arXiv:2504.19874v1 [cs.LG] 28 Apr 2025

TurboQuant: Online Vector Quantization with Near-optimal Distortion Rate

Amir Zandieh
Google Research
zandieh@google.com

Majid Daliri
New York University
daliri.majid@nyu.edu

Majid Hadian
Google DeepMind
majidh@google.com

Vahab Mirrokni
Google Research
mirrokni@google.com

Abstract

Vector quantization, a problem rooted in Shannon's source coding theory, aims to quantize high-dimensional Euclidean vectors while minimizing distortion in their geometric structure. We propose TurboQuant to address both mean-squared error (MSE) and inner product distortion, overcoming limitations of existing methods that fail to achieve optimal distortion rates. Our data-oblivious algorithms, suitable for online applications, achieve near-optimal distortion rates (within a small constant factor) across all bit-widths and dimensions. TurboQuant achieves this by randomly rotating input vectors, inducing a concentrated Beta distribution on coordinates, and leveraging the near-independence property of distinct coordinates in high dimensions to simply apply optimal scalar quantizers per each coordinate. Recognizing that MSE-optimal quantizers introduce bias in inner product estimation, we propose a two-stage approach: applying an MSE quantizer followed by a 1-bit Quantized JL (QJL) transform on the residual, resulting in an unbiased inner product quantizer. We also provide a formal proof of the information-theoretic lower bounds on best achievable distortion rate by any vector quantizer, demonstrating that TurboQuant closely matches these bounds, differing only by a small constant (≈ 2.7) factor. Experimental results validate our theoretical findings, showing that for KV cache quantization, we achieve absolute quality neutrality with 3.5 bits per channel and marginal quality degradation with 2.5 bits per channel. Furthermore, in nearest neighbor search tasks, our method outperforms existing product quantization techniques in recall while reducing indexing time to virtually zero.

1 Introduction

Vector quantization (VQ) in Euclidean space is crucial for efficiently handling high-dimensional vectors across a spectrum of computational domains, from training and deploying large-scale AI and deep learning models to powering vector databases for search/retrieval systems. The core objective is to compress high dimensional vectors by quantizing them–converting floating-point coordinate values to low-bitwidth integers–while minimizing distortion, quantified by metrics such as

mean-squared error (MSE) or inner product errors. By preserving these properties, inner product queries can be answered rapidly, with minimal latency, and using reduced computational and communication resources.

This problem's roots trace back to Shannon's seminal work on Source Coding theory [48, 49], which established that the least distortion achievable by block source codes, now known as vector quantizers, is defined by the Shannon distortion-rate function, determined by the statistical properties of the source and the chosen distortion measure, such as MSE. Today, VQ plays a critical role in fundamental computational domains, including AI, deep learning, and search systems.

A key application of VQ is in the deployment of AI models, including large language models (LLMs) [5, 18, 7, 52]. As LLM capabilities depend heavily on their model size and context length [34], serving them requires substantial memory demands and increased inference latency. This latency is primarily attributed to communication bottlenecks between HBM and SRAM on accelerators, or across distributed clusters. By compressing or quantizing model weights and activations, we can effectively mitigate these bottlenecks, resulting in significant reductions in inference costs. Inner product operations between activations and weights is at the core of deep learning models. Thus, model quantization schemes strive to compress weights and/or activation vectors while accurately preserving these inner products.

Decoder based transformer models [54] present another compelling use case. These models must store key/value (KV) embeddings from previously generated tokens in the KV cache, the size of which scales with both model size (number of layers and attention heads) and context length. This scaling is a significant bottleneck in terms of memory usage and computational speed, especially for long context models. Therefore, reducing the KV cache size without compromising accuracy is essential. In this context, the preservation of the Euclidean structure of these embedding vectors–their inner products and distances–is crucial for maintaining model performance. VQ emerges as the most suitable framework for addressing this challenge, offering a robust approach to compressing high-dimensional embeddings while preserving their essential geometric properties.

Additionally, nearest neighbor (NN) search in high-dimensional spaces with inner product or cosine similarity [1, 27] is a cornerstone of vector databases [4, 2, 3]. These databases are fundamental for retrieval-augmented generation [23, 19] and information retrieval [35, 46]. VQ, a.k.a. product quantization (PQ), plays a critical role in these applications. It enables efficient compression of database vectors, optimizes memory usage, and facilitates low-latency, accurate estimations of inner products with query vectors, thereby enabling fast and precise nearest neighbor searches.

Existing VQ algorithms present a trade-off: either they lack accelerator (vectorization) compatibility and exhibit slow computation, making them unsuitable for real-time AI applications like KV cache quantization, or they suffer from suboptimal distortion bounds relative to bit-width. Our objective is to introduce an algorithm that addresses these limitations. Specifically, we design TurboQuant: a lightweight, capable of online application (crucial for scenarios like KV cache quantization), and highly accelerator-friendly—a critical attribute for modern AI workloads.

The core of TurboQuant is a two-stage process. First, we develop a vector quantizer with optimal distortion rate in terms of mean-squared error (MSE). Subsequently, we apply a 1-bit quantizer to the residual, resulting in an unbiased and low-distortion inner product quantizer. We demonstrate that quantizers optimized for MSE do not produce unbiased estimators for inner products, and

our two-stage solution effectively bridges this gap. Our MSE-optimal quantizer starts by randomly rotating d-dimensional input vectors. Observing the key fact that each coordinate in the rotated vectors follows a Beta distribution, we design optimal Lloyd-Max quantizer [42, 43] for each coordinate by solving a continuous k-means problem. This method gives optimal MSE distortion bound and minimizes the L2 norm of the residual. To obtain an unbiased and low-distortion quantizer for inner products, we compose our quantizer with the recently developed Quantized Johnson-Lindenstrauss (QJL) transform [62], which quantizes each coordinate of the residual vector to a single bit. Our algorithm offers provably optimal distortion bounds for both MSE and inner products, achieving an exponential improvement over existing methods in terms of bit-width dependence.

1.1 Problem Definition

Formally, our goal is to design a quantization map, denoted as $Q : \mathbb{R}^d \to \{0,1\}^B$, that transforms d-dimensional vectors to a binary string of B bits. If we set $B = b \cdot d$ for some $b \geq 0$, this quantizer will have a bit-width of b, representing the average number of bits used to encode each real-valued coordinate of $\mathbb{R}^d$. Crucially, we require an inverse map, $Q^{-1} : \{0,1\}^B \to \mathbb{R}^d$ that performs dequantization, approximately reconstructing original vectors from their quantized representations. Of course, this transformation is inherently lossy, as Q is not a bijection. So, our primary objective is to minimize distortion, with a specific focus on mean-squared error (MSE) and inner product distortion.

We make no assumptions about the input vector dataset, considering the worst-case scenario. We let the quantizer $Q(\cdot)$ to be randomized, leading to stochastic outputs. Considering randomized quantizers, it is more appropriate to define the expected distortion over the randomness of the quantizer's output. Thus, we aim to design quantizers that for any desired bit-width b minimize the following expected distortion measures for any (worst-case) vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$:

$$\textbf{(MSE)} \qquad D_{\texttt{mse}} := \mathop{\mathbb{E}}_{Q}\left[\left\|\boldsymbol{x} - Q^{-1}\left(Q(\boldsymbol{x})\right)\right\|_2^2\right] \tag{1}$$

$$\textbf{(inner-prod error)} \qquad D_{\texttt{prod}} := \mathop{\mathbb{E}}_{Q}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right|^2\right]. \tag{2}$$

The expectations above are takes with respect to the randomness of the quantizer $Q(\cdot)$. Furthermore, for inner-product quantizers, we require unbiasedness of the inner product estimator, a desirable property for numerous applications. More precisely, we require:

$$\textbf{(unbiased inner-prod)} \qquad \mathop{\mathbb{E}}_{Q}\left[\langle \boldsymbol{y}, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right] = \langle \boldsymbol{y}, \boldsymbol{x}\rangle.$$

We aim to design computationally efficient quantizers $Q_{\texttt{mse}}$ and $Q_{\texttt{prod}}$, that achieve optimal bounds for the distortion measures defined above, for any given bit-width b. Additionally, we aim for $Q_{\texttt{prod}}$ to provide unbiased inner product estimates. In particular, assume that we are given n real-valued vectors $x_1, x_2, \ldots x_n \in \mathbb{R}^d$. We design the following primitives:

- QUANT: efficiently quantizes the dataset and computes $Q(\boldsymbol{x}_1), Q(\boldsymbol{x}_2), \ldots Q(\boldsymbol{x}_n)$.
- DEQUANT: given a quantized dataset, can efficiently reconstruct original vectors by computing $Q^{-1}\left(Q(\boldsymbol{x}_i)\right)$ for any $i \in [n]$.

1.2 Related Work

Beginnings of VQ. The vector quantization theory started by Shannon's seminal work [48, 49] on achievable distortion-rate functions. In 1963, Zador [61] made significant advances by employing high-resolution methods to derive the limiting operational distortion-rate function for fixed-rate quantization at high rates that closely matches Shannon's distortion-rate function. However, Zador did not specifically consider implementable algorithms. Gersho's influential paper [25], further advanced the vector quantization by popularizing high-resolution theory, simplifying Zador's results, introducing lattice vector quantization, and proposing a key conjecture that shaped the field. Despite these theoretical advancements, the practical applicability of vector quantization remained unclear in early years. The most straightforward encoding method, brute-force nearest neighbor search, was computationally expensive, hindering the adoption of VQ in practice.

Online vs Offline Quantization. Online (data-oblivious) quantization methods apply instantly without needing data-specific tuning or calibrations [16, 8, 41, 47, 28]. In contrast, offline (data-dependent) methods require heavy preprocessing and learning to adapt the quantization map to the data, making them unsuitable for dynamic data scenarios [37]. For instance, methods such as those presented in [20, 39, 57, 13] use second-order (Hessian) information to tune the quantization map which requires heavy preprocessing and even in some cases post processing as well.

Online KV Cache Compression. Several approaches have been proposed to compress the KV cache. These include architectural modifications [50, 6, 15] which restructure the transformer to minimize the number of stored key-value pairs. Additionally, pruning or evicting redundant or less critical tokens has emerged as another approach [11, 66, 40, 58, 64, 38, 29].

A simple yet effective approach to reducing KV cache size is quantizing the KV cache. Several quantization techniques have been developed specifically for this purpose [60, 59, 17, 33, 65, 41, 30, 36, 28]. Recently, a new quantization called QJL [62] introduced an efficient, data-oblivious 1-bit quantization approach based on sketching techniques, which provides unbiased estimates for inner product queries. This method does not require tuning or adaptation to the input data and we make use of this technology in our quantizer optimized for inner product distortion.

Product Quantization (PQ). In Near Neighbor (NN) search problem with Euclidean datasets, the index size poses a significant memory bottleneck, often mitigated by quantization techniques, commonly referred to as Product Quantization (PQ) in the NN literature. Many of these algorithms rely on constructing a quantization codebook using variations of k-means during the indexing phase [31, 9, 24, 56, 27]. Therefore, these methods are ill-suited for online settings due to their requirement for extensive preprocessing.

Recently, a grid-based PQ method was introduced in [22], eliminating the need for preprocessing. This approach operates by projecting a uniform grid onto the unit sphere and conducting a search to identify the nearest projection to the data points. While the paper's theoretical guarantees are suboptimal, likely due to loose analysis—as practical performance surpasses theoretical bounds—the grid projection and binary search algorithm is also computationally slow and particularly inefficient

on accelerators like GPU because of their algorithm's inherent lack of vectorization, which prevents parallel processing.

1.3 Overview of Techniques and Contributions

MSE Optimzied TurboQuant. Our first VQ algorithm is designed to minimize MSE distortion deinfed in Eq. (1). To achieve this, we apply a random rotation to the input vectors, thereby inducing a Beta distribution on each coordinate, irrespective of the input vectors themselves. In high dimensions d, the distribution of each coordinate converges to a Gaussian distribution $\mathcal{N}(1, 1/d)$ due to concentration of measure and the central limit theorem. Furthermore, any two distinct coordinates become nearly uncorrelated and, more importantly, almost independent (a deeper result that goes beyond just correlation). This near-independence is a crucial aspect that simplifies our quantization design. It allows us to quantize each coordinate using optimal scalar quantization, disregarding interactions or correlations between different coordinates, while still achieving near-optimal distortion.

We find optimal scalar quantizers for random variables with Beta distributions by solving a continuous 1-dimensional k-means problem using the Max-Lloyd algorithm. We precompute and store these optimal codebooks for a range of practically useful bit-widths, to enable efficient subsequent invocations of our TurboQuant algorithm.

In Theorem 1 we prove that the b-bit MSE optimized TurboQuant $Q_{\texttt{mse}} : \mathbb{R}^d \to \{0,1\}^{b\cdot d}$ achieves the following distortion for any worst-case vector $\boldsymbol{x} \in \mathbb{R}^d$ with $\|\boldsymbol{x}\| = 1$:

- $D_{\texttt{mse}}(Q_{\texttt{mse}}) := \mathbb{E}\left[\left\|\boldsymbol{x} - Q^{-1}_{\texttt{mse}}\left(Q_{\texttt{mse}}(\boldsymbol{x})\right)\right\|_2^2\right] \leq \frac{\sqrt{3}\pi}{2} \cdot \frac{1}{4^b}$ for any $b \geq 0$.
- For small bit-widths the above distortion upper bound can be further refined. Specifically, for $b = 1, 2, 3, 4$ we have $D_{\texttt{mse}}(Q_{\texttt{mse}}) \approx \mathbf{0.36}, \mathbf{0.117}, \mathbf{0.03}, \mathbf{0.009}$, respectively.

Note that the unit norm assumption, $\|x\|_2 = 1$, is standard and not restrictive. For datasets that do not satisfy this assumption we can compute and store the $L2$ norms in floating-point precision and rescale the dequantized points using these stored norms.

Inner Product TurboQuant. We show that the MSE optimized quantizers are biased for inner product estimation and thus a different VQ scheme is needed to get an unbiased inner product quantizer. Our solution is a two stage algorithm that first applies the abovementioned $Q_{\texttt{mse}}$ with a bit-width one less than our target budget and then apply a QJL [62] on the residual error. This is proved to be unbiased and also has nearly optimal inner product error rate.

In Theorem 2 we prove that the b-bit inner product optimized TurboQuant $Q_{\texttt{prod}} : \mathbb{R}^d \to \{0,1\}^{b\cdot d}$ achieves the following distortion for any worst-case vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ with $\|\boldsymbol{x}\| = 1$:

- $\mathbb{E}\left[\left\langle \boldsymbol{y}, Q^{-1}_{\texttt{prod}}\left(Q_{\texttt{prod}}(\boldsymbol{x})\right)\right\rangle\right] = \langle \boldsymbol{y}, \boldsymbol{x}\rangle$
- $D_{\texttt{prod}}(Q_{\texttt{prod}}) := \mathbb{E}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, Q^{-1}_{\texttt{prod}}\left(Q_{\texttt{prod}}(\boldsymbol{x})\right)\rangle\right|^2\right] \leq \frac{\sqrt{3}\pi^2 \cdot \|\boldsymbol{y}\|_2^2}{d} \cdot \frac{1}{4^b}$ for any $b \geq 0$.

- For small bit-widths the above distortion upper bound can be further refined. Specifically, for $b = 1, 2, 3, 4$ we have $D_{\texttt{prod}}(Q_{\texttt{prod}}) \approx \frac{\mathbf{1.57}}{d}, \frac{\mathbf{0.56}}{d}, \frac{\mathbf{0.18}}{d}, \frac{\mathbf{0.047}}{d}$, respectively.

Lower Bound. In Theorem 3, we leverage Shannon's lower bound and Yao's minimax principle to prove that for any randomized quantization algorithm $Q : \mathbb{R}^d \to \{0,1\}^{b \cdot d}$ with bit-width b, there exist hard input instances $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ with $\|\boldsymbol{x}\| = 1$ such that the following lower bounds hold:

- $D_{\texttt{mse}}(Q) := \mathbb{E}\left[\left\|\boldsymbol{x} - Q^{-1}\left(Q(\boldsymbol{x})\right)\right\|_2^2\right] \geq \frac{1}{4^b}$
- $D_{\texttt{prod}}(Q) = \mathbb{E}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right|^2\right] \geq \frac{\|\boldsymbol{y}\|_2^2}{d} \cdot \frac{1}{4^b}$

As demonstrated by our lower bounds, TurboQuant's MSE distortion is provably within a factor of at most $\frac{\sqrt{3}\pi}{2} \approx \mathbf{2.7}$ of the information-theoretical lower bound. Notably, for smaller bit-widths, this factor significantly decreases. For instance, at a bit-width of $b = 1$ TurboQuant achieves a distortion that is only a factor of approximately **1.45** away from the optimal which is also confirmed by our experimental results, indicating its efficiency in low-bit-width scenarios.

Experimental Results. In Section 4.1, we empirically validate our theoretical distortion bounds, demonstrating that TurboQuant's observed distortions closely align with our predictions across various real-world datasets, approaching the established lower bounds.

Furthermore, in Section 4.2 and Section 4.3, we showcase TurboQuant's efficacy in online KV cache quantization. Specifically, we achieve perfect long-context retrieval in needle-in-a-haystack tasks and maintain high performance on other long-context downstream tasks, all while compressing the KV cache by a factor exceeding $5\times$.

Finally in Section 4.4 we apply TurboQuant to various high-dimensional near neighbor search tasks. TurboQuant consistently outperforms data-dependent product quantization (PQ), while reducing the indexing time to essentially zero.

2 Preliminaries

We use boldface lowercase letters, such as $\boldsymbol{x}$ and $\boldsymbol{y}$, to denote vectors, and boldface uppercase letters, like $\boldsymbol{M}$, to denote matrices. To denote a slice of a vector $\boldsymbol{x}$ between the coordinate indices i and j inclusive of the endpoints, we use the notation $\boldsymbol{x}_{i:j}$. For a matrix $\boldsymbol{M}$, we write $\boldsymbol{M}_{i,:}$ to denote its i-th row vector, which we will simply refer to as $\boldsymbol{M}_i$.

We use the notation $\mathbb{S}^{d-1}$ to denote the hypersphere in $\mathbb{R}^d$ of radius 1. For a random variable x we denote its differential entropy as $h(x)$. For random variables x and y, the mutual information between them is denoted as $I(x; y) = h(x) - h(x|y)$.

Given that TurboQuant employs random rotation to mitigate worst-case input scenarios, understanding the statistical properties of random points on a hypersphere is essential. The following lemma outlines one such property that we will need for analysis and design purposes:

Lemma 1 (coordinate distribution of random point on hypersphere)**.** *For any positive integer d if $\boldsymbol{x} \in \mathbb{S}^{d-1}$ is a random variable uniformly distributed over the unit hypersphere, then for any $j \in [d]$ the coordinate $\boldsymbol{x}_j$ follows the following (scaled/shifted) Beta distribution:*

$$\boldsymbol{x}_j \sim f_X(x) := \frac{\Gamma(d/2)}{\sqrt{\pi} \cdot \Gamma((d-1)/2)} \left(1 - x^2\right)^{(d-3)/2}.$$

In high dimensions this beta distribtion converges to the normal distribution $f_X(\cdot) \to \mathcal{N}(0, 1/d)$.

Proof. $f_X(x)$ equals the ratio of the area of a sphere with radius $\sqrt{1-x^2}$ in dimension $d-1$ to the volume of a unit sphere in dimension d scaled down by $1/\sqrt{1-x^2}$ (by Pythagorean theorem). Therefore,

$$f_X(x) = \frac{\frac{2\pi^{(d-1)/2}}{\Gamma((d-1)/2)} \cdot (1-x^2)^{(d-2)/2}}{\frac{2\pi^{d/2}}{\Gamma(d/2)}} \cdot 1/\sqrt{1-x^2} = \frac{\Gamma(d/2)}{\sqrt{\pi} \cdot \Gamma((d-1)/2)} \left(1 - x^2\right)^{(d-3)/2}.$$

□

2.1 Shannon Lower Bound on Distortion

The Shannon Lower Bound (SLB) is a powerful tool, derived from Shannon's lossy source coding theorem [49], that provides a universal lower bound on the optimal achievable distortion rate for any lossy compression scheme. Specifically, we use a version of SLB tailored for the mean-squared error (MSE) distortion measure applied to general d-dimensional sources.

Lemma 2 (SLB)**.** *Let $\boldsymbol{x} \in \mathbb{R}^d$ be a random vector with an arbitrary probability distribution p_X and finite differential entropy $h(\boldsymbol{x})$. Define the MSE distortion-rate function $D(B)$ for total bit complexity $B \geq 0$ as:*

$$D(p_X, B) := \inf \left\{ \mathbb{E}\left[\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right] : I(\boldsymbol{x}; \boldsymbol{y}) \leq B \right\},$$

where the infimum is taken over all joint distributions of $\boldsymbol{x}$ and a reconstruction random vector $\boldsymbol{y} \in \mathbb{R}^d$ such that the mutual information $I(\boldsymbol{x}; \boldsymbol{y})$ is at most B and $\mathbb{E}\left[\|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right]$ is the expected MSE distortion, calculated with respect to the joint distribution of $\boldsymbol{x}$ and $\boldsymbol{y}$. Then, for any bit complexity $B \geq 0$, the following Shannon Lower Bound holds:

$$D(p_X, B) \geq \frac{d}{2\pi e} \cdot 2^{(2/d)(h(\boldsymbol{x}) - B)}.$$

This is a classic result proved using backward Gaussian test channel (for a proof see [14]). Our lower bound result uses a corollary of SLB that corresponds to the uniformly distributed random points on the unit hyeprsphere. We present this in the following lemma:

Lemma 3 (SLB for random point on hypersphere)**.** *Let $\boldsymbol{x} \in \mathbb{S}^{d-1}$ be a random variable uniformly distributed over the unit hypersphere and define the MSE distortion-rate function $D(B)$ for total bit complexity B as per Lemma 2. Then, for any bit complexity $B \geq 0$, the following distortion lower bound holds:*

$$D(B) \geq 2^{-2B/d}.$$

Proof. If we let A_d denote the area of the hypersphere $\mathbb{S}^{d-1}$, the entropy of uniform distribution over hypersphere is $h(\boldsymbol{x}) = \log_2 A_d$. Plugging this into the SLB from Lemma 2 we get $D(B) \geq \frac{d}{2\pi e} \cdot {A_d}^{2/d} \cdot 2^{-2B/d}$. Using Stirling's approximation formula for Gamma function we have $A_d = \frac{2\pi^{d/2}}{\Gamma(d/2)} \geq \left(\frac{2\pi e}{d}\right)^{d/2} \cdot \sqrt{\frac{2d}{\pi}} \cdot (1 - O(1/d))$. By substituting this into the inequality obtained from Lemma 2 we get the desired lower bound. □

2.2 QJL: 1-bit inner product quantization

As previously stated, we design two VQ algorithms: one optimized for minimizing MSE and the other for minimizing inner product error. We show that MSE-optimal quantizers do not necessarily provide unbiased inner product estimates, particularly exhibiting significant bias at lower bit-widths. Our solution for inner product quantization is a two-stage algorithm. First, we apply the MSE-optimal quantizer using one less bit than the desired bit-width budget, thus minimizing the L2 norm of the residuals. Next we apply an unbiased and optimal single-bit quantizer to the residual. For the single-bit inner product quantizer, we utilize the recently proposed Quantized Johnson-Lindenstrauss (QJL) algorithm [62], which is an optimal inner product quantizer with a bit-width of one. Here, we present the QJL algorithm and its essential theoretical guarantees.

Definition 1 (QJL)**.** *For any positive integer d the QJL map $Q_{\texttt{qjl}} : \mathbb{R}^d \to \{-1, +1\}^d$ is defined as:*

$$Q_{\texttt{qjl}}(\boldsymbol{x}) := \texttt{sign}\left(\boldsymbol{S} \cdot \boldsymbol{x}\right) \qquad \text{for any } \boldsymbol{x} \in \mathbb{R}^d,$$

where $\boldsymbol{S} \in \mathbb{R}^{d \times d}$ is a random matrix with i.i.d. entries sampled from the normal distribution $\mathcal{N}(0, 1)$ and the sign *function is applied entry-wise to its vector input. The inverse/dequantization map $Q^{-1}_{\texttt{qjl}} : \{-1, +1\}^d \to \mathbb{R}^d$ is defined as:*

$$Q^{-1}_{\texttt{qjl}}(\boldsymbol{z}) := \frac{\sqrt{\pi/2}}{d} \cdot \boldsymbol{S}^\top \cdot \boldsymbol{z} \qquad \text{for any } \boldsymbol{z} \in \{-1, +1\}^d.$$

In the next lemma we restate the results from [62] that show the QJL is unbiased and also has small inner product distortion:

Lemma 4 (performance guarantee: QJL)**.** *Let $Q_{\texttt{qjl}}$ and $Q^{-1}_{\texttt{qjl}}$ be defined as per Definition 1. For any vector $\boldsymbol{x} \in \mathbb{S}^{d-1}$ and any $\boldsymbol{y} \in \mathbb{R}^d$ we have the following:*

- *Unbiased:* $\mathbb{E}\left[\left\langle \boldsymbol{y}, Q^{-1}_{\texttt{qjl}}\left(Q_{\texttt{qjl}}(\boldsymbol{x})\right)\right\rangle\right] = \langle \boldsymbol{y}, \boldsymbol{x}\rangle$.
- *Variance Bound:* $\texttt{Var}\left(\left\langle \boldsymbol{y}, Q^{-1}_{\texttt{qjl}}\left(Q_{\texttt{qjl}}(\boldsymbol{x})\right)\right\rangle\right) \leq \frac{\pi}{2d} \cdot \|\boldsymbol{y}\|_2^2$

Proof. The unbiasedness immediately follows from Lemma 3.2 of [62]. To show the variance bound let $\boldsymbol{s}_1, \boldsymbol{s}_2, \ldots \boldsymbol{s}_m$ denote the rows of the random matrix $\boldsymbol{S}$ in Definition 1. We have:

$$\left\langle \boldsymbol{y}, Q^{-1}_{\texttt{qjl}}\left(Q_{\texttt{qjl}}(\boldsymbol{x})\right)\right\rangle = \frac{1}{d} \sum_{i \in [d]} \sqrt{\pi/2} \cdot \boldsymbol{s}_i^\top \boldsymbol{y} \cdot \texttt{sign}(\boldsymbol{s}_i^\top \boldsymbol{x}).$$

Since $\boldsymbol{s}_i$'s are i.i.d. the above is indeed the average of d i.i.d. random samples defined as $z_i := \sqrt{\pi/2} \cdot \boldsymbol{s}_i^\top \boldsymbol{y} \cdot \texttt{sign}(\boldsymbol{s}_i^\top \boldsymbol{x})$ for $i \in [d]$. Let us now upper bound the variance of a single z_i using Fact 3.4 from [62]:

$$\texttt{Var}\left(z_i\right) = \pi/2 \cdot \texttt{Var}\left(\boldsymbol{s}_i^\top \boldsymbol{y} \cdot \texttt{sign}(\boldsymbol{s}_i^\top \boldsymbol{x})\right) \leq \pi/2 \cdot \mathbb{E}\left[(\boldsymbol{s}_i^\top \boldsymbol{y})^2\right] = \pi/2 \cdot \|y\|_2^2 , \tag{3}$$

where the last equality above follows because $\boldsymbol{s}_i^\top \boldsymbol{y}$ is a Gaussian random variable with mean zero and variance $\|\boldsymbol{y}\|_2^2$. Now the variance of the average of d i.i.d. random samples $z_1, z_2, \ldots z_d$ is:

$$\texttt{Var}\left(\left\langle \boldsymbol{y}, Q_{\texttt{qjl}}^{-1}\left(Q_{\texttt{qjl}}(\boldsymbol{x})\right)\right\rangle\right) = \frac{1}{d^2}\sum_{i\in[d]} \texttt{Var}(z_i) \leq \frac{\pi}{2d} \cdot \|\boldsymbol{y}\|_2^2 .$$

□

3 TurboQuant: High Performance Quantization

We developed two VQ algorithms, each tailored to a specific objective. The first algorithm is designed to minimize the MSE between the original and reconstructed vectors after quantization. The second algorithm is optimized for unbiased inner product estimation, addressing the bias inherent in MSE-optimal quantizers. These algorithms are detailed in the following subsections.

Furthermore, in Section 3.3, we establish information-theoretic lower bounds on the best achievable distortion rates for any vector quantizer. This analysis demonstrates that TurboQuant achieve near-optimality, differing from the lower bound by only a small constant factor across all bit-widths.

3.1 MSE Optimal TurboQuant

Let $\boldsymbol{x} \in \mathbb{S}^{d-1}$ be a (worst-case) vector on the unit sphere in dimension d. We aim to quantize $\boldsymbol{x}$ to b bits per coordinate while minimizing the reconstruction MSE defined in Eq. (1). We start by randomizing this vector by multiplying it with a random rotation matrix $\boldsymbol{\Pi} \in \mathbb{R}^{d\times d}$. We can generate $\boldsymbol{\Pi}$ by applying QR decomposition on a random matrix with i.i.d Normal entries.

The resulting rotated vector, $\boldsymbol{\Pi} \cdot \boldsymbol{x}$, is uniformly distributed on the unit sphere $\mathbb{S}^{d-1}$. As shown in Lemma 1, each coordinate of $\boldsymbol{\Pi} \cdot \boldsymbol{x}$ follows a Beta distribution, which converges to a normal distribution in high dimensions. Furthermore, in high dimensions, distinct coordinates of $\boldsymbol{\Pi} \cdot \boldsymbol{x}$ become nearly independent [55], allowing us to apply optimal scalar quantizers to each coordinate independently. Therefore, by Lemma 1, our task reduces to designing a scalar quantizer for random variables with the distribution $f_X(x) = \frac{\Gamma(d/2)}{\sqrt{\pi}\cdot\Gamma((d-1)/2)}\left(1-x^2\right)^{(d-3)/2}$ for $x \in [-1, 1]$.

The optimal scalar quantization problem, given a known probability distribution, can be framed as a continuous k-means problem in dimension one. Specifically, we aim to partition the interval $[-1, 1]$ into 2^b clusters/buckets. The optimal solution adheres to a Voronoi tessellation [42], meaning interval boundaries are the midpoints between consecutive centroids, when arranged in sorted order. Therefore, with c_i's denoting the centroids in ascending order, we can formulate the scalar

Algorithm 1 TurboQuant$_{\texttt{mse}}$: optimized for MSE

input: dimension d and bit-width b

// Global Parameters for Setting up TurboQuant$_{\texttt{mse}}$

Generate a random rotation matrix $\boldsymbol{\Pi} \in \mathbb{R}^{d\times d}$

Construct codebook by finding centroids $c_1, c_2, \ldots c_{2^b} \in [-1, 1]$ that minimize MSE cost in Eq. (4)

Procedure Quant$_{\texttt{mse}}(\boldsymbol{x})$

$\boldsymbol{y} \leftarrow \boldsymbol{\Pi} \cdot \boldsymbol{x}$

$\texttt{idx}_j \leftarrow \arg\min_{k\in[2^b]} |\boldsymbol{y}_j - c_k|$ for every $j \in [d]$ {$\texttt{idx}_j$'s are b-bit integers}

output: idx

Procedure DeQuant$_{\texttt{mse}}$(idx)

$\tilde{\boldsymbol{y}}_j \leftarrow c_{\texttt{idx}_j}$ for every $j \in [d]$

$\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{\Pi}^\top \cdot \tilde{\boldsymbol{y}}$

output: $\tilde{\boldsymbol{x}}$

quantization as the following k-means optimization problem:

$$\mathcal{C}(f_X, b) := \min_{-1\le c_1\le c_2\le\ldots\le c_{2^b}\le 1} \sum_{i=1}^{2^b} \int_{\frac{c_{i-1}+c_i}{2}}^{\frac{c_i+c_{i+1}}{2}} |x - c_i|^2 \cdot f_X(x)\, dx. \tag{4}$$

Note that $\mathcal{C}(f_X, b)$ in Eq. (4) denotes the optimal MSE cost function for bit-width b, a quantity we will bound to prove the upper bound on the end-to-end MSE of TurboQuant. The problem in Eq. (4) can be solved using iterative numerical methods to achieve any desired precision. We solve Eq. (4) for a range of practically relevant bit-widths b once, and store the results for future uses by the quantizer.

For example, in moderately high dimensions d, where the distribution $f_X(x)$ closely approximates a normal distribution, the optimal quantization centroids for bit-widths $b = 1, 2$ are $\left\{\pm\frac{\sqrt{2/\pi}}{\sqrt{d}}\right\}$ and $\left\{\pm\frac{0.453}{\sqrt{d}}, \pm\frac{1.51}{\sqrt{d}}\right\}$, respectively.

Therefore the quantizer $Q_{\texttt{mse}} : \mathbb{R}^d \to \{0,1\}^{b\cdot d}$ first computes $\boldsymbol{\Pi} \cdot \boldsymbol{x}$ and then computes and stores the indices of the nearest centroids to each coordinate of this vector. The dequantization map $Q^{-1}_{\texttt{mse}} : \{0,1\}^{b\cdot d} \to \mathbb{R}^d$ reconstructs the vector by retrieving the centroids corresponding to the stored indices and then rotating the result back to the original basis through multiplication with $\boldsymbol{\Pi}^\top$. A pseudocode for these procedures is given in Algorithm 1.

We are now ready to prove our main theorem for TurboQuant$_{\texttt{mse}}$.

Theorem 1 (performance guarantee: TurboQuant$_{\texttt{mse}}$)**.** *For any bit-width $b \ge 1$ and any vector $\boldsymbol{x} \in \mathbb{S}^{d-1}$, the procedure* Quant$_{\texttt{mse}}(\boldsymbol{x})$ *in Algorithm 1 outputs an index vector* $\texttt{idx} \in [2^b]^d$*. When this index vector is passed to the primitive* DeQuant$_{\texttt{mse}}$(idx)*, it produces a reconstructed vector $\tilde{\boldsymbol{x}} \in \mathbb{R}^d$ that satisfies the following distortion bounds:*

- *MSE defined as $D_{\texttt{mse}} := \mathbb{E}_{\tilde{\boldsymbol{x}}}[\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2^2]$ is bounded by $D_{\texttt{mse}} \le \frac{\sqrt{3}\pi}{2} \cdot \frac{1}{4^b}$ for any $b \ge 0$.*

- *For small bit-widths, specifically $b = 1, 2, 3, 4$ the MSE exhibits finer-grained distortion values: $D_{\texttt{mse}} \approx \mathbf{0.36}, \mathbf{0.117}, \mathbf{0.03}, \mathbf{0.009}$, respectively.*

Proof. We start the proof by showing that $D_{\texttt{mse}} = d \cdot \mathcal{C}(f_X, b)$, where $\mathcal{C}(f_X, b)$ is the optimal MSE cost for scalar quantizer defined in Eq. (4). Let $\tilde{\boldsymbol{y}}$ be defined as per line 9 of Algorithm 1. Since $\boldsymbol{\Pi}$ is a rotation matrix we can write: $\|\boldsymbol{x} - \tilde{\boldsymbol{x}}\|_2 = \|\boldsymbol{\Pi} \cdot \boldsymbol{x} - \tilde{\boldsymbol{y}}\|_2$. Using the notation $\boldsymbol{y} = \boldsymbol{\Pi} \cdot \boldsymbol{x}$ as per line 5 of Algorithm 1 and plugging this into the definition of $D_{\texttt{mse}}$ we can write:

$$\begin{aligned}
D_{\texttt{mse}} &= \mathbb{E}[\|\boldsymbol{y} - \tilde{\boldsymbol{y}}\|_2^2] \\
&= \sum_{j \in [d]} \mathbb{E}\left[|\boldsymbol{y}_j - \tilde{\boldsymbol{y}}_j|^2\right] \\
&= \sum_{j \in [d]} \mathbb{E}\left[|\boldsymbol{y}_j - c_{\texttt{idx}_j}|^2\right] \\
&= d \cdot \mathbb{E}\left[|\boldsymbol{y}_1 - c_{\texttt{idx}_1}|^2\right] \\
&= d \cdot \min_{-1 \le c_1 \le c_2 \le \ldots \le c_{2^b} \le 1} \sum_{i=1}^{2^b} \int_{\frac{c_{i-1}+c_i}{2}}^{\frac{c_i+c_{i+1}}{2}} |x - c_i|^2 \cdot f_X(x)\, dx \\
&= d \cdot \mathcal{C}(f_X, b).
\end{aligned}$$

The third equality above follows from the definition of $\tilde{\boldsymbol{y}}$ in line 9 of Algorithm 1 and the fourth line above follows because all $\boldsymbol{y}_j$'s have identical distribution of $\boldsymbol{y}_j \sim f_X(\cdot)$ as shown in Lemma 1. The last two lines above follows because $c_{\texttt{idx}_j}$ is chosen to be the nearest centroid to each coordinate $\boldsymbol{y}_j$ in line 6.

Now we must bound the optimal k-means cost $\mathcal{C}(f_X, b)$. For moderate values of d, $f_X \to \mathcal{N}(0, 1/d)$. By numerically solving the optimization problem in Eq. (4) for values $b = 1, 2, 3, 4$ we get that $\mathcal{C}(f_X, b) \approx \frac{0.36}{d}, \frac{0.117}{d}, \frac{0.03}{d}, \frac{0.009}{d}$, respectively. For larger bit-widths $b > 4$, we can apply the Panter-Dite [44] high-resolution formula for the distortion of a fixed-rate scalar quantizer, yielding the following bound:

$$\mathcal{C}(f_X, b) \le \frac{1}{12} \cdot \left(\int f_X(x)^{1/3}\, dx \right)^3 \cdot \frac{1}{4^b} = \frac{\sqrt{3}\pi}{2d} \cdot \frac{1}{4^b}.$$

This completes the proof. □

Entropy Encoding Codebook Pointers. TurboQuant's efficiency can be further increased by applying entropy encoding to the indices that point to the closest codebook elements. Specifically, the probability of each codeword index appearing in the quantized vectors can be computed as $p_\ell := \int_{\frac{c_{\ell-1}+c_\ell}{2}}^{\frac{c_\ell+c_{\ell+1}}{2}} f_X(x)\, dx$. Optimally coding the indices, reduces the average bit-width to nearly the entropy of the distribution $\{p_i\}_{i \in [2^b]}$. This lossless compression does not affect the distortion and provides a bit-width reduction at no cost. The most significant reduction occurs for $b = 4$, where the entropy of $\{p_i\}_{i \in [2^b]}$ is approximately 3.8. Detailed calculations for optimal prefix codes reveal that the average bit-width can be reduced by 5%. However, given the limited gain, we have chosen not to incorporate this technique into TurboQuant to maintain simplicity and speed.

Algorithm 2 TurboQuant$_{\texttt{prod}}$: optimized for inner product

input: dimension d and bit-width b
// Global Parameters for Setting up TurboQuant$_{\texttt{prod}}$
Instantiate a TurboQuant$_{\texttt{mse}}$ with bit-width $b-1$ as per Algorithm 1
Generate a random projection matrix $\boldsymbol{S} \in \mathbb{R}^{d\times d}$ with i.i.d. entries $\boldsymbol{S}_{i,j} \sim \mathcal{N}(0,1)$

Procedure Quant$_{\texttt{prod}}(\boldsymbol{x})$
$\texttt{idx} \leftarrow$ Quant$_{\texttt{mse}}(\boldsymbol{x})$
$\boldsymbol{r} \leftarrow \boldsymbol{x} -$ DeQuant$_{\texttt{mse}}(\texttt{idx})$ {residual vector}
$\texttt{qjl} \leftarrow \texttt{sign}\left(\boldsymbol{S}\cdot\boldsymbol{r}\right)$ {QJL on residual vector}
output: $(\texttt{idx}, \texttt{qjl}, \|\boldsymbol{r}\|_2)$

Procedure DeQuant$_{\texttt{prod}}(\texttt{idx}, \texttt{qjl}, \gamma)$
$\tilde{\boldsymbol{x}}_{\texttt{mse}} \leftarrow$ DeQuant$_{\texttt{mse}}(\texttt{idx})$
$\tilde{\boldsymbol{x}}_{\texttt{qjl}} \leftarrow \frac{\sqrt{\pi/2}}{d}\cdot\gamma\cdot\boldsymbol{S}^\top\cdot\texttt{qjl}$
output: $\tilde{\boldsymbol{x}}_{\texttt{mse}} + \tilde{\boldsymbol{x}}_{\texttt{qjl}}$

3.2 Inner-product Optimal TurboQuant

For important applications like nearest neighbor search, having an unbiased inner product estimator is essential. However, TurboQuant$_{\texttt{mse}}$ presented in Section 3.1 does not provide unbiased inner product estimates with query vectors. To illustrate this, consider the case with a bit-width of $b = 1$. In this scenario, the optimal codebooks that solve the optimization problem in Eq. (4), for sufficiently large d, are $\left\{\pm\sqrt{\frac{2}{\pi d}}\right\}$. This implies that the quantization map for TurboQuant$_{\texttt{mse}}$ is $Q_{\texttt{mse}}(\boldsymbol{x}) = \texttt{sign}\left(\boldsymbol{\Pi}\cdot\boldsymbol{x}\right)$ for any $\boldsymbol{x}\in\mathbb{R}^d$, and the dequantization map is $Q^{-1}_{\texttt{mse}}(\boldsymbol{z}) = \sqrt{\frac{2}{\pi d}}\cdot\boldsymbol{\Pi}^\top\cdot\boldsymbol{z}$ for any $\boldsymbol{z}\in\{-1,+1\}^d$. Therefore, for large enough d, according to Lemma 4, we have $\mathbb{E}\left[\left\langle\boldsymbol{y}, Q^{-1}_{\texttt{mse}}\left(Q_{\texttt{mse}}(\boldsymbol{x})\right)\right\rangle\right] = \frac{2}{\pi}\cdot\langle\boldsymbol{y},\boldsymbol{x}\rangle$, which has a multiplicative bias of $2/\pi$. This bias diminishes with increasing bit-widths b, as we empirically demonstrate in Section 4.1.

To address this bias, we propose a solution that combines TurboQuant$_{\texttt{mse}}$ with an instance of QJL [62]. Specifically, let $Q_{\texttt{mse}}$ be the quantization map corresponding to TurboQuant$_{\texttt{mse}}$ with a bit-width of $b-1$. For any $\boldsymbol{x}\in\mathbb{S}^{d-1}$ the residual vector, defined as $\boldsymbol{r} := \boldsymbol{x} - Q^{-1}_{\texttt{mse}}\left(Q_{\texttt{mse}}(\boldsymbol{x})\right)$, has a small L2 norm, i.e., on expectation $\mathbb{E}[\|\boldsymbol{r}\|] = \sqrt{\mathcal{C}(f_X, b-1)}$ (per Eq. (4)). We can then apply the QJL quantization map $Q_{\texttt{qjl}}$ on this residual vector, resulting in an overall bit-width of b and providing the following unbiased inner product estimator:

$$\left\langle\boldsymbol{y}, Q^{-1}_{\texttt{mse}}\left(Q_{\texttt{mse}}(\boldsymbol{x})\right)\right\rangle + \|\boldsymbol{r}\|_2\cdot\left\langle\boldsymbol{y}, Q^{-1}_{\texttt{qjl}}\left(Q_{\texttt{qjl}}(\boldsymbol{r})\right)\right\rangle.$$

More formally, the quantization map $Q_{\texttt{prod}}:\mathbb{S}^{d-1}\to[2^{b-1}]^d\times\{-1,1\}^d\times\mathbb{R}$ is defined as:

$$Q_{\texttt{prod}}(\boldsymbol{x}) = \left[Q_{\texttt{mse}}(\boldsymbol{x}), Q_{\texttt{qjl}}\left(\boldsymbol{x} - Q^{-1}_{\texttt{mse}}\left(Q_{\texttt{mse}}(\boldsymbol{x})\right)\right), \left\|\boldsymbol{x} - Q^{-1}_{\texttt{mse}}\left(Q_{\texttt{mse}}(\boldsymbol{x})\right)\right\|_2\right].$$

A pseudocode for this procedure is given in Algorithm 2.

We prove the main result for TurboQuant$_{\texttt{prod}}$ in the following theorem.

Theorem 2 (performance guarantee: TurboQuant$_{\texttt{prod}}$)**.** *For any bit-width $b \geq 1$ and any vector $\boldsymbol{x} \in \mathbb{S}^{d-1}$, the procedure* Quant$_{\texttt{prod}}(\boldsymbol{x})$ *in Algorithm 2 outputs an index vector* $\texttt{idx} \in [2^{b-1}]^d$ *along with a sign vector* $\texttt{qjl} \in \{-1, 1\}^d$ *and a positive number $\gamma \geq 0$. When these vectors and the scalar value are passed to the primitive* DeQuant$_{\texttt{prod}}(\texttt{idx}, \texttt{qjl}, \gamma)$*, it produces a reconstructed vector $\tilde{\boldsymbol{x}} \in \mathbb{R}^d$ that for any vector $\boldsymbol{y} \in \mathbb{R}^d$ satisfies the following properties:*

- *Expected inner-product* $\mathbb{E}_{\tilde{\boldsymbol{x}}}\left[\langle \boldsymbol{y}, \tilde{\boldsymbol{x}} \rangle\right] = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$
- *Inner-product distortion defined as* $D_{\texttt{prod}} := \mathbb{E}_{\tilde{\boldsymbol{x}}}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x} \rangle - \langle \boldsymbol{y}, \tilde{\boldsymbol{x}} \rangle\right|^2\right]$ *is bounded by* $D_{\texttt{prod}} \leq \frac{\sqrt{3}\pi^2 \cdot \|\boldsymbol{y}\|_2^2}{d} \cdot \frac{1}{4^b}$ *for any $b \geq 0$.*
- *For small bit-widths, specifically $b = 1, 2, 3, 4$, $D_{\texttt{prod}}$ exhibits finer-grained distortion values:* $D_{\texttt{prod}} \approx \frac{\mathbf{1.57}}{d}, \frac{\mathbf{0.56}}{d}, \frac{\mathbf{0.18}}{d}, \frac{\mathbf{0.047}}{d}$*, respectively.*

Proof. First we compute the conditional expectation of the inner product estimate $\langle \boldsymbol{y}, \tilde{\boldsymbol{x}} \rangle$ conditioned on $\tilde{\boldsymbol{x}}_{\texttt{mse}}$ as follows:

$$\begin{aligned}
\mathbb{E}\left[\langle \boldsymbol{y}, \tilde{\boldsymbol{x}} \rangle | \tilde{\boldsymbol{x}}_{\texttt{mse}}\right] &= \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_{\texttt{qjl}}}\left[\langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{mse}} + \tilde{\boldsymbol{x}}_{\texttt{qjl}} \rangle | \tilde{\boldsymbol{x}}_{\texttt{mse}}\right] \\
&= \langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{mse}} \rangle + \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_{\texttt{qjl}}}\left[\langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{qjl}} \rangle | \tilde{\boldsymbol{x}}_{\texttt{mse}}\right] \\
&= \langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{mse}} \rangle + \langle \boldsymbol{y}, \boldsymbol{r} \rangle \\
&= \langle \boldsymbol{y}, \boldsymbol{x} \rangle,
\end{aligned}$$

where the first equality follows from the definition of $\tilde{\boldsymbol{x}}$ in line 12 of the algorithm. The third equality above follows from Lemma 4 and last line follows from definition of the residual vector $\boldsymbol{r} = \boldsymbol{x} - \tilde{\boldsymbol{x}}_{\texttt{mse}}$ in line 6. Now we can computed the unconditional expectation using the law of total expectation: $\mathbb{E}_{\tilde{\boldsymbol{x}}}\left[\langle \boldsymbol{y}, \tilde{\boldsymbol{x}} \rangle\right] = \mathbb{E}_{\tilde{\boldsymbol{x}}_{\texttt{mse}}}\left[\mathbb{E}\left[\langle \boldsymbol{y}, \tilde{\boldsymbol{x}} \rangle | \tilde{\boldsymbol{x}}_{\texttt{mse}}\right]\right] = \mathbb{E}[\langle \boldsymbol{y}, \boldsymbol{x} \rangle] = \langle \boldsymbol{y}, \boldsymbol{x} \rangle$, which proves the first claim of the theorem.

We apply the same conditioning on $\tilde{\boldsymbol{x}}_{\texttt{mse}}$, when computing the distortion, and then compute the resulting conditional distortion:

$$\begin{aligned}
\mathbb{E}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x} \rangle - \langle \boldsymbol{y}, \tilde{\boldsymbol{x}} \rangle\right|^2 \middle| \tilde{\boldsymbol{x}}_{\texttt{mse}}\right] &= \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_{\texttt{qjl}}}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x} \rangle - \langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{mse}} + \tilde{\boldsymbol{x}}_{\texttt{qjl}} \rangle\right|^2 \middle| \tilde{\boldsymbol{x}}_{\texttt{mse}}\right] \\
&= \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_{\texttt{qjl}}}\left[\left|\langle \boldsymbol{y}, \boldsymbol{r} \rangle - \langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{qjl}} \rangle\right|^2 \middle| \tilde{\boldsymbol{x}}_{\texttt{mse}}\right] \\
&= \texttt{Var}\left(\langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{qjl}} \rangle \middle| \tilde{\boldsymbol{x}}_{\texttt{mse}}\right) \\
&\leq \frac{\pi}{2d} \cdot \|\boldsymbol{r}\|_2^2 \|\boldsymbol{y}\|_2^2,
\end{aligned}$$

where the second equality above follows from the definitions of $\boldsymbol{r}$ and $\tilde{\boldsymbol{x}}_{\texttt{mse}}$ in lines 6 and 10 of Algorithm 2. The third line above follows because $\mathbb{E}[\langle \boldsymbol{y}, \tilde{\boldsymbol{x}}_{\texttt{qjl}} \rangle] = \langle \boldsymbol{y}, \boldsymbol{r} \rangle$, by Lemma 4. The last line follows from the variance bound of QJL estimator shown in Lemma 4 and using the fact that $\tilde{\boldsymbol{x}}_{\texttt{qjl}}$ in line 11 is re-scaled by $\gamma = \|\boldsymbol{r}\|$.

Now by law of total expectation along with the fact that $\boldsymbol{r} = \boldsymbol{x} - \tilde{\boldsymbol{x}}_{\texttt{mse}}$ we can bound the inner product distortion as follows:

$$\begin{aligned} D_{\texttt{prod}} &= \mathop{\mathbb{E}}_{\tilde{\boldsymbol{x}}_{\texttt{mse}}}\left[\mathbb{E}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, \tilde{\boldsymbol{x}}\rangle\right|^2 \middle| \tilde{\boldsymbol{x}}_{\texttt{mse}}\right]\right] \\ &\leq \frac{\pi}{2d} \cdot \|\boldsymbol{y}\|_2^2 \cdot \mathbb{E}[\|\boldsymbol{x} - \tilde{\boldsymbol{x}}_{\texttt{mse}}\|_2^2] \\ &= \frac{\pi}{2d} \cdot \|\boldsymbol{y}\|_2^2 \cdot D_{\texttt{mse}}. \end{aligned}$$

The theorem follows by invoking the MSE bounds from Theorem 1 with bit-width $b-1$. □

3.3 Lower Bounds

We show that TurboQuant achieves an optimal distortion rate, up to a small constant factor, for any bit-width by proving lower bounds on the best achievable distortion for any compression algorithm. Our lower bound proof leverages Yao's minimax principle. This principle allows us to relate the lower bound for randomized algorithms with worst-case deterministic input vectors to the lower bound for deterministic algorithms with randomized input vectors. Subsequently, we derive a lower bound on the achievable distortion rate for the latter using Shannon's lower bound (SLB) presented in Section 2.1. Formally, we prove the following theorem.

Theorem 3 (lower bound on best achievable compression distortion)**.** *For any randomized quantization algorithm $Q : \mathbb{S}^{d-1} \to \{0,1\}^{b \cdot d}$ with bit-width b and any reconstruction map $Q^{-1} : \{0,1\}^{b \cdot d} \to \mathbb{R}^d$, there exist a hard input instance $\boldsymbol{x} \in \mathbb{S}^{d-1}$ such that:*

$$D_{\texttt{mse}}(Q) := \mathbb{E}\left[\left\|\boldsymbol{x} - Q^{-1}\left(Q(\boldsymbol{x})\right)\right\|_2^2\right] \geq \frac{1}{4^b}.$$

Furthermore, there exists a $\boldsymbol{y} \in \mathbb{S}^{d-1}$ such that:

$$D_{\texttt{prod}}(Q) = \mathbb{E}\left[\left|\langle \boldsymbol{y}, \boldsymbol{x}\rangle - \langle \boldsymbol{y}, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right|^2\right] \geq \frac{1}{d} \cdot \frac{1}{4^b}$$

Proof. By Yao's minimax principle the expected MSE of the optimal randomized compression algorithm for worst-case inputs ($D_{\texttt{mse}}$) is equal to the expected MSE of the optimal deterministic compression algorithm when applied to inputs drawn from a maximally difficult randomized distribution. By definition, the MSE of the latter scenario is lower-bounded by the best achievable MSE for inputs uniformly distributed on the unit hypersphere.

The best achievable MSE for a compression algorithm with bit-width b, operating on uniformly distributed inputs from the sphere $\mathbb{S}^{d-1}$, is lower bounded in Lemma 3. Therefore, by invoking Lemma 3 we conclude that $D_{\texttt{mse}} \geq \frac{1}{4^b}$.

Furthermore, from $D_{\texttt{mse}} \geq \frac{1}{4^b}$ and using the definition of $D_{\texttt{mse}}$ we conclude that:

$$\begin{aligned} D_{\texttt{mse}} &= \sum_{j=1}^{d} \mathbb{E}\left[\left|\boldsymbol{x}_j - \left[Q^{-1}\left(Q(\boldsymbol{x})\right)\right]_j\right|^2\right] \\ &= \sum_{j=1}^{d} \mathbb{E}\left[\left|\langle \boldsymbol{e}_j, \boldsymbol{x}\rangle - \langle \boldsymbol{e}_j, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right|^2\right] \\ &\geq \frac{1}{4^b}. \end{aligned}$$

By pigeonhole principle there exist an index $j \in [d]$ such that $\mathbb{E}\left[\left|\langle \boldsymbol{e}_j, \boldsymbol{x}\rangle - \langle \boldsymbol{e}_j, Q^{-1}\left(Q(\boldsymbol{x})\right)\rangle\right|^2\right] \geq \frac{1}{d} \cdot \frac{1}{4^b}$, which completes the proof. □

We note that a comparable lower bound for the *worst-case* distortion in vector quantization can be derived using "sphere packing" arguments (indeed, with larger constants as this is a harder problem) [26]. However, Theorem 3 offers a more robust and relevant lower bound for our analysis. This is because it establishes a lower bound on the *expected distortion*, rather than the worst-case error, and aligns seamlessly with our upper bounds presented in Theorem 1 and Theorem 2.

4 Experiments

All experiments are performed using a single NVIDIA A100 GPU. The experimental section is divided into two parts: one to empirically validate the theoretical results, and another to evaluate the performance of our methods on downstream tasks, specifically KV cache quantization and nearest neighbor vector search.

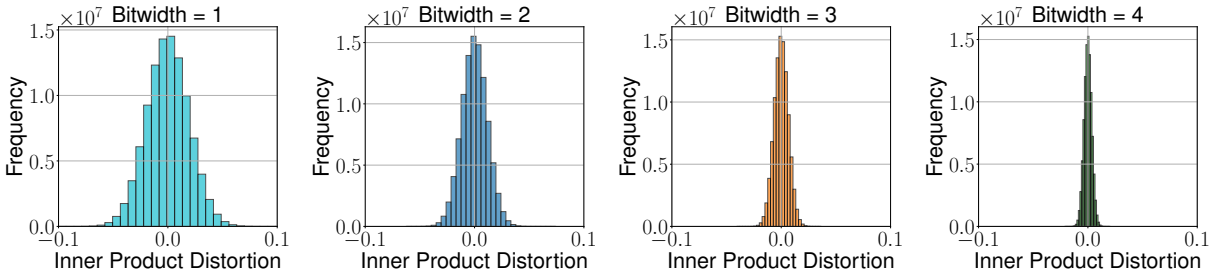
4.1 Empirical Validation

In this section, we verify the theoretical results established in previous sections. We conduct our experiments using the DBpedia Entities dataset, which has been encoded into a 1536-dimensional space using OpenAI3 embeddings. To perform our experiments, we randomly sample 100,000 data points from the dataset, denoted as training set, which serves as our primary dataset. Additionally, we extract 1,000 distinct entries, denoted as query set, to be used as query points.

We evaluate two quantization methods: TurboQuant$_{\texttt{prod}}$ and TurboQuant$_{\texttt{mse}}$. The method TurboQuant$_{\texttt{mse}}$ is designed to be optimzed for estimating the mean squared error (MSE) between the quantized and original vectors. In contrast, TurboQuant$_{\texttt{prod}}$ is unbiased for estimating the inner product between the quantized and original vectors.

Both methods are applied to the task of inner product estimation by quantizing training set and analyzing the distortion in inner product calculations across different bit widths. As shown in Fig. 1, increasing the bit width reduces variance in both methods. However, when used for inner product estimation, TurboQuant$_{\texttt{mse}}$ introduces bias. This bias diminishes as the bit width increases and eventually converges to zero.

(a) TurboQuant$_{\texttt{prod}}$



(b) TurboQuant$_{\texttt{mse}}$

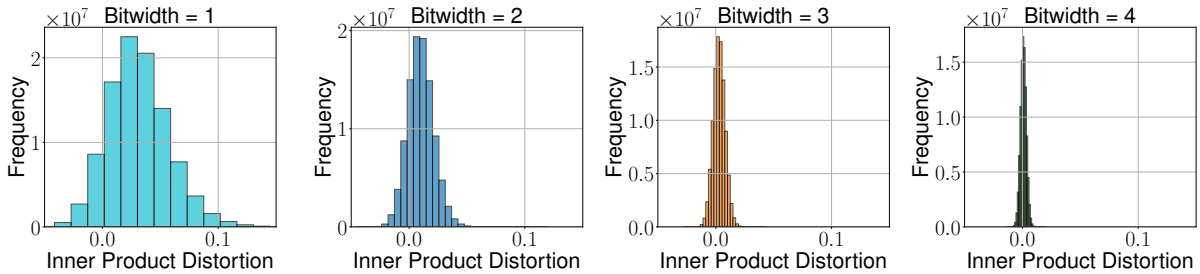


Figure 1: Error distribution of TurboQuant$_{\texttt{prod}}$ and TurboQuant$_{\texttt{mse}}$ for Inner Product Estimation.

The experimental results, illustrated in Fig. 1, confirm that TurboQuant$_{\texttt{prod}}$ remains unbiased for inner product estimation across all bit widths, while TurboQuant$_{\texttt{mse}}$ gradually improves with increasing bit width.

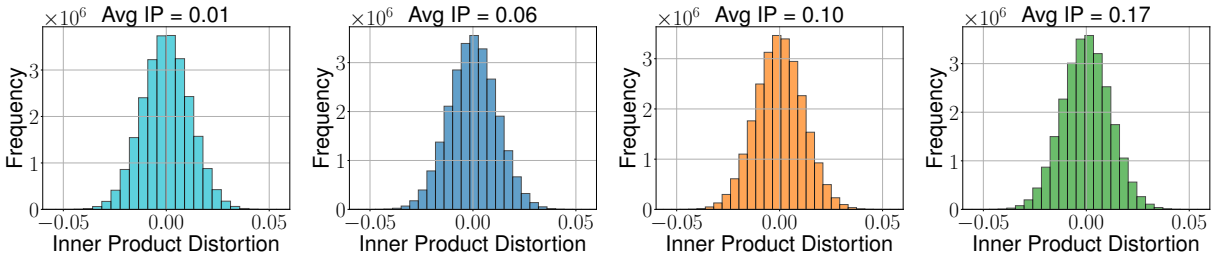
As observed in Fig. 2, when quantizing to 2 bits, the variance remains constant regardless of the inner product of the original vector in the TurboQuant`prod` approach. However, the same plot indicates that the bias in the TurboQuant`mse` approach is dependent on the average inner product. As the average inner product increases, the bias also increases.

Along with the histograms, we also plot Section 4.1 the average inner product error and MSE between the original and quantized vectors across different bit ratios. These plots are drawn alongside the upper and lower bounds established in our theoretical analysis. Our observations confirm that the results align with the theoretical predictions. Specifically, for inner product estimation, the TurboQuant`prod` approach performs better at lower bit ratios. However, as the bit count increases, TurboQuant`mse` reduces bias and ultimately achieves superior performance in inner product estimation.

4.2 Needle-In-A-Haystack

The "Needle-In-A-Haystack Test"" [32] is a benchmark designed to evaluate a model's ability to retrieve specific information embedded within a long document. The test involves placing a unique

(a) TurboQuant$_{\texttt{prod}}$



(b) TurboQuant$_{\texttt{mse}}$

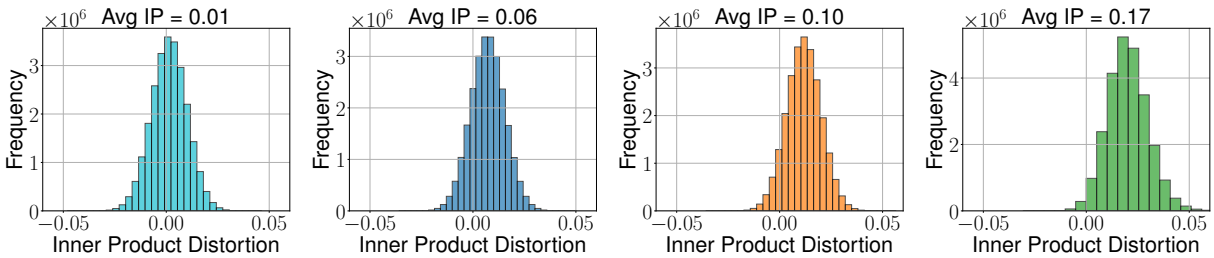


Figure 2: The variance of Inner-product error remains constant for TurboQuant$_{\texttt{prod}}$, while in TurboQuant$_{\texttt{mse}}$ increases with the average inner product. Bit-width is $b = 2$.

sentence (the "needle") at an arbitrary location within a much larger text (the "haystack") and assessing whether the model can successfully extract it.

Following the experimental setup of Fu et al. [21], we conduct evaluations using the `Llama-3.1-8B-Instruct` model. To analyze performance across different input sequence lengths, we vary the document size from *4k to 104k tokens*. The primary metric used for evaluation is the *recall score*, which measures how accurately the model retrieves the hidden sentence.

For comparison, we benchmark our approach against several state-of-the-art memory-efficient methods, including PolarQuant [28], SnapKV [38], PyramidKV [12], and KIVI [41]. Each method is tested under a memory compression ratio of 0.25, meaning that only 25% of the full KV cache is utilized.

The results, illustrated in Fig. 4, reveal that quantization methods with theoretical guarantees, such as PolarQuant and TurboQuant, outperform token-level compression techniques like SnapKV and PyramidKV, as well as scalar quantization approaches like KIVI, which lack formal theoretical guarantees. Notably, TurboQuant achieves identical performance to the full-precision model, even at 4× compression, making it a robust solution for long-context processing.

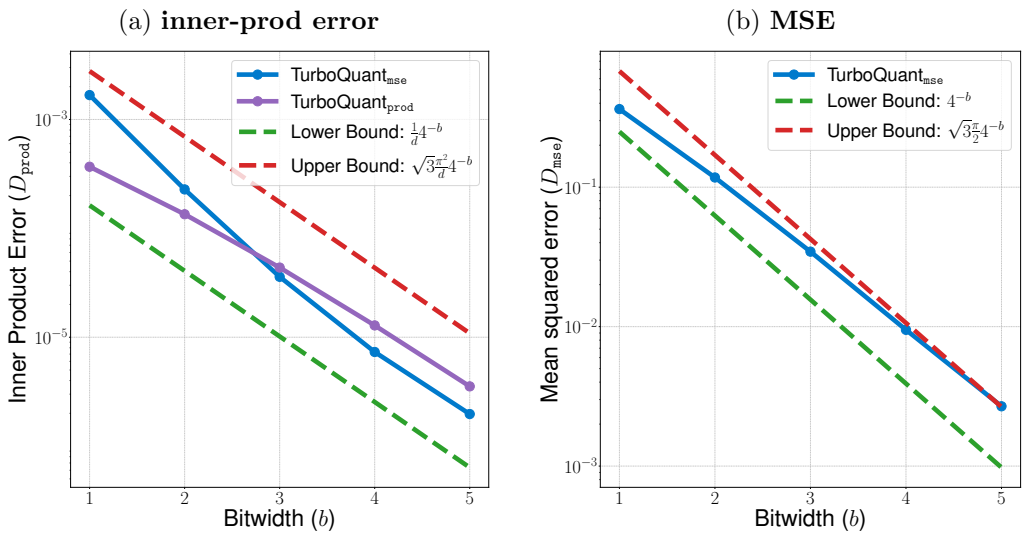


Figure 3: Comparison of inner-product error and MSE against theoretical bounds across different bit ratios.

4.3 End-to-end Generation on LongBench

We experiment with various KV cache compression algorithms on the LongBench dataset [10], which encompasses a broad range of long-text scenarios, including single- and multi-document question-answering, summarization, few-shot learning, synthetic tasks, and code completion. To ensure a balanced evaluation across different context lengths, we employ **LongBench-E**, a subset designed with a more uniform length distribution. This enables a fair assessment of each model's performance across varying context sizes, making it a more reliable benchmark for evaluating compression techniques.

We compare TurboQuant against the leading baseline methods introduced in Section 4.2, using both `Llama-3.1-8B-Instruct` and `Ministral-7B-Instruct`. Unlike existing approaches such as **KIVI** and **PolarQuant**, which leave generated tokens unquantized, our method applies quantization even during the streaming generation process.

As shown in Table 1, our approach outperforms other methods for both `Llama-3.1-8B-Instruct` and `Ministral-7B-Instruct`, achieving significantly higher average scores. We evaluate our method using **2.5-bit** and **3.5-bit** quantization during text generation. These non-integer bit precisions result from our strategy of splitting channels into outlier and non-outlier sets, and applying two independent instances of TurboQuant to each, allocating higher bit precision to outliers. This outlier treatment strategy is consistent with prior work [63, 51] . For example, in our 2.5-bit setup, 32 outlier channels are quantized at 3 bits, while the remaining 96 channels use 2 bits, leading to an effective bit precision of $(32 \times 3 + 96 \times 2)/128 = 2.5$. For 3.5-bit quantization, a different ratio of outliers and regular channels leads to a higher effective bit precision. Despite using fewer bits than competing techniques, TurboQuant maintains performance comparable to unquantized models. Remarkably, we achieve this while compressing quantized vectors by at least a factor of $4.5\times$.

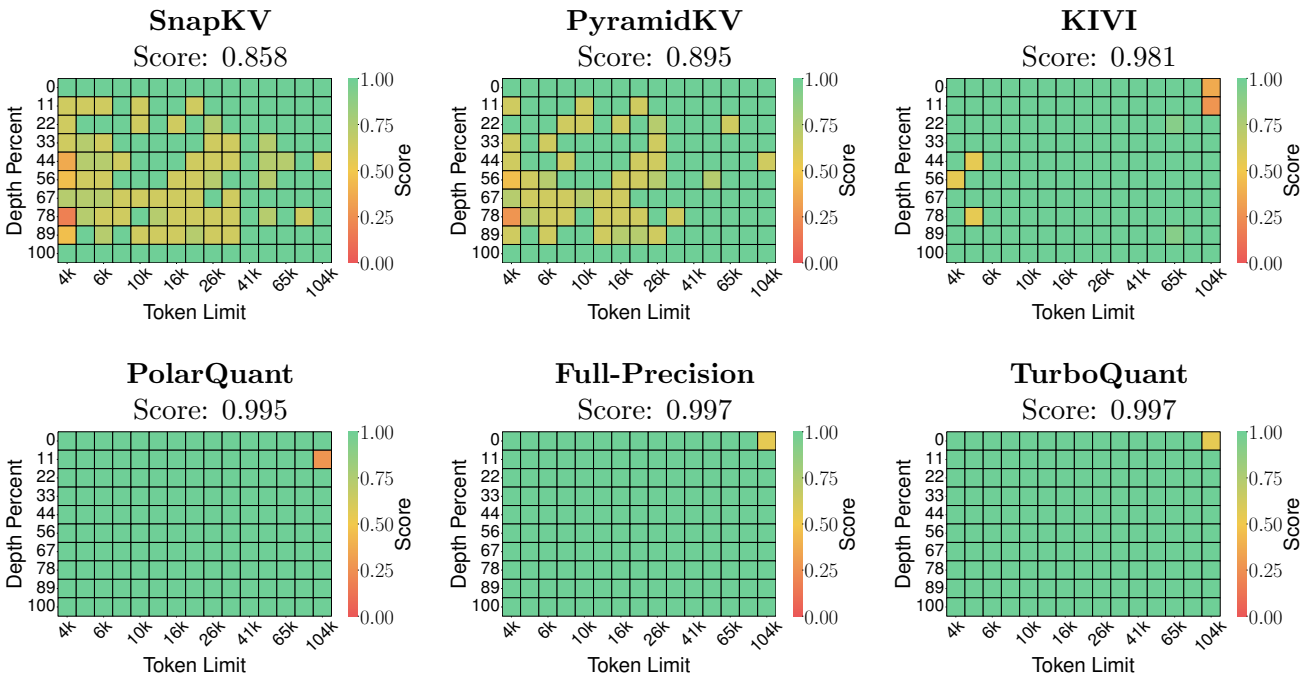


Figure 4: Evaluation of `Llama-3.1-8B-Instruct` on the "Needle-In-A-Haystack" test, where a model must retrieve a hidden sentence from long-context sequences. While some methods struggle with recall, TurboQuant, despite being more than 4× quantized, achieves the same exact performance as the uncompressed baseline.

4.4 Near Neighbour Search Experiments

In this section, we establish the strength of our proposed method, even in the context of near-neighbor search. We conduct our experiments using the DBpedia [53] Entities dataset, which has been encoded into 1536-dimensional[1] and 3072-dimensional [2] spaces using OpenAI3 embeddings. Additionally, we evaluate performance on a lower-dimensional dataset, utilizing the standard GloVe [45] embeddings. To construct our experimental setup, we randomly sample 100,000 data points from the dataset, denoted as training set, which serves as our primary training and evaluation set. Furthermore, we extract 1,000 distinct entries, denoted as query set, to be used as query points for datasets that do not explicitly provide a query set. For the GloVe dataset, we use a pre-existing query set consisting of 10,000 points.

We compare our method, TurboQuant, against two baseline quantization approaches: Product Quantization (PQ) and RabitQ [22]. To ensure a fair comparison, we quantize the dataset training set using all three methods and evaluate their performance based on recall ratio at top-k, denoted as 1@k. Specifically, this metric assesses how often the true top inner product result is captured within the top-k approximated results returned by each algorithm.

Product Quantization (PQ) relies on the k-means algorithm to construct codebooks, which require separate storage. As the number of bits increases, the size of the codebook grows exponen-

[1] https://huggingface.co/datasets/Qdrant/dbpedia-entities-openai3-text-embedding-3-large-1536-1M
[2] https://huggingface.co/datasets/Qdrant/dbpedia-entities-openai3-text-embedding-3-large-3072-1M

Method	KV Size	SingleQA	MultiQA	Summarization	Few shot	Synthetic	Code	Average
				`Llama-3.1-8B-Instruct`				
Full Cache	16	45.29	45.16	26.55	68.38	59.54	46.28	50.06
KIVI	3	43.38	37.99	27.16	68.38	59.50	44.68	48.50
KIVI	5	45.04	45.70	26.47	68.57	59.55	46.41	50.16
PolarQuant	3.9	45.18	44.48	26.23	68.25	60.07	45.24	49.78
TurboQuant (ours)	2.5	44.16	44.96	24.80	68.01	59.65	45.76	49.44
TurboQuant (ours)	3.5	45.01	45.31	26.00	68.63	59.95	46.17	50.06
				`Ministral-7B-Instruct`				
Full Cache	16	47.53	49.06	26.09	66.83	53.50	47.90	49.89
TurboQuant (ours)	2.5	48.38	49.22	24.91	66.69	53.17	46.83	49.62

Table 1: LongBench-V1 [10] results of various KV cache compression methods on `Llama-3.1-8B-Instruct`.

Approach	d=200	d=1536	d=3072
Product Quantization	37.04	239.75	494.42
RabitQ	597.25	2267.59	3957.19
TurboQuant	0.0007	0.0013	0.0021

Table 2: Quantization time (in seconds) for different approaches across various dimensions using 4-bit quantization.

tially, leading to additional storage overhead. In our experiments, we carefully tuned the parameters to match the bit allocation of other methods. The most efficient implementation, designed for rapid querying, employs AVX2 In-Register Lookup Tables (LUTs). Specifically, it uses LUT16 with ($l = 16$) codewords. However, we observed substantial quality degradation at this configuration. To achieve a balance between speed and accuracy, we opted for a version of PQ that uses LUT256, which contains 256 codewords. For 2-bit quantization, it groups 4 coordinates per lookup, while for 4-bit quantization, it groups 2 coordinates per lookup. Notably, since we use the same dataset for both training and evaluation, PQ benefits from an inherent advantage in this setup.

RabitQ. Unlike PQ, RabitQ lacks a fully vectorized implementation, making it impossible to leverage GPU acceleration. As a result, it runs significantly slower on CPU. Additionally, the method incurs extra computational overheads that we do not explicitly account for in the bit ratio comparisons. While RabitQ claims a certain bit ratio, in practice, it utilizes more bits than reported due to these inefficiencies.

Despite the advantages granted to the baseline methods, TurboQuant consistently outperforms both Product Quantization and RabitQ in terms of recall ratio across all experiments. This demonstrates the robustness and efficiency of our approach, making it a compelling alternative for high-dimensional quantization-based search tasks.

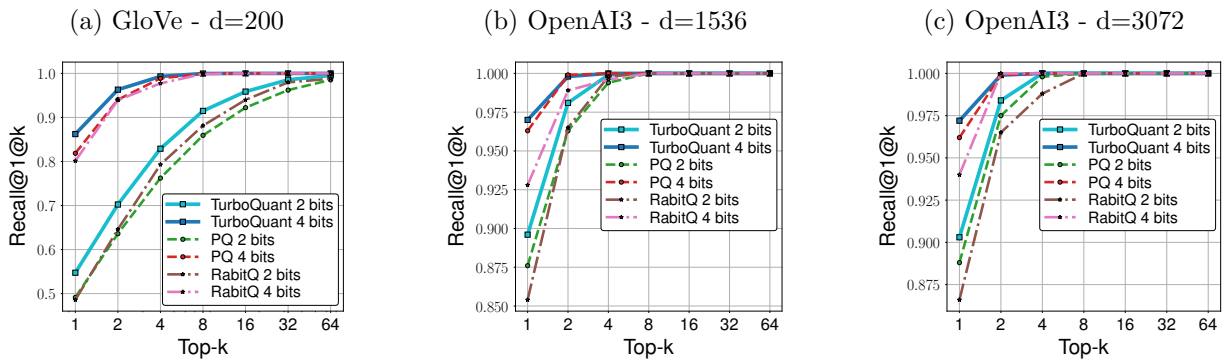


Figure 5: Recall comparison on different datasets with different embedding dimensions.

References

[1] Elastic search., 2025. https://www.elastic.co/enterprise-search/vector-search.

[2] Qdrant vectore search., 2025. https://qdrant.tech/.

[3] Pgvector search., 2025. https://github.com/pgvector/pgvector/.

[4] Pinecone vectore database., 2025. https://www.pinecone.io/.

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

[6] Ainslie, J., Lee-Thorp, J., de Jong, M., Zemlyanskiy, Y., Lebron, F., and Sanghai, S. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 4895–4901, 2023.

[7] Anthropic. Claude, 2024. https://www.anthropic.com/news/claude-3-family.

[8] Ashkboos, S., Mohtashami, A., Croci, M. L., Li, B., Cameron, P., Jaggi, M., Alistarh, D., Hoefler, T., and Hensman, J. Quarot: Outlier-free 4-bit inference in rotated llms. *arXiv preprint arXiv:2404.00456*, 2024.

[9] Babenko, A. and Lempitsky, V. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 931–938, 2014.

[10] Bai, Y., Lv, X., Zhang, J., Lyu, H., Tang, J., Huang, Z., Du, Z., Liu, X., Zeng, A., Hou, L., Dong, Y., Tang, J., and Li, J. Longbench: A bilingual, multitask benchmark for long context understanding. *arXiv preprint arXiv:2308.14508*, 2023.

[11] Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

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

[13] Chee, J., Cai, Y., Kuleshov, V., and De Sa, C. M. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36:4396–4429, 2023.

[14] Cover, T. M. *Elements of information theory*. John Wiley & Sons, 1999.

[15] Dai, D., Deng, C., Zhao, C., Xu, R., Gao, H., Chen, D., Li, J., Zeng, W., Yu, X., Wu, Y., et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

[16] Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35: 30318–30332, 2022.

[17] Dong, S., Cheng, W., Qin, J., and Wang, W. Qaq: Quality adaptive quantization for llm kv cache. *arXiv preprint arXiv:2403.04643*, 2024.

[18] Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[19] Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., Mody, A., Truitt, S., and Larson, J. From local to global: A graph rag approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

[20] Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[21] Fu, Y., Panda, R., Niu, X., Yue, X., Hajishirzi, H., Kim, Y., and Peng, H. Data engineering for scaling language models to 128k context. *arXiv preprint arXiv:2402.10171*, 2024. URL https://github.com/FranxYao/Long-Context-Data-Engineering.

[22] Gao, J., Gou, Y., Xu, Y., Yang, Y., Long, C., and Wong, R. C.-W. Practical and asymptotically optimal quantization of high-dimensional vectors in euclidean space for approximate nearest neighbor search. *arXiv preprint arXiv:2409.09913*, 2024.

[23] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, H., and Wang, H. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2, 2023.

[24] Ge, T., He, K., Ke, Q., and Sun, J. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2946–2953, 2013.

[25] Gersho, A. Asymptotically optimal block quantization. *IEEE Transactions on information theory*, 25(4):373–380, 1979.

[26] Gersho, A. On the structure of vector quantizers. *IEEE Transactions on Information Theory*, 28(2):157–166, 1982.

[27] Guo, R., Sun, P., Lindgren, E., Geng, Q., Simcha, D., Chern, F., and Kumar, S. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pp. 3887–3896. PMLR, 2020.

[28] Han, I., Kacham, P., Karbasi, A., Mirrokni, V., and Zandieh, A. Polarquant: Quantizing kv caches with polar transformation. *arXiv preprint arXiv:2502.02617*, 2025.

[29] Han, I., Kapralov, M., Kochetkova, E., Sheth, K., and Zandieh, A. Balancekv: Kv cache compression through discrepancy theory. *arXiv preprint arXiv:2502.07861*, 2025.

[30] Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *arXiv preprint arXiv:2401.18079*, 2024.

[31] Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[32] Kamradt, G. Needle in a haystack - pressure testing llms., 2023. https://github.com/gkamradt/LLMTest_NeedleInAHaystack.

[33] Kang, H., Zhang, Q., Kundu, S., Jeong, G., Liu, Z., Krishna, T., and Zhao, T. Gear: An efficient kv cache compression recipefor near-lossless generative inference of llm. *arXiv preprint arXiv:2403.05527*, 2024.

[34] Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[35] Khattab, O. and Zaharia, M. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pp. 39–48, 2020.

[36] Kim, J., Park, J., Cho, J., and Papailiopoulos, D. Lexico: Extreme kv cache compression via sparse coding over universal dictionaries. *arXiv preprint arXiv:2412.08890*, 2024.

[37] Kim, S., Hooper, C., Gholami, A., Dong, Z., Li, X., Shen, S., Mahoney, M. W., and Keutzer, K. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

[38] Li, Y., Huang, Y., Yang, B., Venkitesh, B., Locatelli, A., Ye, H., Cai, T., Lewis, P., and Chen, D. Snapkv: Llm knows what you are looking for before generation. *arXiv preprint arXiv:2404.14469*, 2024.

[39] Lin, J., Tang, J., Tang, H., Yang, S., Chen, W.-M., Wang, W.-C., Xiao, G., Dang, X., Gan, C., and Han, S. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[40] Liu, Z., Desai, A., Liao, F., Wang, W., Xie, V., Xu, Z., Kyrillidis, A., and Shrivastava, A. Scissorhands: Exploiting the persistence of importance hypothesis for llm kv cache compression at test time. *Advances in Neural Information Processing Systems*, 36, 2024.

[41] Liu, Z., Yuan, J., Jin, H., Zhong, S., Xu, Z., Braverman, V., Chen, B., and Hu, X. Kivi: A tuning-free asymmetric 2bit quantization for kv cache. *arXiv preprint arXiv:2402.02750*, 2024.

[42] Lloyd, S. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

[43] Max, J. Quantizing for minimum distortion. *IRE Transactions on Information Theory*, 6(1): 7–12, 1960.

[44] Panter, P. and Dite, W. Quantization distortion in pulse-count modulation with nonuniform spacing of levels. *Proceedings of the IRE*, 39(1):44–48, 1951.

[45] Pennington, J., Socher, R., and Manning, C. GloVe: Global vectors for word representation. In Moschitti, A., Pang, B., and Daelemans, W. (eds.), *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://aclanthology.org/D14-1162/.

[46] Santhanam, K., Khattab, O., Saad-Falcon, J., Potts, C., and Zaharia, M. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*, 2021.

[47] Shah, J., Bikshandi, G., Zhang, Y., Thakkar, V., Ramani, P., and Dao, T. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 2024.

[48] Shannon, C. E. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

[49] Shannon, C. E. et al. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, 4(142-163):1, 1959.

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

[51] Su, Z., Chen, Z., Shen, W., Wei, H., Li, L., Yu, H., and Yuan, K. Rotatekv: Accurate and robust 2-bit kv cache quantization for llms via outlier-aware adaptive rotations, 2025. URL https://arxiv.org/abs/2501.16383.

[52] Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[53] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

[54] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. *NeurIPS*, 2017.

[55] Vershynin, R. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[56] Wang, J., Zhang, T., Sebe, N., Shen, H. T., et al. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):769–790, 2017.

[57] Xiao, G., Lin, J., Seznec, M., Wu, H., Demouth, J., and Han, S. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pp. 38087–38099. PMLR, 2023.

[58] Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

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

[60] Yue, Y., Yuan, Z., Duanmu, H., Zhou, S., Wu, J., and Nie, L. Wkvquant: Quantizing weight and key/value cache for large language models gains more. *arXiv preprint arXiv:2402.12065*, 2024.

[61] Zador, P. L. *Development and evaluation of procedures for quantizing multivariate distributions*. Stanford University, 1964.

[62] Zandieh, A., Daliri, M., and Han, I. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead, 2024. URL https://arxiv.org/abs/2406.03482.

[63] Zandieh, A., Daliri, M., and Han, I. Qjl: 1-bit quantized jl transform for kv cache quantization with zero overhead. *arXiv preprint arXiv:2406.03482*, 2024.

[64] Zandieh, A., Han, I., Mirrokni, V., and Karbasi, A. Subgen: Token generation in sublinear time and memory. *arXiv preprint arXiv:2402.06082*, 2024.

[65] Zhang, T., Yi, J., Xu, Z., and Shrivastava, A. Kv cache is 1 bit per channel: Efficient large language model inference with coupled quantization. *arXiv preprint arXiv:2405.03917*, 2024.

[66] Zhang, Z., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C., et al. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *Advances in Neural Information Processing Systems*, 36, 2024.