.33	63.24	40.36	22.67	99.33	48.32	34.35	47.13
SnapKV	36.21	46.78	66.64	57.02	16.35	16.07	70.33	77.53	41.08	22.0	99.33	49.04	35.62	48.77
Uniform	39.28	43.82	64.82	57.84	23.10	19.50	73.00	81.63	39.70	22.00	92.00	44.97	32.19	48.76
BALANCEKV	40.14	43.17	64.46	58.06	22.26	20.32	73.00	80.68	41.07	22.33	92.0	44.95	32.43	**48.84**
Qwen2.5-14B-Instruct														
Exact (Baseline)	43.39	52.63	64.06	53.71	28.07	22.4	74.67	88.75	44.81	22.33	99.0	63.69	46.3	54.14
StreamingLLM	20.73	32.62	49.93	42.39	21.63	18.69	59.67	74.96	29.57	11.67	63.0	46.16	32.25	38.71
PyramidKV	31.76	46.6	62.83	50.0	19.08	17.68	65.0	85.52	42.61	22.0	99.33	60.62	44.4	49.8
SnapKV	32.95	47.53	61.96	50.34	20.29	18.28	60.67	88.75	42.97	22.67	99.33	61.32	45.84	50.22
Uniform	37.07	41.69	61.11	49.96	29.18	22.67	71.67	87.89	40.34	22.0	84.33	58.0	42.57	49.88
BALANCEKV	37.02	41.96	61.74	50.9	29.26	22.64	71.67	88.1	41.14	23.67	87.67	58.64	43.63	**50.62**
Llama-3.1-8B-Instruct														
Exact (Baseline)	42.87	48.54	52.05	38.6	31.31	22.07	71.67	91.85	42.36	20.37	98.13	49.62	42.73	50.17
StreamingLLM	20.65	30.71	39.14	32.43	23.10	18.70	58.00	83.87	28.85	20.36	97.26	33.69	30.46	39.79
PyramidKV	33.86	39.75	47.12	35.96	20.03	17.78	63.67	90.76	40.21	20.4	98.97	45.25	39.51	45.64
SnapKV	33.91	42.55	49.09	36.13	20.48	17.67	62.0	91.7	40.23	20.33	98.86	46.7	39.86	46.12
Uniform	37.18	37.15	47.49	37.82	27.56	21.06	68.67	90.48	36.13	20.33	96.26	45.72	37.09	46.38
BALANCEKV	35.75	37.04	46.37	36.24	27.09	20.84	69.0	90.88	37.88	20.39	96.65	48.45	41.4	**46.77**

Table 1: Comparison of various cache compression methods on LongBench-E using various models. The best results among compression methods for each model are highlighted in bold.

4.2 End-to-end Evalution on LongBench

Next we evaluate BALANCEKV on LongBench dataset [5], which tests long-context understanding across tasks like QA, summarization, few-shot learning, synthetic reasoning, and code completion. Specifically, we test a version of uniform length distribution (LongBench-E). During inference, we compress the key-value cache in the prefill stage using a uniform compression rate of (approximately) 0.25 across all methods, while retaining all streamed embeddings during the decoding phase. We compare against StreamingLLM [76], SnapKV [46], PyramidKV [14], and uniform sampling (see Section 4.1), using their implementations from MInference [37]. The evaluation follows the LongBench protocol, using three pre-trained models at different scales including `Llama-3.1-8B-Instruct`, `Qwen-2.5-14B-Instruct` and `Qwen-2.5-32B-Instruct`. The results are reported in Table 1.

Notably, BALANCEKV consistently achieves the best overall performance among compression methods and across all models, demonstrating its effectiveness in preserving model quality for cache compression. The full experimental setup details can be found in Appendix B.2.

4.3 Needle-In-A-Haystack Benchmark

We evaluate BALANCEKV on the "Needle-In-A-Haystack" benchmark [39], comparing it against SnapKV, PyramidKV, StreamingLLM, and uniform sampling using `Llama-3.1-8B-Instruct`. The test challenges the model to retrieve a specific sentence (the "needle") embedded at an arbitrary position within a long context (the "haystack"). Following the setup in [29], we hide the needle at varying depths, from 0% to 100% of the total context length, across documents ranging from approximately 4K to 100K tokens. As in the previous experiments, all methods are evaluated under a fixed compression ratio of approximately 0.25.

To further enhance performance, we introduce an augmented version of BALANCEKV that deterministically preserves a small set of tokens whose key embeddings are strongly anti-correlated with the rest. The standard BALANCEKV procedure is then applied to the remaining tokens within each layer. As a result, BALANCEKV achieves an average accuracy of 0.99, outperforming SnapKV (0.83), PyramidKV (0.90), StreamingLLM (0.31), and uniform sampling (0.90). Detailed heatmaps of performance across different context lengths and needle depths are in Figure 4 in Appendix B.3.

4.4 System Efficiency Metrics

We measure wall-clock times for both the prefill stage (including cache compression) and the decoding stage using a random input of length 16,384 tokens, followed by the generation of 1,024 tokens.

Results are averaged over 10 independent runs, with the minimum runtime reported to enhance robustness. The full results are provided in Table 2 in Appendix B.4.

All compression methods incur some prefill overhead compared to the uncompressed baseline (Exact). While StreamingLLM achieves the fastest decoding speed, it suffers from significantly lower accuracy (see Table 1). Among the remaining methods, BALANCEKV achieves the lowest prefill latency and consistently delivers the best trade-off between efficiency and accuracy. This demonstrates that our discrepancy-based approach not only scales well in theory but also brings practical gains in end-to-end system performance, making it a compelling choice for real-world deployment scenarios.

4.5 Additional Experiments

We additionally conduct the following experiments and provide their results in Appendix B.5 due to the space limitation.

1. Our main theorem (Theorem 3.4) relies on an upper bound of ℓ_2 norms of both query and key vectors. To validate this, we investigate the ℓ_2 norms of queries, keys, and values (QKV) on the TriviaQA dataset from LongBench [5] using `Llama-3.1-8B-Instruct`. Specifically, we analyze prompts in TriviaQA and compute the average ℓ_2 norms of all QKV vectors across all layers and attention heads during the prefill stage. The key findings are that all QKV norms consistently concentrate around some constants (15 for query, 15 for key, and 3 for value) with small confidence intervals (CI). Importantly, the norms remain stable across a wide range of sequence lengths, suggesting that these norms do not grow with input sequence length.
2. We perform the evaluation of BALANCEKV and the uniform sampling when applied to the `InternVL2.5-8B` multimodal LLM for compression rates 1/4 and 1/16, for evaluation on the MS COCO image captioning dataset. The experiment was run on a NVIDIA A100 GPU with 80 GB VRAM.
3. We repeat the experiment in Section 4.2 in the extremely low compression rate regime on some of the datasets from LongBench [5]. More specifically, we compress the key-value cache in the prefill stage of inference using a uniform compression rate of (approximately) 0.8, 0.9, and 0.95 with uniform sampling as well as BALANCEKV, while retaining all streamed embeddings during the decoding phase. BALANCEKV demonstrates improved performance over uniform sampling across each of the compression rates and datasets.
4. We augment Section 4.2 by adding comparison to ClusterGen [83] using `Llama-3.1-8B-Instruct`. The results are reported in Table 6.

5 Conclusion

We propose BALANCEKV, a token pruning method grounded in discrepancy theory. BALANCEKV enables approximate attention computation, which we both establish theoretically and validate empirically. To demonstrate the effectiveness of BALANCEKV as a KV cache compression algorithm, we conduct end-to-end experiments on a range of popular benchmarks and models of varying sizes. Finally, our work introduces a theoretical problem of optimal streaming attention space complexity.

References

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

[2] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebron, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4895–4901, 2023.

[3] Ryan Alweiss, Yang P. Liu, and Mehtaab Sawhney. Discrepancy minimization via a self-balancing walk. *Proceedings of the 53rd ACM Symposium on the Theory of Computing (STOC '2021)*, 2021.

[4] Saleh Ashkboos, Amirkeivan Mohtashami, Maximilian L Croci, Bo Li, Pashmina Cameron, Martin Jaggi, Dan Alistarh, Torsten Hoefler, and James Hensman. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

[5] 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. *arXiv preprint arXiv:2308.14508*, 2023.

[6] Wojciech Banaszczyk. Balancing vectors and gaussian measures of n-dimensional convex bodies. *Random Structures and Algorithms 12 (1998), 351–360*, 1998.

[7] Wojciech Banaszczyk. On series of signed vectors and their rearrangements. *Random Structures and Algorithms 40 (2012), 301–316*, 2012.

[8] Nikhil Bansal. Constructive algorithms for discrepancy minimization. *51th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2010), arXiv:1002.2259*, 2010.

[9] Nikhil Bansal, Haotian Jiang, Sahil Singla, and Makrand Sinha. Online vector balancing and geometric discrepancy. *In Proceedings of the 52nd Annual ACM Symposium on Theory of Computing (STOC '2020), arXiv:1912.03350*, 2019.

[10] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[11] Aline Bessa, Majid Daliri, Juliana Freire, Cameron Musco, Christopher Musco, Aécio Santos, and Haoxiang Zhang. Weighted minwise hashing beats linear sketching for inner product estimation. In *Proceedings of the 42nd ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '23, page 169–181, New York, NY, USA, 2023. Association for Computing Machinery.

[12] Stéphane Boucheron, Gábor Lugosi, and Olivier Bousquet. Concentration inequalities. In *Summer school on machine learning*, pages 208–240. Springer, 2003.

[13] Vladimir Braverman, Avinatan Hassidim, Yossi Matias, Mariano Schain, Sandeep Silwal, and Samson Zhou. Adversarial robustness of streaming algorithms through importance sampling. *Advances in Neural Information Processing Systems*, 34:3544–3557, 2021.

[14] Zefan Cai, Yichi Zhang, Bofei Gao, Yuliang Liu, Tianyu Liu, Keming Lu, Wayne Xiong, Yue Dong, Baobao Chang, Junjie Hu, et al. Pyramidkv: Dynamic kv cache compression based on pyramidal information funneling. *arXiv preprint arXiv:2406.02069*, 2024.

[15] Moses Charikar, Michael Kapralov, and Erik Waingarten. A quasi-monte carlo data structure for smooth kernel evaluations. *In Proceedings of the 35th ACM-SIAM Symposium on Discrete Algorithms (SODA '2024), arXiv:2401.02562*, 2024.

[16] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.

[17] Daniel Dadush, Aleksandar Nikolov, Kunal Talwar, and Nicole Tomczak-Jaegermann. Balancing vectors in any norm. *59th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2018)*, 2018.

[18] Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

[19] Majid Daliri, Juliana Freire, Christopher Musco, Aécio Santos, and Haoxiang Zhang. Sampling methods for inner product sketching. *Proc. VLDB Endow.*, 2024.

[20] Majid Daliri, Juliana Freire, Christopher Musco, Aécio Santos, and Haoxiang Zhang. Sampling methods for inner product sketching, 2024.

[21] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[22] Sanjoy Dasgupta and Anupam Gupta. An elementary proof of a theorem of johnson and lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003.

[23] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332, 2022.

[24] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

[25] Tim Dettmers, Ruslan Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. Spqr: A sparse-quantized representation for near-lossless llm weight compression. *arXiv preprint arXiv:2306.03078*, 2023.

[26] Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. Qaq: Quality adaptive quantization for llm kv cache. *arXiv preprint arXiv:2403.04643*, 2024.

[27] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[28] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[29] Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*, 2024.

[30] Yu Fu, Zefan Cai, Abedelkadir Asi, Wayne Xiong, Yue Dong, and Wen Xiao. Not all heads matter: A head-level kv cache compression method with integrated retrieval and reasoning. *arXiv preprint arXiv:2410.19258*, 2024.

[31] Jianyang Gao and Cheng Long. Rabitq: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data*, 2(3):1–27, 2024.

[32] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 2023. https://github.com/EleutherAI/lm-evaluation-harness.

[33] Mina Ghashami, Edo Liberty, Jeff M. Phillips, and David P Woodruff. Frequent directions: Simple and deterministic matrix sketching. *SIAM J. Comput., 45(5):1762–1792*, 2016.

[34] Insu Han, Rajesh Jarayam, Amin Karbasi, Vahab Mirrokni, David Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. *arXiv preprint arXiv:2310.05869*, 2023.

[35] 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. *arXiv preprint arXiv:2401.18079*, 2024.

[36] Jianqiu Ji, Jianmin Li, Shuicheng Yan, Bo Zhang, and Qi Tian. Super-bit locality-sensitive hashing. *Advances in neural information processing systems*, 25, 2012.

[37] Huiqiang Jiang, YUCHENG LI, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[38] William B Johnson, Joram Lindenstrauss, and Gideon Schechtman. Extensions of lipschitz maps into banach spaces. *Israel Journal of Mathematics*, 54(2):129–138, 1986.

[39] Greg Kamradt. Needle in a haystack-pressure testing llms. *Github Repository*, page 28, 2023.

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

[41] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[42] Junhyuck Kim, Jongho Park, Jaewoong Cho, and Dimitris Papailiopoulos. Lexico: Extreme kv cache compression via sparse coding over universal dictionaries. *arXiv preprint arXiv:2412.08890*, 2024.

[43] Janardhan Kulkarni, Victor Reis, and Thomas Rothvoss. Optimal online discrepancy minimization. *In Proceedings of the 56th Annual ACM Symposium on Theory of Computing (STOC '2024), arXiv:2308.01406*, 2023.

[44] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[45] Dacheng Li, Rulin Shao, Anze Xie, Ying Sheng, Lianmin Zheng, Joseph Gonzalez, Ion Stoica, Xuezhe Ma, and Hao Zhang. How long can open-source llms truly promise on context length?, 2023. https://huggingface.co/lmsys/longchat-7b-v1.5-32k.

[46] 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. *arXiv preprint arXiv:2404.14469*, 2024.

[47] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration. *arXiv preprint arXiv:2306.00978*, 2023.

[48] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36, 2024.

[49] Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[50] Namiko Matsumoto and Arya Mazumdar. Binary iterative hard thresholding converges with optimal number of measurements for 1-bit compressed sensing. *J. ACM*, 71(5), October 2024.

[51] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program., 2(2):143–152*, 1982.

[52] Mistral AI team. Mistral ai, 2024. https://mistral.ai/news/ministraux/.

[53] OpenAI. Introducing gpt-4o, 2024. https://openai.com/index/hello-gpt-4o/.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[55] Jeff M Phillips and Wai Ming Tai. Near-optimal coresets for kernel density estimates. *Discrete and Computational Geometry, 63(4):867–887*, 2020.

[56] Yaniv Plan, Roman Vershynin, and Elena Yudovina. High-dimensional estimation with geometric constraints. *Information and Inference: A Journal of the IMA*, 6(1):1–40, 2017.

[57] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

[58] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

[59] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[60] Nataniel Ruiz, Yuanzhen Li, Varun Jampani, Yael Pritch, Michael Rubinstein, and Kfir Aberman. Dreambooth: Fine tuning text-to-image diffusion models for subject-driven generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22500–22510, 2023.

[61] Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024.

[62] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *arXiv preprint arXiv:1911.02150*, 2019.

[63] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[64] Hanshi Sun, Li-Wen Chang, Wenlei Bao, Size Zheng, Ningxin Zheng, Xin Liu, Harry Dong, Yuejie Chi, and Beidi Chen. Shadowkv: Kv cache in shadows for high-throughput long-context llm inference. *arXiv preprint arXiv:2410.21465*, 2024.

[65] Antropic Team. claude, 2024. https://www.anthropic.com/news/claude-3-family.

[66] FireFly Team. Adobe firefly, 2023. https://firefly.adobe.com/.

[67] Llama3 Team. Llama3, 2024. https://github.com/meta-llama/llama3.

[68] Microsoft Copilot Team. Microsoft copilot, 2023. https://github.com/features/copilot.

[69] Midjourney Team. Midjourney, 2022. https://www.midjourney.com/home.

[70] OpenAI Team. Sora: Creating video from text, 2024. https://openai.com/index/sora/.

[71] Qwen Team. Qwen2.5: A party of foundation models, September 2024.

[72] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. 2017.

[74] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. arxiv. *arXiv preprint arXiv:1910.03771*, 2019.

[75] David Woodruff. Cs 15-859: Algorithms for big data - lecture 11. https://www.cs.cmu.edu/afs/cs/user/dwoodruf/www/teaching/15859-fall20/Scribe_Lecture_11-1.pdf?utm_source=chatgpt.com, 2020.

[76] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

[77] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.

[78] June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization. *arXiv preprint arXiv:2402.18096*, 2024.

[79] Felix Xinnan X Yu, Ananda Theertha Suresh, Krzysztof M Choromanski, Daniel N Holtmann-Rice, and Sanjiv Kumar. Orthogonal random features. *Advances in neural information processing systems*, 29, 2016.

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

[81] Amir Zandieh, Majid Daliri, and Insu Han. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. *arXiv preprint arXiv:2406.03482*, 2024.

[82] Amir Zandieh, Insu Han, Majid Daliri, and Amin Karbasi. KDEformer: Accelerating transformers via kernel density estimation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 40605–40623. PMLR, 23–29 Jul 2023.

[83] Amir Zandieh, Insu Han, Vahab Mirrokni, and Amin Karbasi. Subgen: Token generation in sublinear time and memory. *arXiv preprint arXiv:2402.06082*, 2024.

[84] Tianyi Zhang, Jonah Yi, Zhaozhuo Xu, and Anshumali Shrivastava. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *arXiv preprint arXiv:2405.03917*, 2024.

[85] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Theorem 3.1 provides the main theorem for the performance of our algorithm.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: In Section 4.3 we point out that an enhancement of our main algorithm is needed for better performance on the Needle-In-a-Haystack experiments.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

Justification: Please refer to Section 3 and the proofs of the claims made there to find the full set of assumptions made and the full proofs.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: Please refer to Section 4 and the links inside to find details of all experiments.

   Guidelines:

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

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

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

Answer: [Yes]

Justification: The code for all experiments is provided in the supplementary material.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: Please refer to Section 4 for details regarding the experiments.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: The experiments conducted in Section 4 have been performed with error bars whenever applicable.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: Please refer to Section 4 and the links therein to find the details of all details.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: All studies conducted in this paper conform with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: The paper is fundamental in nature and has no direct path for societal risks or consequences. While the motivation comes from large deep learning models, which of course have myriad such potential ramifications, our paper does not add to those ramifications.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [NA]

    Justification: As discussed in answer to the previous question, no specific safeguards are necessary for the work described in this paper.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [NA]

    Justification: Existing assets are used in adherence to their licenses and usage terms and with appropriate credit to their creators.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [NA]

    Justification: No new assets.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: No research with human subjects.

    Guidelines:

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

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

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

    Answer: [NA]

    Justification: No IRB approval needed for the research described in this paper.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

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

Answer: [NA]

Justification: We only used LLMs for visualizing results for submission and facilitating or running experiments.

Guidelines:

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

A Full Proofs

A.1 Proof of Theorem 3.1

Finally equipped with Theorem 3.3 and Theorem 3.4 we now state the proof of the main Theorem 3.1.

Proof of Theorem 3.1. Recall that BALANCEKV approximates attention by finding good estimations for the numerator and the denominator of attention separately. In line (8), we erase those terms from the numerator, whose value vectors have sufficiently small ℓ_2 norms. In what follows, we:

- Bound the space and time requirements of BALANCEKV, as well as it's probability of failure. Each of the bounds readily follows from Theorem 3.4,
- Bound the contribution of the erased terms to attention,
- Show that BALANCEKV (or, more concretely, procedures MR-NUMERATOR$_i$) approximate the rest of the terms of the numerator well,
- Show that BALANCEKV (its subroutine MR-DENOMINATOR) approximates the denominator of attention well.

We begin by analyzing the time/space requirements and probability of success of BALANCEKV. It never runs MR-NUMERATOR$_i$ for $i > \log_2(v_{\max})$ and $i < \log_2(\varepsilon \cdot n^{-1/2} \cdot v_{\max})$, so it never keeps more than $\log_2(\sqrt{n}/\varepsilon) = O(\log(n))$ of them. BALANCEKV at any step j performs one iteration of procedure MR-DENOMINATOR, one iteration of MR-NUMERATOR$_i$ with $2^i \geq \|v_j\|_2 \geq 2^{i-1}$, computes the subsets $K_1, \ldots, K_T, V_1, \ldots, V_T$ and computes a function of the selected points in the subsets in line. Therefore, the runtime of BALANCEKV during one iteration is bounded by the maximum of the runtime of MERGEANDREDUCE during one iteration and time to compute the output. The latter is equal to its total memory. This maximum, by definition of t, is equal to $\widetilde{O}(d^2 e^{4r^2/\sqrt{d}}/\varepsilon^2)$. The memory of the algorithm is the union of memory of MR-NUMERATOR$_i$ for all i and MR-DENOMINATOR, so the space complexity of the algorithm is $\widetilde{O}(d\sqrt{d}e^{2r^2/\sqrt{d}}/\varepsilon)$. The failure probability is bounded by union bounding the failure probabilities of all instances of MERGEANDREDUCE and at most n queries in the stream, and is equal to $1/\mathrm{poly}(n)$.

Next, we bound the contribution of the erased terms. Formally, let $v_{\max}(j) := \max_{i \leq j} \|v_i\|_2$ and define $i(j) := \max_i \left\{ 2^i \leq \frac{\varepsilon}{\sqrt{n}} v_{\max} \right\}$. Observe that

$$
\begin{aligned}
\left\| \frac{\sum_{\substack{i:i\leq j \\ \|v_i\|_2 \leq 2^{i(j)}}} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right) v_i}{\sum_{i=1}^{j} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right)} \right\|_2 &\leq \frac{\sum_{\substack{i:i\leq j \\ \|v_i\|_2 \leq 2^{i(j)}}} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right) \|v_i\|_2}{\sum_{i=1}^{j} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right)} && \text{by triangle inequality} \\
&\leq \frac{\varepsilon}{2\sqrt{n}} \cdot \frac{\sum_{\substack{i:i\leq j \\ \|v_i\|_2 \leq 2^{i(j)}}} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right)}{\sum_{i=1}^{j} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right)} && \text{by the definition of } i(j) \\
&\leq \frac{\varepsilon}{2\sqrt{n}} v_{\max} \\
&\leq \varepsilon \cdot \left\| \mathrm{softmax}\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right) \right\|_2 \cdot \|V_j\|_F,
\end{aligned}
\tag{3}
$$

where the last inequality follows from the general inequality $\left\| \mathrm{softmax}\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right) \right\|_2 \geq \frac{1}{\sqrt{j}} \geq \frac{1}{\sqrt{n}}$ and $\|V_j\|_F \geq v_{\max}$.

We analyze the quality of approximation of the denominator together with the quality of approximation of the numerator, as both follow from Theorem 3.4. At time step j, procedure MR-DENOMINATOR

returns subsets $K_1, \ldots, K_T$ such that

$$\left| \sum_{i \in [j]} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right) - \sum_{l=0}^{T} 2^l \sum_{\{k,v\} \in K^l} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right) \right| \leq \varepsilon \cdot j \cdot e^{\frac{-r^2}{\sqrt{d}}} \leq \varepsilon \cdot \sum_{i \in [j]} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right) \tag{4}$$

as follows from Theorem 3.4 by plugging in scalars $v_1 = \ldots = v_j = 1$. Next, define $P_{i,j} = \{\{k_l, v_l\} : l \leq j, 2^i \geq \|v_l\|_2 \geq 2^{i-1}\}$. Intuitively, $P_{i,j}$ aggregates all of the tokens proccessed by MR-NUMERATOR$_i$ which appeated before time step j. MR-NUMERATOR$_i$ returns subsets $V_i^1, \ldots, V_i^T \subseteq P_{i,j}$ for all i such that,

$$\begin{aligned} &\left\| \sum_{\{k,v\} \in P_{i,j}} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right) v - \sum_{l=0}^{T} 2^l \sum_{\{k,v\} \in V_i^l} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right) v \right\|_2 \\ &\leq \varepsilon |P_{i,j}| \cdot e^{-r^2/\sqrt{d}} \cdot 2^i \leq \frac{\varepsilon |P_{i,j}| \cdot 2^i}{\sqrt{j}} \cdot \left\| \exp\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right) \right\|_2, \end{aligned}$$

as follows from Theorem 3.4 observing that $\max_{v:\{k,v\} \in P_{i,j}} \|v\|_2 \leq 2^i$. The last inequality holds because $\exp\left(\frac{\langle k,q \rangle}{\sqrt{d}}\right) \geq e^{-r^2/\sqrt{d}}$ and, therefore, $\left\| \exp\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right) \right\|_2 \geq \sqrt{j} \cdot e^{-r^2/\sqrt{d}}$.

Now, observe that $\|V_j\|_F \leq \sqrt{\sum_i |P_{i,j}| \cdot 2^{2i}}$ and $j = \sum_i |P_{i,j}|$. By the Cauchy-Schwartz inequality,

$$\sum_i |P_{i,j}| \cdot 2^i \leq \sqrt{\sum_i |P_{i,j}| \cdot 2^{2i}} \cdot \sqrt{\sum_i |P_{i,j}|}. \tag{5}$$

By triangle inequality, the sum of the outputs of procedures MR-NUMERATOR$_i$ approximates the sum of the terms in the numerator of attention that were not erased in line (8) up to an additive error

$$\frac{\varepsilon}{\sqrt{j}} \cdot \left\| \exp\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right) \right\|_2 \sum_i |P_{i,j}| \cdot 2^i \leq \varepsilon \cdot \left\| \exp\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right) \right\|_2 \|V_j\|_F, \tag{6}$$

where the last inequality follows from Equation (5).

It remains to show how the statement of the theorem follows from the derived bounds. Consider the following abstract derivation. Let u and u' be vectors such that $\|u - u'\|_2 \leq \alpha$, and let b and b' be positive numbers such that

$$\frac{1}{1+\gamma} \cdot \frac{1}{b} \leq \frac{1}{b'} \leq \frac{1}{1-\gamma} \cdot \frac{1}{b}.$$

Then, by application of triangle inequalities,

$$\begin{aligned} \left\| \frac{u}{b} - \frac{u'}{b'} \right\|_2 &\leq \frac{1}{b'} \cdot \|u - u'\|_2 + \|u\|_2 \cdot \left| \frac{1}{b} - \frac{1}{b'} \right| \\ &\leq \frac{\alpha}{1-\gamma} \cdot \frac{1}{b} + \|u\|_2 \cdot \left(\frac{1}{1-\gamma} - 1 \right) \cdot \frac{1}{b} = \\ &= \frac{\alpha}{1-\gamma} \cdot \frac{1}{b} + \|u\|_2 \cdot \frac{\gamma}{1-\gamma} \cdot \frac{1}{b}. \end{aligned} \tag{7}$$

From Equation (4),

$$\frac{1}{1+\varepsilon} \cdot \frac{1}{\sum_{i \in [j]} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right)} \leq \frac{1}{\sum_{l=0}^{T} 2^l \sum_{\{k,v\} \in V_i^l} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right)} \leq \frac{1}{1-\varepsilon} \cdot \frac{1}{\sum_{i \in [j]} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right)}. \tag{8}$$

For simplicity of notation, let $D \subseteq [j]$ denote the subset of indices of all tokens discarded in line (8). Take $\gamma = \varepsilon, b = \sum_{i \in [j]} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right)$, $b' = \sum_{l=0}^{T} 2^l \sum_{\{k,v\} \in V_i^l} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right)$, $u = \sum_{i \in [j] \setminus D} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right) v_i$, $u' = \sum_{l=0}^{T} 2^l \sum_{\{k,v\} \in V_i^l} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right) v$ and, finally, $\alpha = \varepsilon \cdot$

$\left\|\exp\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right)\right\|_2 \|V_j\|_F$. The first of the preconditions of our derivation holds by Equation (6). The second one holds by Equation (8). Hence,

$$\left\| \frac{\sum_{l=0}^{T} 2^l \sum_{\{k,v\}\in V^l} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right) v}{\sum_{l=0}^{T} 2^l \sum_{\{k,v\}\in K^l} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right)} - \frac{\sum_{i\in[j]\setminus D} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right) v_i}{\sum_{i\in[j]} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right)} \right\|_2 \leq \frac{2\varepsilon}{1-\varepsilon} \cdot \left\|\text{softmax}\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right)\right\|_2 \cdot \|V_j\|_F,$$

since $\|u\|_2 = \left\|\sum_{i\in[j]\setminus D} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right) v_i\right\|_2 \leq \left\|\exp\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right)\right\|_2 \|V_j\|_F$.

Finally, combining the above with Equation (3) via triangle inequality, we get

$$\left\| \frac{\sum_{l=0}^{T} 2^l \sum_{\{k,v\}\in V^l} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right) v}{\sum_{l=0}^{T} 2^l \sum_{\{k,v\}\in K^l} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right)} - \text{Attn}(q_j, K_j, V_j) \right\|_2 \leq \frac{2\varepsilon}{1-\varepsilon} \left\|\text{softmax}\left(\frac{K_j \cdot q_j}{\sqrt{d}}\right)\right\|_2 \|V_j\|_F .$$

By rescaling $\varepsilon \to \varepsilon/4$, we get the desired approximation Equation (2). □

A.2 Theoretical Guarantees of SoftmaxBalance

Algorithm Self-Balancing Walk introduced in [3] receives as input vectors $u_1, \ldots, u_n$ and selects signs for them so that, for any direction, the signed sum of the vectors is balanced along that direction with high probability. The following theorem readily follows from theorem 1.1 in [3]:

Theorem A.1 (Theorem 1.1 in [3])**.** *For any $n, d \in \mathbb{N}$, there exists a randomized algorithm which receives as input a set of vectors $U = \{u_1, \ldots, u_n\} \in \mathbb{R}^d$ and a parameter $\delta > 0$. The algorithm outputs a (random) subset $U' \subset U$ such that, for any vector $u \in \mathbb{R}^d$, with probability at least $1 - \delta$,*

$$\left|\sum_{i\in U'} \langle u_i, u\rangle - \sum_{i\notin U'} \langle u_i, u\rangle\right| \leq O\left(\log(n/\delta) \cdot \max_{i\in[n]} \|u_i\|_2 \cdot \|u\|_2\right).$$

Algorithm 3 Self-Balancing Walk $((u_j)_j, r, \delta)$

input: stream of $\leq n$ vectors u_j, radius r: $\max_j \|u_j\|_2 \leq r$, probability of failure δ.
$c \leftarrow 30\log(n/\delta)$
$U_-, U_+ \leftarrow \emptyset$
for i from 1 and until the end of the stream **do**
  if $\left|\sum_{u\in U_+} \langle u, u_i\rangle - \sum_{u\in U_-} \langle u, u_i\rangle\right| > c \cdot r^2$ **then**
    Fail
  end if
  $p_i \leftarrow \frac{1}{2} - \frac{\sum_{u\in U_+} \langle u, u_i\rangle - \sum_{u\in U_-} \langle u, u_i\rangle}{2c\cdot r^2}$
  $\varepsilon_i \leftarrow +$ with probability p_i, and $\varepsilon_i \leftarrow -$ with probability $1 - p_i$
  $U_{\varepsilon_i} \leftarrow U_{\varepsilon_i} \cup \{k_i\}$
end for
if $|U_+| \leq |U_-|$ **then**
  output: U_+
else
  output: U_-
end if

Proof of Theorem 3.3. Define for any $k \in \mathbb{R}^d$ an embedding function $\varphi(k)$:

$$\varphi(k) = \left(\frac{(k/d^{0.25})^{\otimes i}}{\sqrt{i!}}\right)_{i\geq 0}.$$

It is easy to see that for any two vectors $k, q \in \mathbb{R}^d$

$$\langle \varphi(k), \varphi(q) \rangle = \exp\left(\frac{\langle k, q \rangle}{\sqrt{d}}\right),$$

and for any $k \in \mathbb{R}^d$

$$\|\varphi(k)\|_2^2 = \exp\left(\frac{\|k\|_2^2}{\sqrt{d}}\right).$$

Consider the set of vectors $\varphi(k_1) \otimes v_1, \ldots, \varphi(k_n) \otimes v_n$. Run the Self-Balancing Walk algorithm on the set of vectors $\varphi(k_1) \otimes v_1, \ldots, \varphi(k_n) \otimes v_n$ with failure parameter set to δ/s and denote by C' and $C \backslash C'$ the partition of C returned by the algorithm. Observe that, even though vectors $\varphi(k_i) \otimes v_i$ are infinite dimensional, Self-Balancing Walk still can be implemented. The algorithm never has to keep these vectors in the memory because the only operation which requires the knowledge of the embeddings – the inner product – can be performed if we just store vector pairs $\{k_i, v_i\}$:

$$\langle \varphi(k_i) \otimes v_i, \varphi(k_j) \otimes v_j \rangle = \exp\left(\frac{\langle k_i, k_j \rangle}{\sqrt{d}}\right) \cdot \langle v_i, v_j \rangle.$$

Denote by $e_1, \ldots, e_s$ the standard orthonormal basis in $\mathbb{R}^s$. By Theorem A.1, for any $i \in [s]$ with probability $1 - \delta/s$

$$\begin{aligned} &\left| \sum_{\{k,v\} \in C'} \langle \varphi(k) \otimes v, \varphi(q) \otimes e_i \rangle - \sum_{\{k,v\} \notin C'} \langle \varphi(k) \otimes v, \varphi(q) \otimes e_i \rangle \right| \\ &\qquad \leq O\left(\log(ns/\delta) \cdot \max_{\{k,v\} \in C} \|\varphi(k) \otimes v\|_2 \cdot \|\varphi(q) \otimes e_i\|_2 \right), \end{aligned} \tag{9}$$

and so with probability at least $1 - \delta$ all of the above inequalities hold simultaneously. To simplify the right hand side, notice that $\|\varphi(k) \otimes v\|_2 = \exp\left(\frac{\|k\|_2^2}{2\sqrt{d}}\right) \cdot \|v\|_2$ and $\|\varphi(q) \otimes e_i\|_2 = \exp\left(\frac{\|q\|_2^2}{2\sqrt{d}}\right)$. Observe that for any i $\langle \varphi(k) \otimes v, \varphi(q) \otimes e_i \rangle = \langle \varphi(k), \varphi(q) \rangle \cdot [v]_i = \exp\left(\frac{\langle k,q \rangle}{\sqrt{d}}\right) \cdot [v]_i$, where by $[v]_i$ we denote the i-th coordinate of the vector v. Therefore, the left hand side of the expression above is simply the absolute value of the i-th coordinate of the vector

$$\sum_{\{k,v\} \in C'} \exp\left(\frac{\langle k, q \rangle}{\sqrt{d}}\right) v - \sum_{\{k,v\} \notin C'} \exp\left(\frac{\langle k, q \rangle}{\sqrt{d}}\right) v.$$

Thus, Equation (9) provides a uniform upper bound on the absolute values of coordinates of the above vector. Since the l_∞ norm of a vector is the maximum of the absolute values of its coordinates,

$$\begin{aligned} &\left\| \sum_{\{k,v\} \in C'} \exp\left(\frac{\langle k, q \rangle}{\sqrt{d}}\right) v - \sum_{\{k,v\} \notin C'} \exp\left(\frac{\langle k, q \rangle}{\sqrt{d}}\right) v \right\|_\infty \\ &= \max_{i \in [s]} \left| \sum_{\{k,v\} \in C'} \langle \varphi(k) \otimes v, \varphi(q) \otimes e_i \rangle - \sum_{\{k,v\} \notin C'} \langle \varphi(k) \otimes v, \varphi(q) \otimes e_i \rangle \right| \\ &\leq O\left(\log(ns/\delta) \cdot \max_{\{k,v\} \in C} \left(\exp\left(\frac{\|k\|_2^2}{2\sqrt{d}}\right) \cdot \|v\|_2 \right) \cdot \exp\left(\frac{\|q\|_2^2}{2\sqrt{d}}\right) \right). \end{aligned}$$

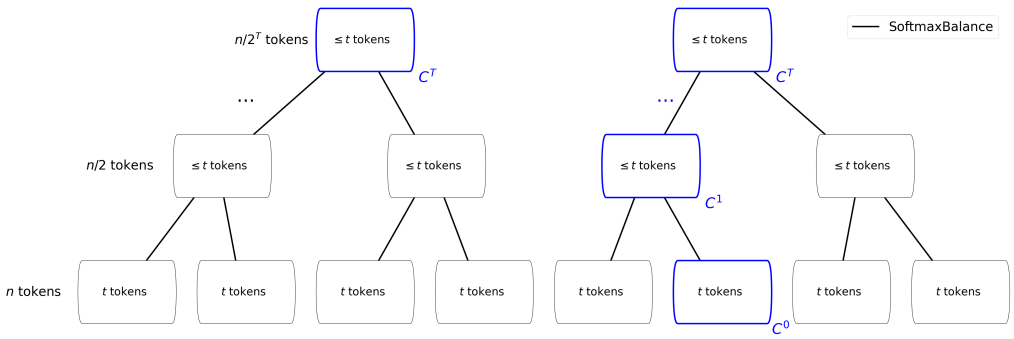

Figure 2: Illustration of the tree structure of MERGEANDREDUCE

Finally, we go from bounding the l_∞ norm a vector to bounding its l_2 norm:

$$\begin{aligned}
&\left\| \sum_{\{k,v\}\in C'} \exp\left(\frac{\langle k,q\rangle}{\sqrt{d}}\right) v - \sum_{\{k,v\}\notin C'} \exp\left(\frac{\langle k,q\rangle}{\sqrt{d}}\right) v \right\|_2 \\
&\leq \sqrt{s} \left\| \sum_{\{k,v\}\in C'} \exp\left(\frac{\langle k,q\rangle}{\sqrt{d}}\right) v - \sum_{\{k,v\}\notin C'} \exp\left(\frac{\langle k,q\rangle}{\sqrt{d}}\right) v \right\|_\infty \\
&\leq O\left(\sqrt{s}\cdot \log(ns/\delta) \cdot \max_{\{k,v\}\in C} \left(\exp\left(\frac{\|k\|_2^2}{2\sqrt{d}}\right)\cdot \|v\|_2\right) \cdot \exp\left(\frac{\|q\|_2^2}{2\sqrt{d}}\right)\right).
\end{aligned}$$

□

A.3 MERGEANDREDUCE

A.3.1 Pseudocode for MERGEANDREDUCE

The pseudocode for MERGEANDREDUCE is presented in 4.

Algorithm 4 MERGEANDREDUCE$((k_j, v_j)_j, t, T, \varepsilon)$

input: stream of $\leq n$ tokens (k_j, v_j), batch size t, compression rate 2^{-T}, precision parameter ε.
Let SOFTMAXBALANCE be the algorithm as per Theorem 3.3.
Initialize i-th level subset $C^i, i = 0, \ldots, T$, to empty
repeat
  $C^0 \leftarrow C^0 \cup \{\{k_j, v_j\}\}$
  if p is not a multiple of t then **then**
    output $C^0, \ldots, C^T$
    continue
  end if
  /*Update subsets every t steps*/
  $p \leftarrow j/t, i \leftarrow 0$
  while p is an integer and **until** $i = T$ **do**
    $C^{i+1} \leftarrow C^{i+1} \cup$ SOFTMAXBALANCE$(C^i, r_{\text{key}}, r_{\text{value}}, 1/\text{poly}(n))$
    $C^i \leftarrow \emptyset$
    $i \leftarrow i + 1$
    $p \leftarrow p/2$
  end while
  output $C^0, \ldots, C^T$
until token stream ends

A.3.2 Theoretical Guarantees of MERGEANDREDUCE

We now present the full proof of Theorem 3.4.

Theorem 3.4. *For any $r, \varepsilon > 0$, any set of tokens $(q_1, k_1, v_1), \ldots, (q_n, k_n, v_n)$ where $q_j, k_j \in \mathbb{R}^d$ satisfy $\|q_j\|_2, \|k_j\|_2 \leq r$, $v_j \in \mathbb{R}^s$ for $s \leq d$ suppose,*

$$\text{batch size } t = \widetilde{O}(\sqrt{s}e^{2r^2/\sqrt{d}}/\varepsilon) \text{ and compression rate } 2^{-T} \text{ with } T = \log(n/t).$$

Then MERGEANDREDUCE *on input parameters t, r, d, s, ε, outputs at every step j of the stream subsets of key-value embedding pairs $C^0, \ldots, C^T \subset C := \{(k_1, v_1), \ldots, (k_n, v_n)\}$ such that, $z_j := \sum_{i=0}^{T} 2^i \sum_{\{k,v\} \in C^i} \exp\left(\frac{\langle k, q_j \rangle}{\sqrt{d}}\right) v$, satisfies with probability at least $1 - 1/poly(n)$,*

$$\left\| \sum_{i=1}^{j} \exp\left(\frac{\langle k_i, q_j \rangle}{\sqrt{d}}\right) v_i - z_j \right\|_2 \leq \varepsilon j \cdot e^{-r^2/\sqrt{d}} \cdot \max_{i \in [n]} \|v_i\|_2.$$

Total memory of the algorithm is $\widetilde{O}(d\sqrt{s}e^{2r^2/\sqrt{d}}/\varepsilon)$, its j-th iteration runtime is $\widetilde{O}(dse^{4r^2/\sqrt{d}}/\varepsilon^2)$.

Proof. Let us first consider the performance of the procedure at time steps which are multiples of t. Note that since in the statement of the theorem $T = \log_2(n/t)$, condition **until** in line **while** is redundant. Observe that at any such j-th step the procedure is an online implementation of the following simple offline recursive algorithm on dataset $\{\{k_1, v_1\}, \ldots, \{k_j, v_j\}\}$:

1. Set $p = j/t$ and $i = 1$. Split the dataset $\{\{k_1, v_1\}, \ldots, \{k_j, v_j\}\}$ into batches $B_1^0, \ldots, B_p^0$ of size t.
2. While p is an integer:
   - Run SOFTMAXBALANCE on the batches $B_1^{i-1}, \ldots, B_p^{i-1}$ independently
   - If p is odd, store the output of SOFTMAXBALANCE on B_p^{i-1} in C^i
   - For every l, merge the outputs of SOFTMAXBALANCE on B_{2l-1}^{i-1} and B_{2l}^{i-1} into one batch and store them in B_l^i,
   - Update $p \leftarrow \lfloor p/2 \rfloor$, $i \leftarrow i + 1$. Stop when $p = 1$.

Therefore, we will analyze space complexity and performance guarantees of the above offline algorithm.

Probability of success.

Note that our algorithm performs correctly if each of the calls to SOFTMAXBALANCE produces small error on each of the queries q (as in theorem Theorem 3.3). Throughout the stream, we make $O(n/t)$ calls to SOFTMAXBALANCE, and we apply each to at most n queries, so, it is enough to require that that all SOFTMAXBALANCE have failure probability parameter $\delta = 1/\text{poly}(n)$.

Space complexity.

Observe that after each iteration of step 2 the number of batches decreases by a factor of two. The maximum batch size is always bounded by t. This is because a batch B_l at iteration i of step 2 is a union of SOFTMAXBALANCE(B_{2l-1}) and SOFTMAXBALANCE(B_{2l}) for B_{2l-1} and B_{2l} at iteration $i - 1$ of step 2, and SOFTMAXBALANCE reduced the size of the dataset which it has been applied to at least by a factor of 2.

The memory of the procedure is the collection of memory cells C^i, and $|C^i| \leq t$. Since at time step j at most $\log_2(p) \leq \log_2(n/t)$ memory cells are occupied, the total memory is bounded by $O(dt \log_2(n/t)) = O(dtT)$, which, using the $\widetilde{O}$ notation, is equal to $\widetilde{O}(d\sqrt{d}e^{2r^2/\sqrt{d}}/\varepsilon)$.

Performance of the algorithm.

Define $B^i = \cup_l B_l^i$ – the data points which remained in the batches after i iterations of step 2. By triangle inequality,

$$\left\| \sum_{i=1}^{T} 2^i \sum_{\{k,v\}\in C^i} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right) v - \sum_{i=1}^{j} \exp\left(\frac{\langle k_i, q_j\rangle}{\sqrt{d}}\right) v_i \right\|_2$$
$$\leq \sum_{i=0}^{T-1} \left\| 2^{i+1} \sum_{\{k,v\}\in B^{i+1}\cup C^{i+1}} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right) v - 2^i \sum_{\{k,v\}\in B^i} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right) v \right\|_2$$
$$\leq \sum_{i=0}^{T-1} 2^i \left\| \sum_{\{k,v\}\in B^{i+1}\cup C^{i+1}} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right) v - \sum_{\{k,v\}\in B^i\setminus(B^{i+1}\cup C^{i+1})} \exp\left(\frac{\langle k, q_j\rangle}{\sqrt{d}}\right) v \right\|_2.$$

We will refer to the i-th summand (starting from 0) on the right hand side as the error produced by the $i+1$-st iteration of step 2. At the $i+1$-st iteration of step 2 we apply SOFTMAXBALANCE to $p/2^i$ batches $B_1^i, B_2^i, \ldots$ of size t, and we save the outputs of SOFTMAXBALANCE in batches $C^{i+1}, B_1^{i+1}, B_2^{i+1}, \ldots$. Therefore, by Theorem 3.3 and triangle inequality, the error vector produced by the procedure at the $i+1$-st iteration of step 2 has l_2 norm bounded by

$$O\left(2^i \cdot \sqrt{s} \cdot \log(sn) \cdot \left(\frac{p}{2^i}\right) \cdot e^{r^2/\sqrt{d}} \max_{j\in[n]} \|v_j\|_2\right) = O\left(\sqrt{s} \cdot \log(sn) \cdot p \cdot e^{r^2/\sqrt{d}} \max_{j\in[n]} \|v_j\|_2\right),$$

since the error parameter δ of all instances of SOFTMAXBALANCE is set to $1/\text{poly}(n)$. The l_2 norm of the total error of our procedure is bounded by

$$O\left(\sqrt{s} \cdot \log(sn) \cdot T \cdot p \cdot e^{r^2/\sqrt{d}} \max_{j\in[n]} \|v_j\|_2\right).$$

By definition, $p = j/t$. In order to ensure that the statement of the theorem is correct, the upper bound on the l_2 norm of the error vector of the procedure should be less than the desired error $e^{-r^2/\sqrt{d}} \cdot \max_{i\in[n]} \|v_i\|_2$:

$$O\left(\sqrt{s} \cdot \log(sn) \cdot T \cdot \frac{j}{t} \cdot e^{r^2/\sqrt{d}} \max_{j\in[n]} \|v_j\|_2\right) \leq \varepsilon j \cdot e^{-r^2/\sqrt{d}} \cdot \max_{j\in[n]} \|v_j\|_2.$$

And, since by definition

$$t = O\left(\frac{\log^2(sn) \cdot \sqrt{s} \cdot e^{2r^2/\sqrt{d}}}{\varepsilon}\right),$$

the above inequality holds.

Runtime during one time step. At worst, during j-th time step the algorithm has to launch SOFTMAXBALANCE $\log_2(p) \leq \log_2(n/t) = T$ times on batches of size t, so the runtime is bounded by $O(dt^2T)$. In the $\widetilde{O}$ notation, the runtime is equal to $\widetilde{O}(d^2 e^{4r^2/\sqrt{d}}/\varepsilon^2)$.

As the final step, we will analyze the performance of the procedure at time steps j' which are not multiples of t. Define $j_t = \lfloor j'/t \rfloor \cdot t$. Note that at any such time step the procedure simply saves the triplet $(q_{j'}, k_{j'}, v_{j'})$ and outputs the sum of the approximation z_{j_t} such that

$$\left\| \sum_{i=1}^{j_t} \exp\left(\frac{\langle k_i, q'_j\rangle}{\sqrt{d}}\right) v_i - z_{j_t} \right\|_2 \leq \varepsilon j_t \cdot e^{-r^2/\sqrt{d}} \cdot \max_{i\in[n]} \|v_i\|_2,$$

and $\sum_{i=j_t+1}^{j'} \exp\left(\frac{\langle k_i, q'_j\rangle}{\sqrt{d}}\right) v_i$. From the above inequality,

$$\left\| \sum_{i=1}^{j'} \exp\left(\frac{\langle k_i, q'_j \rangle}{\sqrt{d}}\right) v_i - \left(z_{j_t} + \sum_{i=j_t+1}^{j'} \exp\left(\frac{\langle k_i, q'_j \rangle}{\sqrt{d}}\right) v_i \right) \right\|_2 \leq \varepsilon j_t \cdot e^{-r^2/\sqrt{d}} \cdot \max_{i \in [n]} \|v_i\|_2,$$

as desired. □

B Full Experimental Details

Experiments in Section 4.1 and Section 4.3 are performed on a single NVIDIA A100 GPU with 80GB VRAM, and the rest on a single NVIDIA RTX A6000 GPU with 48GB VRAM.

Implementation Detail. To enhance the practical performance of our algorithm, we implement BALANCEKV with parallel operations. Specifically, we consider the cache embeddings of length n and dimension d as a sequence of blocks with length b and reshape them into a tensor of shape $b \times (n/b) \times d$. Then, BALANCEKV is applied in parallel to all blocks of length b. For cases where n is not divisible by b, we pad the embeddings with zeros. After sign assignment to all embeddings in each block (i.e., line 9 in Algorithm 2), it is reshaped to its original length, and we strictly select $n/2$ embeddings, repeating this process for T iterations.

B.1 Ablation Studies on Single Layer Attention Approximation

In this section we re-state with full details the single layer attention approximation experiments presented in Section 4.1.

We empirically evaluate the performance of BALANCEKV for approximating a single attention layer, and compare it with independent uniform sampling. We use the pretrained `Llama-3.1-8B-Instruct` [27] and `Ministral-8B-Instruct-2410` [52] and TriviaQA dataset from LongBench [5], and consider the 1^{st},2^{nd} and 5^{th} layers of the models for attention approximation.

For given a prompt with length n, we store the corresponding query, key, and value embeddings for all layers. Denote a pair of embeddings in some layer by $(q_1, k_1, v_1), \ldots, (q_n, k_n, v_n)$ and the goal is to approximate the attention $\text{Attn}(q_j, K_j, V_j)$ for the latest 256 queries, i.e. $j \in [n - 256, n]$. Specifically, we keep several first and recent tokens separately and apply BALANCEKV to the intermediate row vectors in K_j. This is motivated by StreamingLLM [76] as important contexts are likely contained in the first and latest tokens. We retain the first 256 embeddings and the recent ones from $n - 256$ to j and our compressed cache contains tokens whose indices are in $[256] \cup S \cup \{n - 256, \ldots, j\}$ where $S \subseteq [257, n - 256]$ can be obtained from BALANCEKV. We explore four compression parameters $T \in \{1, 2, 3, 4\}$ which reduces the cache memory by a factor of 2^{-T}. Let z_j be our approximation using BALANCEKV plus the recent and first few embeddings at the stream $j \in [n - 256, n]$. We compute relative errors $\|z_j - \text{Attn}(q_j, K_j, V_j)\|_F / \|\text{Attn}(q_j, K_j, V_j)\|_F$ for all $j \in [n - 256, n]$, batches, heads and input prompts in the dataset. We repeat this with 10 different random seeds and compute their average and standard deviations. We also compare our method to independent uniform sampling, in which we replace the application of BALANCEKV with sampling a 2^{-T} fraction of key and value embeddings with indices in $[257, n - 256]$ uniformly at random. The results are reported in Figure 1.

Next we present the results of the ablation studies described in Section 4.1 which demonstrate how batch size and compression rate affect the relative error in attention approximation for layers 1 and 15 for `Llama-3.1-8B-Instruct`.

B.2 End-to-End Evaluation on LongBench

We now provide the complete experimental details on the end-to-end evaluation in Section 4.2. We benchmark our algorithm on LongBench dataset [5], a comprehensive collection of datasets designed to evaluate the long-context understanding capabilities of large language models. Specifically, we test a version of uniform length distribution (LongBench-E). The benchmark consists of various long-text application scenarios, including single-document question-answering, multi-document question-answering, summarization, few-shot learning, synthetic tasks and code completion. We use BALANCEKV to compress the key value cache generated in the prefill stage, and maintain all

Batch Size	1/2	1/4	1/8
256	0.0603	0.1190	0.1793
128	0.0320	0.0624	0.0922
64	0.0189	0.0349	0.0508

(a) Layer 1 Runtime (s)

	1/2	1/4	1/8
256	0.1036	0.1764	0.2655
128	0.1082	0.1833	0.2741
64	0.1137	0.1921	0.2858

(b) Layer 1 Relative Error

	1/2	1/4	1/8
256	0.3920	0.4505	0.5096
128	0.3654	0.3951	0.4256
64	0.3592	0.3753	0.3910

(c) Layer 15 Runtime (s)

	1/2	1/4	1/8
256	0.1107	0.1935	0.2798
128	0.1121	0.1952	0.2813
64	0.1141	0.1978	0.2845

(d) Layer 15 Relative Error

Figure 3: Runtime and relative error for across different layers and block sizes. In each figure the rows are corresponding to various batch sizes and columns corresponding to various compression rates

streamed embeddings (q_j, k_j, v_j) during the token decoding/generation stage. This is because the number of generated tokens is much smaller than the input sequence length. We set $b = 256$ and $T = 2$, achieving a consistent compression rate of 0.25 across all inputs.

We evaluate our method against several token-level key value cache compression schemes, including StreamingLLM [76], SnapKV [46], and PyramidKV [14] as well as uniform sampling described in Section 4.1. We use their implementations from MInference [37], and configure their hyperparameters to match a uniform compression rate with 0.25. We follow the same evaluation metrics from [5]. We test them on `Llama-3.1-8B-Instruct` as well as bigger 14B and 32B parameter models `Qwen-2.5-14B-Instruct` and `Qwen-2.5-32B-Instruct` [77, 71], with results summarized in Table 1.

Our method consistently achieves the highest average performance among compression methods and across all models, demonstrating its effectiveness in preserving model quality for the cache compression. Notably, on the triviaqa dataset, it achieves near-exact scores compared to uncompressed baselines (e.g., 80.68 vs. 81.14 with Qwen2.5-32B), highlighting its ability to retain high-quality information. We observe that uniform sampling performs competitively with our method and this result justifies that a subset obtained from discrepancy theory has practical impacts on various LLM tasks.

B.3 Needle-In-A-Haystack

In this section we report the plots corresponding to the Needle in a Haystack experiment described in Section 4.3. They are presented in Figure 4.

B.4 System Efficiency Metrics

In this section we present the prefill and decoding time numbers in Table 2 as described in the system efficiency experimental details in Section 4.4.

Method	Prefill Time (sec)	Decoding Time (sec)
Exact	3.032	37.769
SnapKV	3.755	40.426
PyramidKV	3.748	37.241
StreamingLLM	3.681	40.276
BALANCEKV	3.662	38.054

Table 2: Minimum wall-clock runtime (in seconds) over 10 trials for prefill and decoding stages.

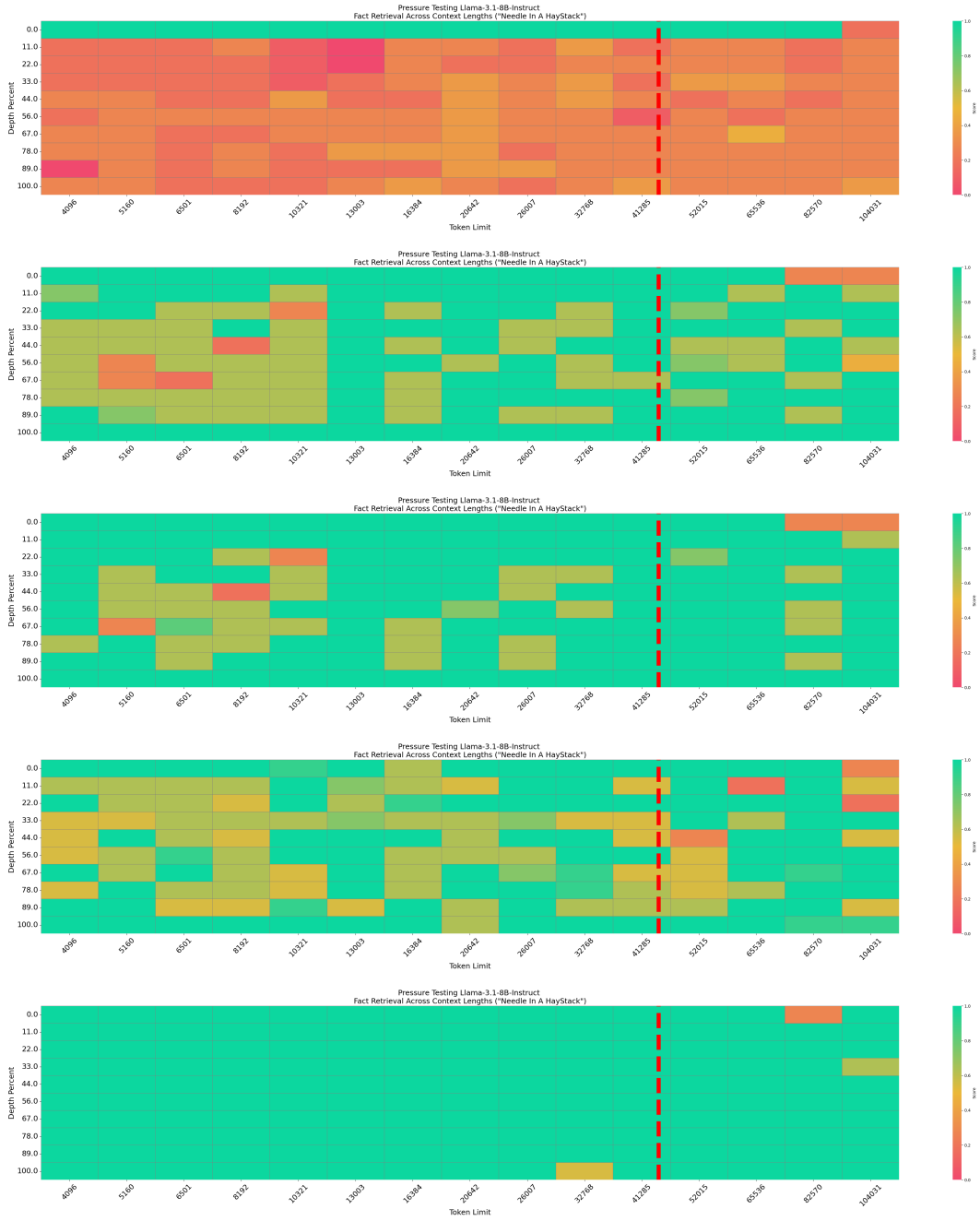

Figure 4: Comparison of performance on Needle in a Haystack task using `Llama-3.1-8B-Instruct`. The methods corresponding to figures from top to bottom are StreamingLLM, SnapKV, PyramidKV, Unif. Sampling and BALANCEKV respectively.

B.5 Additional Experiments

1. In Table 3, we report the results of the experiment analyzing the ℓ_2 norms of query (Q), key (K) and value (V) embeddings described in Section 4.5. Due to the space limit, we provide representative results in the below table from (randomly chosen) prompts of various sequence lengths. We note that the reported ℓ_2 norms of keys shifted by their average, as opposed to the norms of the keys, because our implementation of BALANCEKV shifts the keys by their

average before the compression. It is also easy to see that attention is invariant to the operation of shifting the keys by their average, so it is non-restrictive to assume that the average of the key values is zero.

Table 3: Statistics of norms of qkv embeddings for randomly chosen prompts from TriviaQA.

Prompt ID	Seq Len	Query		Key Shifted		Value	
		Mean	95% CI	Mean	95% CI	Mean	95% CI
10	2281	14.4512	0.0029	15.3451	0.0065	3.3398	0.0043
24	3131	14.6003	0.0025	15.6603	0.0058	3.3539	0.0036
30	3388	14.7406	0.0024	15.5152	0.0054	3.3419	0.0035
22	4230	14.8339	0.0021	15.6907	0.0049	3.3468	0.0032
5	5734	14.9259	0.0019	15.7149	0.0041	3.3482	0.0027
14	6616	14.9000	0.0017	15.6757	0.0038	3.3596	0.0025
4	6962	14.9743	0.0017	15.7346	0.0039	3.3450	0.0024
21	8041	14.9364	0.0016	15.7025	0.0035	3.3491	0.0023
26	17337	15.1437	0.0011	15.9531	0.0024	3.3654	0.0016
27	21274	15.2065	0.0010	15.8745	0.0022	3.3683	0.0014

2. In Table 4, we report the results of the multimodal task experiment described in Section 4.5.

Table 4: Comparison of BALANCEKV to uniform sampling on MS-COCO, using `InternVL2.5-8B`. The bracket for every method contains the compression rate.

Method	Bleu_1	Bleu_2	Bleu_3	Bleu_4	METEOR	RougeL	CIDEr
Exact	0.795	0.629	0.476	0.351	0.291	0.580	1.255
BalanceKV (1/4)	0.794	0.628	0.475	0.351	0.290	0.579	1.251
Unif (1/4)	0.794	0.629	0.476	0.350	0.290	0.578	1.247
BalanceKV (1/16)	0.789	0.622	0.468	0.343	0.286	0.573	1.221
Unif (1/16)	0.789	0.619	0.465	0.340	0.284	0.571	1.207

3. In Table 5, we present the results of the end-to-end experiment on LongBench in the extremely low error regime, described in Section 4.5. We note, that etremely low error regime corresponds to the low compression rate regime, and therefore our experiment is equivalent to exploring the performance of BALANCEKV in the low compression rate regime. For compression rates of 0.8, 0.9 and 0.95, we randomly select a dataset from LongBench and apply both uniform sampling and BALANCEKV to achieve the desired compression rate. More specifically, if we wish to compress a KV cache to $1-\alpha$ of it's original size, we select its subset of size 2α, compress it by a factor of 2 with either uniform sampling or BALANCEKV and keep the rest exactly.

Table 5: Comparison of BALANCEKV to uniform sampling in LongBench in the extremely low-error regime.

Compression Rate	Dataset	BalanceKV	Uniform	Baseline
0.8	HotpotQA	50.2	48.4	51.9
0.8	TriviaQA	91.6	86.3	91.6
0.9	MultiFieldQA	47.5	44.9	47.8
0.9	Qasper	42.3	39.6	43.1
0.95	LCC	49.3	45.7	49.5
0.95	P.Count	20.7	20.1	20.7

4. In Table 6, we present the results of the end-to-end evaluation of ClusterGen [83] as per the experimental setup in Appendix B.2 and compare its performance to both BALANCEKV and exact attention.

Method	qasper	multi	hotpotqa	2wiki	gov	multinews	trec	triviaqa	samsum	p.count	p.ret	lcc	repo-p	average
Llama-3.1-8B-Instruct														
Exact (Baseline)	42.87	48.54	52.05	38.6	31.31	22.07	71.67	91.85	42.36	20.37	98.13	49.62	42.73	50.17
ClusterGen	33.93	42.31	50.85	37.24	21.16	19.31	67.67	90.82	39.49	20.20	96.57	47.23	39.26	46.62
BALANCEKV	35.75	37.04	46.37	36.24	27.09	20.84	69.0	90.88	37.88	20.39	96.65	48.45	41.4	**46.77**

Table 6: Comparison of ClusterGen, BalanceKV and exact attention on LongBench-E using `Llama-3.1-8B-Instruct`. The best results among compression methods for each model are highlighted in bold.

C Lower Bound

In this section, we prove the lower bound on the space complexity of an algorithm approximating the $\mathrm{Attn}(\cdot, K, V)$ function. More formally,

Theorem 3.2. *Suppose that $r^2 \leq d$. Any streaming algorithm which on input $(\{k_1, v_1\}, \ldots, \{k_n, v_n\}, q)$, $\|q\|_2, \|k_i\|_2 \leq r$, outputs z_q satisfying Equation (2) with probability 0.999 has space complexity $\Omega\left(\min\{\frac{1}{\varepsilon^2}, d\exp(2r^2/\sqrt{d})\}\right)$.*

The proof will be a reduction to the well-known INDEX problem.

C.1 Reduction to the INDEX Problem

Definition C.1 (The INDEX problem)**.** Alice gets a bit string $x \sim \mathrm{Unif}\{0,1\}^n$ and Bob gets $i \sim \mathrm{Unif}[n]$. Then, the goal is to compute $f(x,i) = x_i$ on Bob's end with a single message m from Alice. Denote by $R_\delta^{pub,\rightarrow}$ the public coin one-way communication complexity of computing a function $f(x,y)$ with error probability at most δ: Alice holds x, Bob holds y, they share a source of random bits and Alice sends a single message to Bob, after which he must output the correct answer with probability at least $1-\delta$.

Theorem C.2 (Proven in [75])**.**

$$R_{2/3}^{pub,\rightarrow}(INDEX) \geq \Omega(n).$$

Proof of Theorem 3.2. Let c be the small constant such that $R_{2/3}^{pub,\rightarrow}(INDEX) \geq c \cdot n$.

Assume the contrary to the statement of the Theorem 3.2 – that there exists a streaming algorithm of space complexity $const \cdot \min\{\frac{1}{\varepsilon^2}, d\exp(2r^2/\sqrt{d})\}$ for any sufficiently small constant $const$. We will show that given a string of length $\min\{\frac{1}{\varepsilon^2}, d\exp(2r^2/\sqrt{d})\}$ Alice can solve the INDEX problem as follows. She instantiates such an algorithm with $const < c/C$ for a sufficiently large constant C, gives it as input a carefully selected set of keys and values, and sends the state of its memory to Bob. Bob, on his end, can determine whether any randomly drawn bit $i \sim \mathrm{Unif}[n]$ equals 0 or 1 with probability 0.8 by issuing a corresponding (carefully crafted) query to the streaming algorithm and observing its output. Thus, if the streaming algorithm uses small space, we get a contradiction with Theorem C.2, and therefore obtain a proof of Theorem 3.2.

The reduction. Suppose Alice's input to the INDEX problem is a bit string $x \in \{0,1\}^n$ of length $n = \min\{\frac{1}{\varepsilon^2}, d\exp(2r^2/\sqrt{d})\}$. Using public coins, Alice and Bob jointly generate n/d key vectors $\tilde{k}_1, \ldots, \tilde{k}_{n/d} \sim \mathrm{Unif}\left\{-\frac{r}{\sqrt{d}}, \frac{r}{\sqrt{d}}\right\}^d$, function $\pi : [n] \to [n/d] \times [d]$ which randomly partitions the n bits into groups of size d, and n random signs $\sigma_1, \ldots, \sigma_n \sim \mathrm{Unif}\{-1,1\}$.

Let $\pi(i)_1 \in [n/d]$ be the first component of $\pi(i)$ and $\pi(i)_2 \in [d]$ – the second component of $\pi(i)$. Let $e_1, \ldots, e_d$ be the standard orthonormal basis in $\mathbb{R}^d$. We build the dataset of n key-value pairs in the following way. We associate with the i-th bit the key vector $k_i := \tilde{k}_{\pi(i)_1}$ and the value vector $v_i := \sigma_i \cdot e_{\pi(i)_2}$.

Define

$$U = \{\{k_i, v_i\} : x_i = 1\}$$

the set of key-value pairs corresponding to entries 1 in Alice's bit string x. Alice instantiates the streaming algorithm for approximating $\text{Attn}(\cdot, K, V)$ with space complexity $\frac{c}{C} \cdot \min\{\frac{1}{\varepsilon^2}, d\exp(2r^2/\sqrt{d})\}$ with a big enough constant C which we specify later and sends the state of it's memory before reading q to Bob. When Bob receives the message, he uses it to approximate $\text{Attn}(q_i, K, V)$ where $q_i = k_i$. If the value written in the only non-zero coordinate of v_i is larger than $\frac{1}{40} \cdot \frac{\exp(r^2/\sqrt{d})}{\max\{d\exp(r^2/\sqrt{d}), |U|\}}$, Bob reports that the i-th bit of Alice's string is equal to 1, and otherwise - 0.

Analysis of the reduction.

Proof sketch. Before moving to formal proofs we briefly outline the main idea of the analysis. Observe that, by the choice of key vectors, any $\exp(\langle k_j, q_i\rangle/\sqrt{d})$ for $j \neq i$ is in expectation insignificantly small compared to $\exp(\langle k_i, q_i\rangle/\sqrt{d})$ – sometimes we will even refer to these terms as "noise". This statement is formalized in Lemma C.4. Therefore, when Bob computes an approximation to $\text{Attn}(q_i, K, V)$, he will observe a *large* value in the coordinate where v_i is non-zero if $\{k_i, v_i\} \in U$ and a *small* value otherwise.

Lemma C.3. $\mathbb{E}_{x,y\sim Unif\left\{-\frac{r}{\sqrt{d}}, \frac{r}{\sqrt{d}}\right\}^d}[\exp(C\langle x, y\rangle/\sqrt{d})] = \Theta(1)$ *for any constant* C.

Proof.

$$\mathbb{E}_{x,y\sim \text{Unif}\left\{-\frac{r}{\sqrt{d}}, \frac{r}{\sqrt{d}}\right\}^d}[\exp(C\langle x, y\rangle/\sqrt{d})]$$
$$= \left(\frac{1}{2}\exp(Cr^2/d^{3/2}) + \frac{1}{2}\exp(-Cr^2/d^{3/2}))\right)^d = \cosh\left(\frac{Cr^2}{d^{3/2}}\right)^d,$$
$$1 \leq \exp(C^2r^4/4d^2) \leq \cosh\left(\frac{Cr^2}{d^{3/2}}\right)^d \leq \exp(C^2r^4/d^2) \leq \exp(C^2)$$

where we used the assumption that $r^2/d \leq 1$. □

Lemma C.4. *Fix $i \in [n]$, select $q_i = k_i$. Let $|U_i|$ be the number of key-value pairs in $U \setminus \{k_i, v_i\}$ whose value vector has non-zero entry in the same coordinate as v_i.*

If $\{k_i, v_i\} \in U$ then with probability $> 1 - \frac{1}{1000} \cdot \frac{|U_i|}{|U|}$

$$\left|\sum_{\{k,v\}\in U} \exp\left(\frac{\langle k, q_i\rangle}{\sqrt{d}}\right)\langle v, v_i\rangle\right| \geq \exp(r^2/\sqrt{d}) - O\left(\sqrt{|U|}\right).$$

Otherwise, with probability $> 1 - \frac{1}{1000} \cdot \frac{|U_i|}{|U|}$

$$\left|\sum_{\{k,v\}\in U} \exp\left(\frac{\langle k, q_i\rangle}{\sqrt{d}}\right)\langle v, v_i\rangle\right| \leq O\left(\sqrt{|U|}\right).$$

Proof. We prove both statements using Chebyshev's inequality.

In the first case, i.e. when $\{k_i, v_i\} \in U$, the sum contains the term $\exp(\langle k_i, q_i\rangle/\sqrt{d}) = \exp(r^2/\sqrt{d})$, and otherwise it does not. It therefore remains to upper bound the absolute value of the sum

$$X = \sum_{\substack{\{k,v\}\in U,\\ \{k,v\}\neq\{k_i,v_i\}}} \sigma_k \exp\left(\frac{\langle k, q_i\rangle}{\sqrt{d}}\right)\langle v, v_i\rangle = \sum_{\substack{\{k,v\}\in U_i,\\ \{k,v\}\neq\{k_i,v_i\}}} \sigma_k \exp\left(\frac{\langle k, q_i\rangle}{\sqrt{d}}\right)\langle v, v_i\rangle,$$

$\sigma_k \sim \text{Unif}\{-1, 1\}$, which effectively introduces "noise" in Bob's estimate of whether $x_i = 1$. We upper bound this sum now.

$\mathbb{E}[X] = 0$, and $Var(X) = |U_i| \cdot Var_{x,y\sim \text{Unif}\left\{-\frac{r}{\sqrt{d}},\frac{r}{\sqrt{d}}\right\}^d}(\exp(\langle x,y\rangle/\sqrt{d})) \leq O(|U_i|)$ because

$$Var_{x,y\sim \text{Unif}\left\{-\frac{r}{\sqrt{d}},\frac{r}{\sqrt{d}}\right\}^d}(\exp(\langle x,y\rangle/\sqrt{d})) \leq \mathbb{E}_{x,y\sim \text{Unif}\left\{-\frac{r}{\sqrt{d}},\frac{r}{\sqrt{d}}\right\}^d}[\exp(2\langle x,y\rangle/\sqrt{d})] \leq \exp(4),$$

by Lemma C.3. We therefore get by Chebyshev's inequality

$$\Pr\left[|X| \geq 1000\sqrt{|U|}\right] \leq \frac{1}{1000} \cdot \frac{|U_i|}{|U|}.$$

Therefore, with probability $1 - \frac{1}{1000} \cdot \frac{|U_i|}{|U|}$

$$\left|\sum_{\{k,v\}\in U} \exp\left(\frac{\langle k,q_i\rangle}{\sqrt{d}}\right)\langle v,v_i\rangle\right| \geq \exp(r^2/\sqrt{d}) - 1000\sqrt{|U|}.$$

In the second case, the entire sum equals $X = \sum_{k\in U_i} \sigma_k \exp(\langle k,q_i\rangle/\sqrt{d})$. As shown above, $\Pr\left[|X| \geq 1000\sqrt{|U|}\right] \leq \frac{1}{1000} \cdot \frac{|U_i|}{|U|}$. Hence, with probability $1 - \frac{1}{1000} \cdot \frac{|U_i|}{|U|}$

$$\left|\sum_{\{k,v\}\in U} \exp\left(\frac{\langle k,q_i\rangle}{\sqrt{d}}\right)\langle v,v_i\rangle\right| \leq 1000\sqrt{|U|}.$$

□

Corollary C.5. *Suppose bits $i_1,\ldots,i_d$ form a group – that is, $\pi(i_1)_1 = \pi(i_2)_1 = \ldots = \pi(i_d)_1$. Then all $v_{i_1},\ldots,v_{i_d}$ have different non-zero coordinates, and therefore $\sum_{j=1}^{d} |U_{i_j}| \leq |U|$.*

Therefore, by the union bound argument, the conclusion of Lemma C.4 holds for all d bits which form one group simultaneously with probability 0.999.

Lemma C.6. *Fix a bit i. With probability 0.98 the following holds:*

1. *The error of the approximating algorithm in the only non-zero coordinate of v_i is bounded by*

$$O\left(\frac{\varepsilon}{\sqrt{d}} \cdot \|\textit{softmax}(K\cdot q)\|_2 \cdot \|V\|_F\right).$$

2. *If the i-th bit is 1 then*

$$\left|\sum_{\{k,v\}\in U} \exp\left(\frac{\langle k,q_i\rangle}{\sqrt{d}}\right)\langle v,v_i\rangle\right| \geq \exp(r^2/\sqrt{d}) - O\left(\frac{\sqrt{|U|}}{\sqrt{d}}\right),$$

   and

$$\left|\sum_{\{k,v\}\in U} \exp\left(\frac{\langle k,q_i\rangle}{\sqrt{d}}\right)\langle v,v_i\rangle\right| \leq O\left(\frac{\sqrt{|U|}}{\sqrt{d}}\right).$$

   otherwise.

Proof. We may think that the process of generating the dataset and the approximating streaming algorithm has the following order: first Alice and Bob jointly generate the partition π, the key vectors $\tilde{k}_1,\ldots,\tilde{k}_{n/d}$ and the value vectors $v_1,\ldots,v_n$ all using public randomness. To generate Bob's input position $i \in [n]$ we generate pair $a \sim \text{Unif}[n/d]$, $b \sim \text{Unif}[d]$ and declare $i = \pi^{-1}(a,b)$. We may assume that a is chosen before the datasets K and V are generated, and b – after.

Before b is drawn, the key vector k_i of $i = \pi^{-1}(a, b)$ is already defined, as well as the datasets K, V and U. Alice can therefore already apply the streaming algorithm to U, and Bob can already apply it to $q_i = k_i$. Therefore, the error vector which the streaming algorithm yields when applied to $q_i = k_i$ is also defined before b is known.

Clearly, there are no more than $0.0001 \cdot d$ coordinates in which the error of approximation exceeds $10000 \cdot \frac{\varepsilon}{\sqrt{d}} \cdot \|\text{softmax}(K \cdot q)\|_2 \cdot \|V\|_F$. Since every value vector has only one non-zero entry, there are no more than $0.0001 \cdot d$ coordinates where at least $10000 \cdot \frac{U}{d}$ of value vectors from U have non-zero value. We call all coordinates which are in neither of these two groups *safe*. From the above, at least 99% of the coordinates are safe. Recall that $b \sim \text{Unif}[d]$, and choosing b is equivalent to choosing the coordinate in which v_i is non-zero. Therefore, with probability 0.99 over the choice of b the only non-zero coordinate of v_i is safe.

At the same time, similarly to Lemma C.4, by Chebyshev inequality, if $U_i \subset U$ is the set of all key-value pairs in U whose value vector has the same non-zero coordinate as v_i then with probability 0.999 if the i-th bit is 1 then

$$\left| \sum_{\{k,v\} \in U} \exp\left(\frac{\langle k, q_i \rangle}{\sqrt{d}} \right) \langle v, v_i \rangle \right| \geq \exp(r^2/\sqrt{d}) - O\left(\sqrt{|U_i|} \right),$$

and

$$\left| \sum_{\{k,v\} \in U} \exp\left(\frac{\langle k, q_i \rangle}{\sqrt{d}} \right) \langle v, v_i \rangle \right| \leq O\left(\sqrt{|U_i|} \right).$$

otherwise.

By union bounding over these two events, we get that the statement of the lemma is correct with high constant probability.

□

Conclusion of the proof. Let $U^i \subset U$ be the set of all pairs from U with the same key as $\{k_i, v_i\}$. Since Alice's string is drawn from $\text{Unif}\{-1, 1\}^n$, with probability 0.9 $|U^i| \geq 0.4 \cdot d$. This is because every bit in the same group as k_i belongs to U with probability 1/2.

Observe that by Chebyshev inequality, with high probability 0.999, the denominator of $\text{softmax}(K \cdot q_i)$ lies in range

$$\left[|U^i| \cdot \exp(r^2/\sqrt{d}) + \frac{1}{5} \cdot |U|, |U^i| \cdot \exp(r^2/\sqrt{d}) + 20 \cdot |U| \right],$$

This is because every summand in the denominator, except for $\exp(\langle k_i, q_i \rangle / \sqrt{d})$, is distributed as $\exp(\langle x, y \rangle / \sqrt{d})$, $x, y \sim \text{Unif}\left\{ -\frac{r}{\sqrt{d}}, \frac{r}{\sqrt{d}} \right\}^d$, and the expectation and the variance of this distribution, as shown in Lemma C.3, is $\Theta(1)$. This range is contained in $\left[\frac{1}{5} \cdot (\max\{de^{r^2/\sqrt{d}}, |U|\}), 20 \cdot (\max\{de^{r^2/\sqrt{d}}, |U|\}) \right]$. We will denote the denominator as D.

Similarly, by Chebyshev inequality, with probability 0.999 the numerator of $\text{softmax}(K \cdot q_i)$ lies in

$$\left[\sqrt{|U^i| \cdot \exp(2r^2/\sqrt{d}) + \frac{1}{5} \cdot |U|}, \sqrt{|U^i| \cdot \exp(2r^2/\sqrt{d}) + 200 \cdot |U|} \right]$$

which, since $|U| \leq d \exp(2r^2/\sqrt{d})$, is bounded by $\sqrt{200} \cdot \sqrt{d} e^{r^2/\sqrt{d}}$.

Suppose that the i-th bit is 1. Then,

- When $\frac{1}{\varepsilon} \geq \sqrt{d}e^{r^2/\sqrt{d}}$, by selecting $|U| = \frac{c}{C} \cdot de^{2r^2/\sqrt{d}}$ for some enough constant C the value written in the only non-zero coordinate of v_i is at least $\frac{e^{r^2/\sqrt{d}}}{D} - \frac{1}{100}\frac{e^{r^2/\sqrt{d}}}{D}$ and at most $\frac{1}{100}\frac{e^{r^2/\sqrt{d}}}{D}$ otherwise, as follows from Lemma C.4;
- When $\frac{1}{\varepsilon} < \sqrt{d}e^{r^2/\sqrt{d}}$, by selecting $|U| = \frac{c}{C} \cdot \frac{1}{\varepsilon^2}$ for some big enough constant C the value written in the only non-zero coordinate of v_i is at least $\frac{e^{r^2/\sqrt{d}}}{D} - \frac{1}{100} \cdot \frac{1}{\varepsilon \cdot D \cdot \sqrt{d}}$, and at most $\frac{1}{100} \cdot \frac{1}{\varepsilon \cdot D \cdot \sqrt{d}}$ otherwise, as follows from Lemma C.4.

The error which the approximator can have in the non-zero coordinate of v_i is bounded by

$$10000\frac{\varepsilon}{\sqrt{d}} \cdot \|\text{softmax}(K \cdot q)\|_2 \cdot \|V\|_F \leq 10000\frac{\varepsilon}{\sqrt{d}} \cdot \frac{\sqrt{20} \cdot \sqrt{d}e^{r^2/\sqrt{d}}}{D} \cdot \sqrt{|U|},$$

as shown in Lemma C.6. Below, we show that this error is smaller than the gap between $\frac{1}{40} \cdot \frac{e^{r^2/\sqrt{d}}}{\max\{de^{r^2/\sqrt{d}}, |U|\}}$ and the value written in the coordinate, which means that, even though the approximator introduces some error, Bob is still capable to tell whether the i-th bit is 1 or 0.

- When $\frac{1}{\varepsilon} \geq \sqrt{d}e^{r^2/\sqrt{d}}$, the gap between the value written in the coordinate and $\frac{1}{40}\frac{e^{r^2/\sqrt{d}}}{\max\{de^{r^2/\sqrt{d}}, |U|\}}$ is at least $\frac{1}{1000} \cdot \frac{e^{r^2/\sqrt{d}}}{\max\{de^{r^2/\sqrt{d}}, |U|\}}$, and the error

  $$\frac{10000\varepsilon}{\sqrt{d}} \cdot \frac{\sqrt{20} \cdot \sqrt{d}e^{r^2/\sqrt{d}}}{D} \cdot \sqrt{|U|} \leq \frac{\varepsilon}{2000} \cdot \frac{\sqrt{d}\exp(2r^2/\sqrt{d})}{\max\{de^{r^2/\sqrt{d}}, |U|\}} \leq \frac{1}{2000} \cdot \frac{e^{r^2/\sqrt{d}}}{\max\{de^{r^2/\sqrt{d}}, |U|\}}$$

  by an appropriate choice of C.
- When $\frac{1}{\varepsilon} < \sqrt{d}e^{r^2/\sqrt{d}}$, the gap between the value written in the coordinate and $\frac{1}{40}\frac{e^{r^2/\sqrt{d}}}{\max\{de^{r^2/\sqrt{d}}, |U|\}}$ is at least $\frac{1}{1000} \cdot \frac{e^{r^2/\sqrt{d}}}{\max\{de^{r^2/\sqrt{d}}, |U|\}}$ and the error

  $$\frac{10000\varepsilon}{\sqrt{d}} \cdot \frac{\sqrt{20} \cdot \sqrt{d}e^{r^2/\sqrt{d}}}{D} \cdot \sqrt{|U|} \leq \frac{\varepsilon}{2000} \cdot \frac{\exp(r^2/\sqrt{d})}{\max\{de^{r^2/\sqrt{d}}, |U|\} \cdot \varepsilon} \leq \frac{1}{2000} \cdot \frac{\exp(r^2/\sqrt{d})}{\max\{de^{r^2/\sqrt{d}}, |U|\}}$$

  by an appropriate choice of C.

□