Quantization Meets Projection: A Happy Marriage for Approximate k-Nearest Neighbor Search [Scalable Data Science]

Mingyu Yang
The Hong Kong University of Science and Technology (Guangzhou)
myang250@connect.hkust-gz.edu.cn

Liuchang Jing
The Hong Kong University of Science and Technology (Guangzhou)
ljing248@connect.hkust-gz.edu.cn

Wentao Li
University of Leicester
wl226@leicester.ac.uk

Wei Wang
The Hong Kong University of Science and Technology (Guangzhou)
weiwcs@ust.hk

ABSTRACT

Approximate k-nearest neighbor (AKNN) search is a fundamental problem with wide applications. To reduce memory and accelerate search, vector quantization is widely adopted. However, existing quantization methods either rely on codebooks—whose sizes are flexible but whose query speed is limited by costly table lookups—or adopt dimension-wise quantization, which maps each vector dimension to a small quantized code for fast search. The latter, however, suffers from a fixed compression ratio because the quantized code length is inherently tied to the original dimensionality. To overcome these limitations, we propose MRQ, a new approach that integrates projection with quantization. The key insight is that, after projection, high-dimensional vectors tend to concentrate most of their information in the leading dimensions. MRQ exploits this property by quantizing only the information-dense projected subspace—whose size is fully user-tunable—thereby decoupling the quantized code length from the original dimensionality. The remaining tail dimensions are captured using lightweight statistical summaries. By doing so, MRQ preserves the high query throughput of dimension-wise quantization while achieving arbitrary compression ratios enabled by the projection step. Extensive experiments show that MRQ substantially outperforms the state-of-the-art method, achieving up to 3× faster search with only one-third the quantization bits for comparable accuracy

PVLDB Reference Format:
Mingyu Yang, Liuchang Jing, Wentao Li, and Wei Wang. Quantization Meets Projection: A Happy Marriage for Approximate k-Nearest Neighbor Search [Scalable Data Science]. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/mingyu-hkustgz/RESQ.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

1 INTRODUCTION

The k-Nearest Neighbor (KNN) search for vectors is critical to many applications, including recommendation systems [56], information retrieval [47], and Retrieval-Augmented Generation for Large Language Models [45]. However, the curse of dimensionality [39] renders exact KNN search computationally expensive in high-dimensional space. This has led to growing interest in approximate KNN (AKNN) search [11, 13, 16, 22–24, 26, 29, 33, 36, 38, 41, 53, 57, 61, 67, 69, 71, 73, 75], which trades off a small amount of accuracy for substantially improved efficiency in large-scale settings.

Quantization. The vectors used in AKNN search are typically stored in floating-point format, making large-scale in-memory storage expensive. To mitigate this, **quantization** has become a widely adopted technique for reducing storage overhead [34, 52]. Prominent quantization methods include Binary Quantization (BQ) [27, 32, 44, 46, 55], Scalar Quantization (SQ) [2, 25, 74], Product Quantization (PQ) [2–4, 7, 14, 28, 43, 51, 62, 68], and Additive Quantization (AQ) [7, 8, 70]. These methods can be broadly grouped into two families based on their encoding strategies. **(1) Codebook-Based Quantization.** Methods such as PQ, AQ, and their variants [8, 62, 70] learn a set of representative vectors (codewords), collectively referred to as a *codebook*. Each original vector is then approximately represented as a composition of codewords. These methods offer flexible compression ratios through configurable codebook sizes. Yet, they rely on large lookup tables and codeword aggregation for distance computation during AKNN search, which can become a computational bottleneck without specialized optimizations [3]. **(2) Dimension-Wise Quantization.** Methods such as BQ and SQ quantize each vector dimension independently. For example, BQ converts each dimension to a single bit, while SQ transforms the numeric format of each dimension (e.g., from a 32-bit float to an 8-bit or 4-bit integer). These methods achieve very high query throughput as they avoid codeword lookups required by codebook-based approaches.

Motivations. As confirmed by recent studies, dimension-wise quantization methods significantly outperform codebook-based approaches in both efficiency and accuracy [1, 2, 25, 27]. Owing to their strong empirical performance, they have been widely deployed in large-scale data science applications and systems [20, 42, 73]. Their speed advantage primarily arises from their Single Instruction, Multiple Data (SIMD)–friendly design—leveraging low-level

instructions such as POPCNT and AVX-int8—as well as beneficial geometric properties like unbiased inner-product estimation.

However, a key limitation of dimension-wise quantization is their restricted compression ratio. Because these methods quantize each original dimension independently, the length of the quantized code remains tied to the original vector dimensionality. For example, the state-of-the-art dimension-wise method RabitQ [27] requires the length of the quantized code to exactly match the dimensionality of the original vector. This inflexibility introduces two major issues: (1) users cannot achieve higher compression ratios, which is problematic for large-scale datasets containing billions of vectors that may still exceed main-memory capacity even after quantization; and (2) users cannot freely adjust the bit-width, causing the quantized code length to misalign with SIMD widths and thereby requiring extra SIMD instructions, resulting in suboptimal performance.

Our Solution. To summarize, existing quantization methods either suffer from slow query performance or impose a fixed compression ratio as the code length is tied to the original dimensionality. This raises a natural question: *Can we retain high-performance quantization while avoiding the constraints of a fixed compression ratio?* We argue that a principled integration of projection and quantization provides a compelling answer. Our key insight is grounded in an empirical analysis of modern high-dimensional vector embeddings: after applying an information-aware projection such as PCA, the per-dimension variance exhibits a pronounced long-tailed distribution (see Fig. 2). This reveals that most information is concentrated within a low-dimensional *subspace*, motivating differentiated treatment of projected dimensions during quantization.

Building on this observation, we propose our method MRQ (Minimized Residual Quantization). MRQ first projects the original vector into a predefined number of dimensions, forming a **projected vector**, while the remaining dimensions constitute the **residual vector**. For the projected vector—which captures most of the variance—we apply RabitQ [27]-style quantization, but extended to support *arbitrary* bit lengths. For the residual vector, we introduce a key innovation: instead of storing it directly, we model its distance contribution using lightweight statistics (e.g., variance). This enables accurate distance estimation with bounded error, without incurring the cost of storing the residual explicitly. By combining the quantized projected vector with the residual component, MRQ performs multi-stage distance computation to ensure efficient and accurate AKNN search. The process starts with fast, lightweight estimation and progressively refines the result using more accurate, but costlier, computations. Fig. 1 illustrates this process. Crucially, MRQ supports an arbitrary compression ratio for quantization—unlike prior work—while still ensuring accuracy through the use of error bounds for re-ranking.

Contributions. Our main contributions are summarized as follows: *A Flexible Quantization Mechanism.* To address the limitation of the dimension-wise quantization methods (especially the state-of-the-art method bitQ [27]), which enforces a fixed compression ratio, we are the first to systematically analyze and exploit the long-tailed variance distribution of high-dimensional data after projection rotation. This insight underpins a novel AKNN search method that seamlessly integrates projection and quantization.

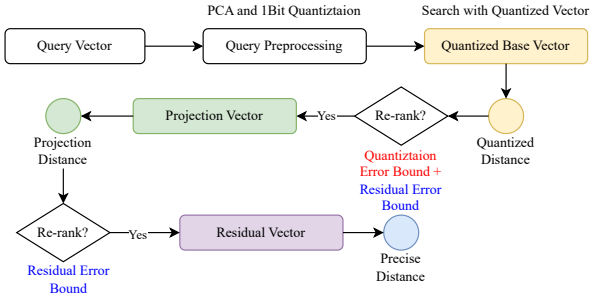


Figure 1: Workflow of MRQ. After projection, both base and query vectors are split into projected and residual components. MRQ performs distance computation in three stages: (1) The projected vectors are quantized with arbitrary bit lengths, enabling fast distance estimation with error bounds for re-ranking. (2) For higher accuracy, the original projected vectors replace the quantized ones, yielding more precise distances while retaining the error bounds. (3) If exact computation is needed, the full vector (projected + residual) is used.

A Novel Distance Computation Framework. We design a multi-stage distance computation framework that asymmetrically handles projected and residual components (see Fig. 1). It combines distance estimation and error bounds derived from both components for efficient and accurate AKNN search.

Efficient Implementation. We incorporate our multi-stage distance computation into a highly optimized IVF-based index. This design leverages custom memory layouts and approximation strategies to improve cache locality and dramatically reduce index construction time, ensuring scalability to large-scale datasets.

Extensive Experimental Evaluation. We conduct comprehensive evaluations on real-world datasets, benchmarking against strong baselines including the graph-based method HNSW and the quantization-based method RabitQ. Our method consistently outperforms existing approaches, achieving up to a 3× improvement in search efficiency while preserving comparable accuracy.

Due to space limitations, some discussions and experiments are omitted here and can be found in our technical report [65].

2 PRELIMINARY AND RELATED WORK

Section 2.1 introduces the AKNN search problem and reviews existing AKNN methods. Section 2.2 presents vector quantization techniques for reducing space overhead. Section 2.3 describes re-ranking strategies used to improve search accuracy.

2.1 AKNN Search and Methods

Given a database S of n vectors/points in D-dimensional Euclidean space $\mathbb{R}^D$ and a query point $q \in \mathbb{R}^D$, the problem of KNN search is to retrieve the top k points in S closest to q under the Euclidean distance. Due to the prohibitive cost of exact KNN search in high-dimensional spaces [39], recent efforts have focused on approximate KNN (AKNN) methods. A common metric for evaluating AKNN accuracy is Recall@K, defined as the proportion of true nearest neighbors successfully retrieved: Recall@$k = \frac{|T \cap G|}{|G|}$ where T is the set returned by the AKNN method, and G is the set of ground-truth KNN for the query q.

The goal of AKNN methods is to achieve high recall with minimal computational cost. Recently, a variety of AKNN methods have

been proposed, which can be broadly categorized into four classes: (1) hashing-based methods [16, 18, 24, 26, 29, 30, 37, 38, 57, 58, 71], (2) inverted-file (IVF)-based methods [3, 7, 9, 27, 28, 32, 35, 43, 62, 70], (3) tree-based methods [6, 10, 15, 19], and (4) graph-based methods [6, 6, 13, 22, 23, 36, 40, 48–50, 54, 59, 61, 63]. This paper focuses on IVF- and graph-based methods, which build specialized indexes to accelerate the search and achieve state-of-the-art performance.

IVF-based methods first partition points in database S using clustering algorithms such as k-means to obtain cluster centroids. Each data point is then assigned to its nearest centroid. During query, we identify a small number of centroids closest to the query and search only the associated clusters to retrieve the top-k results.

Graph-based methods construct an index by modeling the high-dimensional data points as nodes in a graph, where edges connect each node to its nearby neighbors. This results in a navigable small-world graph structure. At query time, the search starts from an entry node and iteratively traverses toward nodes that are progressively closer to the query.

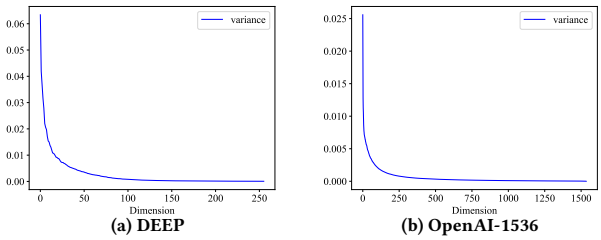
To reduce space cost, these methods often adopt quantization techniques, as storing original float-type vectors is memory-intensive. Also, re-ranking strategies are employed to improve search accuracy, since distance computations based on quantized vectors are inherently approximate.

2.2 Quantization Techniques

Quantization compresses high-dimensional vectors into compact codes to reduce storage cost and improve search efficiency. Based on their encoding strategy, existing methods can be broadly categorized into *codebook-based* and *dimension-wise* approaches.

Codebook-Based Quantization. These methods approximate vectors using a finite set of representative centroids (i.e., codewords). For example, PQ [3, 35, 62] decomposes a high-dimensional space into multiple low-dimensional subspaces and performs clustering within each subspace. Variants such as Optimized PQ (OPQ) [28, 43] and Additive Quantization (AQ) [7] further reduce quantization error through orthogonal rotations or additive code composition. These methods offer flexible compression ratios (controlled by the number of subspaces and codebook size). However, they suffer from a fundamental efficiency bottleneck: distance computation involves table lookups and floating-point additions, which are difficult to fully accelerate using SIMD instructions compared with bitwise operations. Although hardware-aware optimizations (e.g., 4-bit PQ with AVX-512 [3]) improve performance, they typically impose strict constraints on codebook sizes or incur accuracy degradation.

Dimension-Wise Quantization. These methods quantize each dimension of a vector independently, mapping continuous values to discrete integers or bits. For example, Scalar Quantization (SQ) [1, 2, 21, 25, 74] typically maps `float32` values to `int8` and leverages low-precision AVX-512 instructions for efficient computation. Binary Quantization (BQ) [12, 27] further pushes this idea by mapping each dimension to a single bit, enabling ultra-fast distance computation using bitwise operations (e.g., `POPCNT`). The state-of-the-art RabitQ [25, 27] improves accuracy by introducing an unbiased estimator and error bounds for efficient re-ranking. However, all dimension-wise encoding methods suffer from a *fixed-ratio constraint*: the code length is rigidly tied to the original dimensionality (e.g., a 960-dimensional vector produces a 960-bit binary code). This inflexibility prevents users from tuning the compression ratio to meet memory budgets or from aligning code lengths with hardware register widths for optimal SIMD performance.



(a) DEEP **(b) OpenAI-1536**

Figure 2: Variance Distribution of Vectors After PCA

2.3 Re-Ranking

One limitation of quantization techniques is the reduction in search accuracy (i.e., recall) due to approximate distance computation. To mitigate this issue, a common strategy is to apply re-ranking using exact distances [25–27, 40, 60]. Specifically, the method first performs AKNN search using approximate distances and retains a larger candidate set, such as the top-R results (where $R > k$). Then, it re-computes the exact distances for all candidates and selects the final top-K nearest neighbors based on these corrected values.

Error Bounds. Re-ranking introduces additional overhead because it requires many exact distance computations. To alleviate this cost, recent AKNN methods such as RabitQ [27] incorporate error bounds to prune unlikely candidates early. By exploiting statistical properties of the quantization process, RabitQ can determine whether the true distance of a candidate is guaranteed to exceed that of the current k-th nearest neighbor. If so, the candidate can be safely discarded without performing an exact distance computation. However, as discussed in the Introduction, this optimization is inherently tied to the constraint that the code length must equal the vector dimensionality, limiting its applicability. Our work, MRQ, aims to retain the benefits of error-bounded re-ranking while removing this restrictive constraint.

3 PROBLEM ANALYSIS AND OBSERVATION

This section reiterates the limitations of existing methods and then presents our key observation that motivates the design of MRQ.

3.1 Problem Analysis

Existing dimension-wise quantization methods cannot achieve flexible compression ratios because they must quantize *every* dimension of the original vectors. A straightforward idea to improve the compression ratio is to truncate or randomly drop some dimensions of the original vectors. For example, consider an original vector in `float32` format: under binary quantization, achieving a 128× compression ratio would require discarding roughly 75% of its dimensions. However, such naive truncation introduces severe bias in distance estimation, which in turn leads to significant degradation in AKNN search accuracy. This raises two natural questions: (1) can we minimize the information loss caused by discarding residual dimensions; and (2) can we still extract useful information from the discarded residual dimensions to further improve accuracy?

3.2 Observation

To benefit from the significant space reduction offered by quantization methods while overcoming the limitation of a fixed compression ratio, we present the following observation that motivate the development of our new approach.

Variance Distribution Analysis. We begin our analysis with vector data held in vector databases. These data typically comprise high-dimensional embeddings—often exceeding 1,000 dimensions—derived from neural networks for audio, image, and text representations. After applying PCA, we observe that the **variance distribution of the vector data exhibits a long-tailed pattern.**

Example 1. *We present the observation in Fig. 2. After applying PCA to the text embeddings generated by the OpenAI-1536 model, we observe that the first 512 dimensions capture nearly 90% of the total variance, while the remaining 1000 dimensions account for only 10%. A similar trend is observed in other data: for image embeddings from the DEEP dataset, only 128 PCA dimensions are sufficient to retain 90% of all the variance.*

From an information-theoretic perspective, higher variance in a dimension indicates greater information content. Consequently, only a small number of dimensions are required to preserve most of the information in the original vector. This suggests that treating all dimensions equally during quantization—such as by randomly dropping or truncating them—is suboptimal compared to using PCA, which more effectively reduces information loss. Recent studies [17, 66] have analyzed PCA also minimizes the variance of the residual dimension, and lower variance in the residual dimensions leads to a smaller mean squared error (MSE) when projecting to a lower-dimensional space. Therefore, given their limited information content and low contribution to reconstruction error, these residual dimensions can be safely discarded during quantization. Beyond the quantized portion of a vector, we observe that the residual part still carries useful information for improving AKNN search accuracy. In particular, the norm of the residual can be leveraged to obtain a more accurate distance estimate, while the variance of the residual provides a tight bound for distance approximation. In summary, focusing on the high-variance components after PCA not only (1) offers a flexible way to discard dimensions while minimizing information loss, but also (2) enables the use of residual statistics for further refinement.

4 OUR QUANTIZATION METHOD

Based on the above observation, we propose a novel quantization method, MRQ (Minimized Residual Quantization). In Section 4.1, we present our vector decomposition strategy, which enables user-controllable compression ratios. In Section 4.2, we introduce a multi-stage distance computation scheme to ensure high AKNN accuracy.

4.1 Vector Decomposition

The observation of variance distribution reveals that after applying PCA to vectors in a dataset, only a small number of dimensions retain most of the information. This motivates us to treat the high-variance dimensions differently by *applying quantization only to the projected vectors in this informative subspace.* The remaining dimensions carry little information and can be discarded or handled separately if needed. Fig. 3 describes this insight. Importantly, the number of retained dimensions is fully user-controlled, enabling a flexible compression ratio.

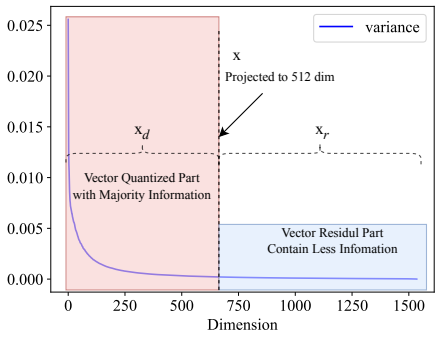


Figure 3: The Example of Vector Split and Quantization

Specifically, consider a vector dataset S consisting of N points in a D-dimensional Euclidean space $\mathbb{R}^D$. Each vector $\mathbf{x} \in S$ can be projected into a lower-dimensional space of dimension d (where $d < D$) using an orthogonal matrix $\mathbf{R}$, i.e., $\mathbf{x}_d = \mathbf{R}\mathbf{x}$. The exact squared Euclidean distance dis between a database vector $\mathbf{x}$ and a query vector $\mathbf{q}$ can then be expressed as:

$$dis = ||\mathbf{x} - \mathbf{q}||^2 = ||\mathbf{x}_d - \mathbf{q}_d||^2 + ||\mathbf{x}_r - \mathbf{q}_r||^2 \tag{1}$$

where $\mathbf{x}_r$ is the residual part ($r + d = D$). We then apply a quantization method (with state-of-the-art method RabitQ as the example) to the projected vector $\mathbf{x}_d$. Specifically, similar to RabitQ, we first normalize and center the vectors after projection. Let $\mathbf{c}$ denote the centroid of the projected data $\mathbf{x}_d$. The normalized projected vector is defined as $\mathbf{x}_b := \frac{\mathbf{x}_d - \mathbf{c}}{\|\mathbf{x}_d - \mathbf{c}\|}$, and the query vector is similarly normalized as $\mathbf{q}_b := \frac{\mathbf{q}_d - \mathbf{c}}{\|\mathbf{q}_d - \mathbf{c}\|}$. After normalization, the projected distance $\|\mathbf{x}_d - \mathbf{q}_d\|^2$ can be expressed in terms of the inner product and vector norms, as formally defined below.

$$\begin{aligned} ||\mathbf{x}_d - \mathbf{q}_d||^2 = {} & ||\mathbf{x}_d - \mathbf{c}||^2 + ||\mathbf{q}_d - \mathbf{c}||^2 \\ & - 2 \cdot ||\mathbf{x}_d - \mathbf{c}|| \cdot ||\mathbf{q}_d - \mathbf{c}|| \cdot \langle \mathbf{x}_b, \mathbf{q}_b \rangle \end{aligned} \tag{2}$$

Note that the inner-product term $\langle \mathbf{x}_d, \mathbf{q}_d \rangle$ can be approximated by quantizing $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ using methods such as RabitQ, yielding a d-bit binary representation. By precomputing the norms and other components in Equation 2, the distance within the projected subspace can be efficiently calculated. For the residual component $||\mathbf{x}_r - \mathbf{q}_r||^2$, it can be decomposed as $||\mathbf{x}_r||^2 + ||\mathbf{q}_r||^2 - 2 \cdot \langle \mathbf{x}_r, \mathbf{q}_r \rangle$. Incorporating this into Equation 2 yields the final formulation for computing the overall distance.

$$\begin{aligned} ||\mathbf{x} - \mathbf{q}||^2 = {} & ||\mathbf{x}_d - \mathbf{c}||^2 + ||\mathbf{q}_d - \mathbf{c}||^2 + ||\mathbf{x}_r||^2 + ||\mathbf{q}_r||^2 \\ & - 2 \cdot ||\mathbf{x}_d - \mathbf{c}|| \cdot ||\mathbf{q}_d - \mathbf{c}|| \cdot \langle \mathbf{x}_b, \mathbf{q}_b \rangle - 2 \cdot \langle \mathbf{x}_r, \mathbf{q}_r \rangle \end{aligned} \tag{3}$$

Example 4.1. Fig. 4 illustrates an example of Equation 3. We first apply PCA to transform the original vectors $\mathbf{x}$ and $\mathbf{q}$, yielding their projected components $\mathbf{x}_d$, $\mathbf{q}_d$ and residual components $\mathbf{x}_r$, $\mathbf{q}_r$. The projected components are then normalized and quantized to compute the inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$, while the residual components can be precomputed to account for the residual distance term.

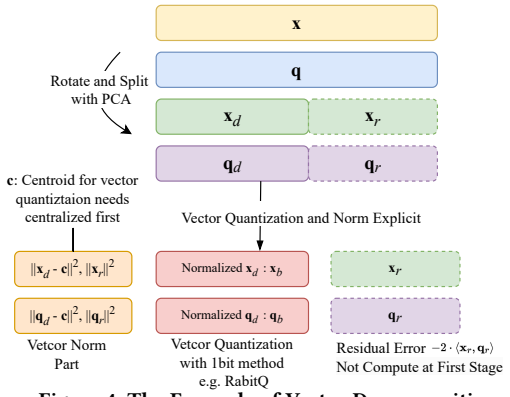


Figure 4: The Example of Vector Decomposition

Remark. Equation 3 shows that we only need to quantize the projected vectors $\mathbf{x}_d$ and $\mathbf{q}_d$ into $\mathbf{x}_b$ and $\mathbf{q}_b$, while leaving the residual vectors $\mathbf{x}_r$ and $\mathbf{q}_r$ untouched. This allows the quantization process to be flexible, as the projection dimension d is user-controllable.

4.2 Multi-Stage Distance Computation

After vector decomposition, we focus on computing distances for AKNN search. Given the quantized distance in Equation 2 and the overall distance formulation in Equation 3, we now elaborate on the multi-stage distance computation (as illustrated in Fig. 1).

Overview. Specifically, we define three components of the distance in Equation 3: (1) The norm component: $C_1 := ||\mathbf{x}_d - \mathbf{c}||^2 + ||\mathbf{q}_d - \mathbf{c}||^2 + ||\mathbf{x}_r||^2 + ||\mathbf{q}_r||^2$; (2) The quantized component: $C_2 := -2 \cdot ||\mathbf{x}_d - \mathbf{c}|| \cdot ||\mathbf{q}_d - \mathbf{c}|| \cdot \langle \mathbf{x}_b, \mathbf{q}_b \rangle$; and (3) The residual component: $C_3 := -2 \cdot \langle \mathbf{x}_r, \mathbf{q}_r \rangle$. From an efficiency perspective, the norms of the base vector, i.e., $||\mathbf{x}_d - \mathbf{c}||$ and $||\mathbf{x}_r||$, can be precomputed and stored. Consequently, computing C_1 at query time only requires calculating $||\mathbf{q}_d - \mathbf{c}||$ and $||\mathbf{q}_r||$, which are computed once per query. With all norms precomputed, the remaining computation reduces to evaluating the inner products $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ and $\langle \mathbf{x}_r, \mathbf{q}_r \rangle$.

As shown in Fig. 1, the multi-stage computation proceeds as follows. Stage (1) estimates the distance using the quantized inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ (details provided below). This approximation supports efficient filtering: if the estimated distance is already too large, the candidate is discarded. Otherwise, we proceed to Stage (2), where we re-rank candidates using the original projected inner product $\langle \mathbf{x}_d, \mathbf{q}_d \rangle$, which can be computed directly. If even higher precision is required, Stage (3) computes $\langle \mathbf{x}_r, \mathbf{q}_r \rangle$ to obtain the exact distance. In what follows, we focus on Stage (1)—distance estimation via the quantized inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ and the use of error bounds for re-ranking—as Stages (2) and (3) follow the same reasoning.

Distance Estimation. The next question is how to estimate the quantized inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ for distance approximation. Note that $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ is the inner product between two normalized vectors, where $\mathbf{q}_b := \frac{\mathbf{q}_d - \mathbf{c}}{|\mathbf{q}_d - \mathbf{c}|}$ and $\mathbf{x}_b := \frac{\mathbf{x}_d - \mathbf{c}}{|\mathbf{x}_d - \mathbf{c}|}$. To estimate this inner product, we follow the design of RabitQ to obtain the quantized representation of an arbitrary vector $\mathbf{x}$. Specifically, RabitQ implicitly defines a codebook $C := \left\{-\frac{1}{\sqrt{D}}, +\frac{1}{\sqrt{D}}\right\}^D$, and seeks the closest vector Pu to $\mathbf{x}$, where P is a random orthogonal matrix and $u \in C$. The vector $\mathbf{x}$ is then encoded using the *signs* (positive or negative) of the entries in u, yielding the dimension-wise quantized code $\bar{\mathbf{x}}_b$ of $\mathbf{x}_b$. RabitQ provides an efficient procedure to obtain these signs *without explicitly constructing* the vector u, which makes the encoding highly efficient. The inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ is then approximated using the following unbiased estimator: $\frac{\langle \bar{\mathbf{x}}_b, \mathbf{q}_b \rangle}{\langle \bar{\mathbf{x}}_b, \mathbf{x}_b \rangle}$.

Using this estimator for $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$, the remaining term in the distance computation is $\langle \mathbf{x}_r, \mathbf{q}_r \rangle$. Both empirical evidence and theoretical analysis indicate that the residual component contributes minimally to the total distance. Therefore, we can omit the residual inner product and obtain the final approximate distance:

$$dis' = ||\mathbf{x}_d - \mathbf{c}||^2 + ||\mathbf{q}_d - \mathbf{c}||^2 + ||\mathbf{x}_r||^2 + ||\mathbf{q}_r||^2 - 2 \cdot ||\mathbf{x}_d - \mathbf{c}|| \cdot ||\mathbf{q}_d - \mathbf{c}|| \cdot \langle \bar{\mathbf{x}}_b, \mathbf{q}_b \rangle / \langle \bar{\mathbf{x}}_b, \mathbf{x}_b \rangle \tag{4}$$

A key advantage of MRQ is that the computation of the inner product $\langle \bar{\mathbf{x}}_b, \mathbf{q}_b \rangle$ is highly amenable to hardware acceleration. Since the projection dimension d can be chosen to match SIMD vector widths (e.g., 128, 256, or 512 bits), we can ensure optimal instruction throughput. The binary codes $\bar{\mathbf{x}}_b$ can then be processed efficiently using SIMD instructions (e.g., `POPCNT` on AVX2/AVX-512 registers), which are subsequently mapped to inner-product values. This stands in contrast to methods whose code length is tied to the original vector dimensionality, which may not align with SIMD widths and thus results in suboptimal performance.

Error Bounds for Re-Ranking. Note that for the re-ranking step introduced in Section 2.3, we estimate a lower bound on the distance between a data point $\mathbf{x}$ and the query $\mathbf{q}$, and check whether this bound exceeds the current top-K threshold (i.e., the distance to the K-th nearest neighbor found so far). If so, the candidate can be safely pruned. Otherwise, a more accurate distance computation is required, and we need to proceed to the second and third stages of the distance estimation process (see Fig. 1).

We now analyze the error in approximate distance computation, defined as the gap between the true distance dis and the estimated distance dis'. This error arises from two sources: (1) quantization of the inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ introduces quantization error ϵ_q, and (2) omission of the residual dimensions leads to residual error ϵ_r. Fortunately, due to the decorrelation introduced by PCA-based rotation, each dimension becomes approximately independent. This allows us to analyze ϵ_q and ϵ_r separately. *(1) ϵ_q Part.* The confidence interval of the estimator $\frac{\langle \bar{\mathbf{x}}, \mathbf{q} \rangle}{\langle \bar{\mathbf{x}}, \mathbf{x} \rangle}$ is given by [27]:

$$\mathbb{P}\left\{\left|\frac{\langle \bar{\mathbf{x}}, \mathbf{q} \rangle}{\langle \bar{\mathbf{x}}, \mathbf{q} \rangle} - \langle \mathbf{x}, \mathbf{q} \rangle\right| > \sqrt{\frac{1 - \langle \bar{\mathbf{x}}, \mathbf{x} \rangle^2}{\langle \bar{\mathbf{x}}, \mathbf{x} \rangle^2}} \cdot \frac{\epsilon_0}{\sqrt{D-1}}\right\} \leq 2e^{-c_0 \epsilon_0^2} \tag{5}$$

where ϵ_0 is a tunable parameter controlling the failure probability, and c_0 is a constant [27]. *(2) ϵ_r Part.* For the residual inner product $\langle \mathbf{x}_r, \mathbf{q}_r \rangle$, PCA ensures minimal variance in the residual dimensions. A natural approach is to use this variance to bound the error. Let σ_i denote the standard deviation of the i-th dimension of $\mathbf{x}$, then the variance of $\langle \mathbf{x}_r, \mathbf{q}_r \rangle$ can be expressed as:

$$\sigma^2 = Var(\langle \mathbf{x}_r, \mathbf{q}_r \rangle) = \sum_{i=d+1}^{i \leq D} \mathbf{q}_i^2 \sigma_i^2 \tag{6}$$

Assuming the data is centered (i.e., $\mathbf{x}$ has zero mean), we can then apply Chebyshev's inequality to bound this residual error, that is, $\mathbb{P}(|\langle \mathbf{x}_r, \mathbf{q}_r \rangle| \geq m\sigma) \leq \frac{1}{m^2}$. Since the quantization and residual errors are independent, their bounds can be combined additively to provide an overall error bound for MRQ in distance computation.

Algorithm 1: MRQ Preprocessing

Input: The database raw vector set S and quantization bit d
Output: The IVF index $\mathcal{I}$; The PCA matrix P_p; The random matrix P_r; The quantization code; The pre-compute results of $||\mathbf{x}_d - \mathbf{c}||$, $||\mathbf{x}_r||$ and $\langle \bar{\mathbf{x}}, \mathbf{x} \rangle$; The residual variance σ_i

1 Train PCA matrix P_p with database vector set S;
2 Derive residual variance σ_i from PCA training process;
3 Rotate $\mathbf{x} \in S$ with P_p by $\mathbf{x}_p = P_p\mathbf{x}$;
4 Compute the residual vector norm $||\mathbf{x}_r||$ from $\mathbf{x}_p$;
5 Take the first d-dimension of $\mathbf{x}_p$ and get $\mathbf{x}_d$;
6 Train IVF centroids $\mathbf{c}_d$ and index $\mathbf{x}_d$ to get $\mathcal{I}$;
7 Normalized $\mathbf{x}_d$ and quantized to $\bar{\mathbf{x}}_d$ with random matrix P_r;
8 Compute $||\mathbf{x}_d - \mathbf{c}_d||$ and $\langle \bar{\mathbf{x}}, \mathbf{x} \rangle$;

Algorithm 2: MRQ Query

Input: The query vector $\mathbf{q}$; Index $\mathcal{I}$; N^{nprobe}; Quantized vector $\bar{\mathbf{x}}_d$; Matrix P_p, P_r; PCA data $\mathbf{x}_p$; Error bound parameter ϵ_0 and m; The pre-compute results of $||\mathbf{x}_d - \mathbf{c}||$, $||\mathbf{x}_r||$ and $\langle \bar{\mathbf{x}}, \mathbf{x} \rangle$; The residual variance σ_i
Output: The K nearest neighbor of $\mathbf{q}$

1 Rotate query vector with PCA matrix $\mathbf{q}_p = P_p\mathbf{q}$;
2 Take first d dimension of $\mathbf{q}_p$ as $\mathbf{q}_d$ and rest as $\mathbf{q}_r$;
3 Compute residual variance σ by Equation (6) with $\mathbf{q}_r$ and σ_i;
4 Normalized and random rotate $\mathbf{q}_d$ by P_r to obtain $\mathbf{q}'_d$;
5 Quantized query $\mathbf{q}'_d$ to $\bar{\mathbf{q}}$;
6 Result $Q \leftarrow \emptyset$;
7 **for** ***each*** $ID \in \mathcal{I}$ ***in*** *top* N^{probe} *centroid to* $\mathbf{q}_d$ **do**
8 Compute the approximate distance dis' based on Equation 4; // MRQ
9 $\tau \leftarrow$ threshold of result queue Q;
10 $\epsilon_b \leftarrow$ RabitQ error with ϵ_0; // Quantized Error
11 $\epsilon_r \leftarrow m \cdot \sigma$; // Residual Error
12 **if** $dis' - \epsilon_b - \epsilon_r < \tau$ **then**
13 Compute the project distance dis'_o based on Equation 7; // MRQ$^+$
14 **if** $dis'_o - \epsilon_r < \tau$ **then**
15 compute $dis = ||\mathbf{x}_p, \mathbf{q}_p||^2$;
16 update queue Q with dis;
17 **return** Q;

5 INTEGRATE WITH EXISTING METHOD

Our quantization method MRQ provides an effective way to compute distances, but it must be integrated with existing AKNN search methods for practical deployment. As an example, we implement and integrate MRQ with IVF-based methods, owing to their relatively simple implementation and compact index size.

5.1 Preprocessing Process

We first examine the implementation of MRQ and its integration with the IVF-based indexing structure. Note that IVF partitions the dataset into disjoint clusters, each represented by a centroid that serves as the index entry. The overall preprocessing procedure is summarized in Algorithm 1.

Algorithm. We first apply PCA to project the original data into a lower-dimensional space and obtain the variance of the residual dimensions via eigenvalue decomposition (Lines 1–2). Next, we rotate the original dataset S and compute the norm of the residual vectors (Lines 3–4). We then construct the IVF index using the projected vectors (Line 5). The projected vectors are subsequently quantized using a method similar to RabitQ, where the IVF centroids can also serve to centralize the projected data (Lines 7–8). Note that we use PCA-rotated vectors as base vectors, as Euclidean distances are preserved under the same transformation. With this setup, all information needed for distance computation is ready for efficient use at query time.

5.2 Query Process

After preprocessing, we discuss how to conduct the AKNN search under in-memory settings. The query process is detailed in Algorithm 2. The parameter ϵ_0 controls the confidence interval in Equation (5), while m determines the standard deviation bound based on Chebyshev's inequality. The parameter N^{probe} specifies the number of scanned clusters, balancing efficiency and accuracy.
Query Algorithm. The query process starts with applying PCA rotation to the query vector $\mathbf{q}$ (Line 1), followed by computing the residual variance for distance correction (Lines 2–3). The projected query $\mathbf{q}_d$ is then quantized to accelerate distance computation (Lines 4–5). The IVF index ranks cluster centroids by distance and scans vectors in the top N^{probe} clusters to identify the K nearest neighbors of $\mathbf{q}$, maintaining the top-K candidates in a result queue (heap) Q (Lines 6–7). For each candidate vector, we first estimate the approximate distance using the quantized representation and precomputed values based on Equation 4 (Line 8). We then check whether the estimated distance falls below the current threshold with high probability (Lines 9–12). If it does, we compute the exact distance and update the result queue accordingly (Lines 14–15).
Optimizations. In Fig.1, we propose a multi-stage distance computation scheme. Line 12 of Algorithm 2 performs stage (1), while Line 14 performs stage (3). We further propose an optimization to include stage (2): instead of using quantized distance $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$, we directly use projected distance $\langle \mathbf{x}_d, \mathbf{q}_d \rangle$ to derive

$$dis'_o = ||\mathbf{x}||^2 + ||\mathbf{q}||^2 - 2 \cdot \langle \mathbf{x}_d, \mathbf{q}_d \rangle . \tag{7}$$

Using the residual error bound for re-ranking, we derive the pruning condition $dis'_o - m \cdot \sigma < \tau$ (Line 13) to skip unnecessary exact computations. We also optimize the memory layout following [26] to enhance cache efficiency.

5.3 Discussion

Parameter Selection. A key advantage of MRQ is the flexibility to choose the projection dimension d, enabling it to adapt to different deployment needs. (1) **Memory-constrained setting.** When memory is limited, d can simply be set to the largest value that fits the space budget. (2) **Efficiency-oriented setting.** Recall that d directly affects the multi-stage distance computation, an operation that relies heavily on hardware acceleration (e.g., SIMD). Thus, d should be aligned to SIMD-friendly block sizes (e.g., multiples of 64), but not so large as to incur excessive memory usage. While one could search over d with a step size of 64 to locate a good trade-off, repeatedly rebuilding the index is expensive. To avoid this overhead, we adopt an empirical rule: choose d as the smallest power of two, $2^{i^\star}$, that (i) is at least 128 to match SIMD widths, and (ii) captures at least 80% of the total variance:

$$i^\star = \min \left\{ i \in \mathbb{N} : \ 2^i \geq 128, \ \frac{\sum_{j=1}^{2^i} \sigma_j}{\sum_{j=1}^{d} \sigma_j} \geq 0.8 \right\} . \tag{8}$$

This rule preserves most information while ensuring small space overhead and, as shown in § 6.2, delivers near-optimal performance.

Table 1: Dataset Statistics

Dataset	Dimension	Size	Query Size	Type
MSONG	420	992.272	200	Audio
GIST	960	1,000,000	1000	Image
DEEP	256	1,000,000	1000	Image
TINY	384	5,000,000	1000	Image
WORD2VEC	300	1,000,000	1000	Text
MSMARCO10M	1024	10,000,000	1000	Text
OpenAI-1536	1536	999,000	1000	Text
OpenAI-3072	3072	999,000	1000	Text

Distributed Deployment. When the vector data is extremely large, a natural solution is to distribute the data across a cluster. Our proposed method MRQ readily extends to this setting. In particular, when combined with IVF-based methods, MRQ can be integrated into existing distributed IVF systems such as [31, 64, 72]. Because MRQ introduces no additional dependencies on operations such as indexing and query procedures in Section 5, it can be implemented in a parallel or fully distributed manner. One subtle issue is the use of PCA to obtain global statistics. This can be addressed either by employing distributed PCA or by sampling a small subset of vectors on a single machine. We plan to further explore distributed AKNN search under this framework.

6 EXPERIMENT

6.1 Experiment Settings

Datasets. We evaluate on 8 public datasets, including widely used benchmarks (MSONG, GIST, DEEP, TINY)[1] and datasets derived from recent embedding models: MSMARCO10M (10M slice)[2], OpenAI-1536, and OpenAI-3072[3]. Table 1 summarizes key statistics, including dataset size, query count, dimensionality, and data type (audio, image, or text). For datasets lacking separate query vectors, we randomly sample 1,000 base vectors as queries and remove them from the base set. All data are stored in float32 format.

Evaluation Metrics. We evaluate the accuracy using recall@k (see § 2) for AKNN search, with k set to 20. For efficiency, we measure queries per second (QPS), which is the number of queries processed per second. All metrics are averaged over the entire query set.

Algorithms. We integrate our quantization method MRQ with the IVF method, as described in § 5. For comparison, we evaluate both graph-based and IVF-based AKNN methods as baselines. The evaluated algorithms are as follows:

- IVF-RabitQ: IVF combined with the quantization method RabitQ.
- IVF-RabitQ$^+$: An optimized version of IVF-RabitQ using the same memory-layout optimizations as in IVF-MRQ$^+$ (Algorithm 2).
- IVF-MRQ: IVF integrated with our MRQ method (Algorithm 2 without Lines 13–14).
- IVF-MRQ$^+$: IVF-MRQ enhanced with (i) projection-distance optimization (i.e., using Stage (2) for multi-stage distance computation in Algorithm 2), and (ii) optimized memory layout.
- HNSW [50]: the most widely used graph-based ANN method.

[1]https://www.cse.cuhk.edu.hk/systems/hash/gqr/datasets.html
[2]https://huggingface.co/datasets/Cohere/msmarco-v2.1-embed-english-v3
[3]https://huggingface.co/datasets/Qdrant/dbpedia-entities-openai3-text-embedding-3-large-3072-1M

Configurations. We implement the MRQ method in C++ using g++ version 11.4.0 with the -Ofast optimization flag and -march=native to enable AVX instructions. The number of IVF centroids is set to 4096, following the recommendation of the Faiss library [42]. For the IVF-RabitQ baseline, the quantization bit length is fixed to the original vector dimensionality, as required by its design. For all datasets except `WORD2VEC` and `MSONG`, we use the raw dimensionality. For `WORD2VEC` and `MSONG`, we zero-pad the vectors to dimensions 320 and 448, respectively, to enable more efficient SIMD execution. The quantization bit lengths for MRQ are determined entirely using our empirical rule in § 5.3: 128 bits for `MSONG`, `DEEP`, `TINY`, and `GIST`; 256 bits for `WORD2VEC`; and 512 bits for `OpenAI-1536`, `OpenAI-3072`, and `MSMARCO10M`. For IVF-RabitQ$^+$, we use the same projection dimension as MRQ$^+$ with PCA-rotated vectors for a fair comparison. For IVF-OPQfsr, we set the re-ranking range to 1000, following the best-performing configuration reported in [27]. For the baseline HNSW , we use M = 16 and efConstruction = 500, consistent with the recommended settings and recent benchmarks [5, 50].

6.2 Experiment Results

Exp-1: Test of Performance. Fig. 5 presents the time–accuracy trade-off for all in-memory AKNN search methods, where curves closer to the upper-left corner indicate better performance. Our key findings are as follows:

(1) Superior efficiency while requiring fewer quantization bits. Our distance computation method MRQ significantly outperforms RabitQ in efficiency while requiring substantially fewer quantization bits. Specifically: On the GIST dataset, MRQ achieves a 2× speedup using only 1/7 of the bits; On OpenAI-1536, it delivers a 2.6× improvement with 1/3 of the bits; On OpenAI-3072, it provides a 3.4× gain using just 1/6 of the bits. Moreover, RabitQ$^+$ yields almost no performance improvement over RabitQ, as RabitQ already prunes most unnecessary distance computations, leaving little room for further gains.

(2) Consistent performance advantage over graph-based methods. Our method outperforms the widely used graph-based HNSW in most cases. For instance, on GIST, MRQ is 5.85× faster than HNSW at high recall levels. While HNSW performs better on OpenAI-1536 and OpenAI-3072 due to scanning fewer vectors than IVF at equivalent recall, it incurs significantly higher index construction time and memory cost. Moreover, HNSW performs poorly on WORD2VEC, whereas MRQ maintains competitive performance across all datasets.

Exp-2: Preprocessing Time and Space. We report the preprocessing time and space cost (excluding base vectors) for all compared AKNN methods in Fig. 6.

(1) Preprocessing Time. As shown in Fig. 6 (a), our method achieves significantly faster index construction. In particular, it requires only half the time of RabitQ and one-quarter of HNSW.

(2) Preprocessing Space. Fig. 6 (b) shows that our method also incurs substantially lower space overhead. For example, it consumes over 5× less memory than HNSW and 2× less than IVF-RabitQ. The advantages in both preprocessing time and size make our method particularly suitable for resource-constrained environments.

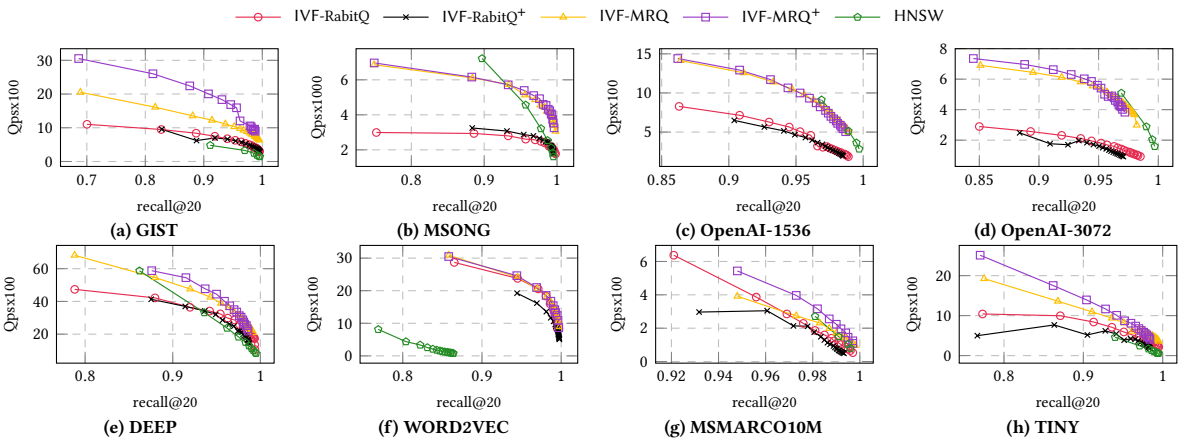


Figure 5: The Performance Test Among Various Method

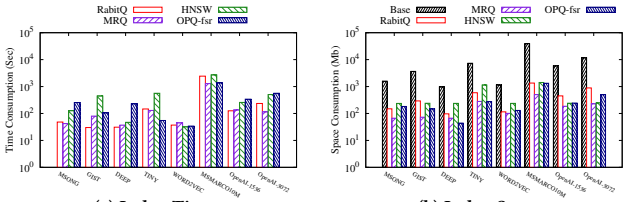


Figure 6: The Comparison of the Time Cost and Index Space

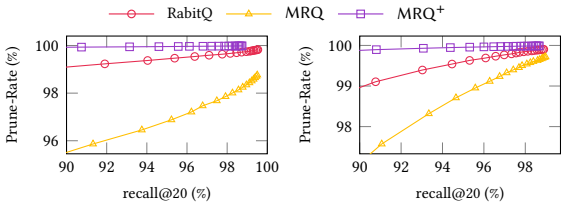


Figure 7: The In-Depth Analysis of Our Methods' Effectiveness

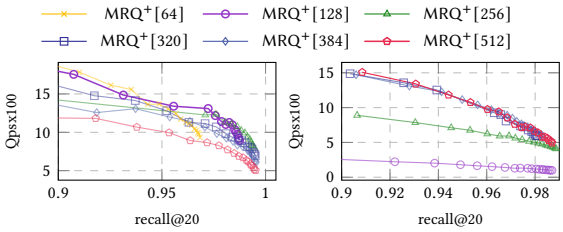


Figure 8: The Validation of Parameters Selection

Exp-3: Understanding Effectiveness. To clarify why the proposed method MRQ is effective for AKNN search, we conduct an in-depth analysis against the state-of-the-art quantization method RabitQ. We measure the *prune ratio*, defined as the fraction of pruned distance computations over all exact distance computations—where a larger ratio indicates greater effectiveness because more unnecessary computations are removed.

Fig. 7 shows that RabitQ can prune more than 99% of exact distance computations (y-axis: prune rate using original vectors). Although RabitQ is strong in pruning, its fixed compression ratio prevents it from operating efficiently with fewer bits. Even when we adapt the same distance-optimization and memory-layout techniques used in MRQ (i.e., to form RabitQ+), the overall speedup remains limited, as also confirmed in Fig. 5. In contrast, MRQ uses only 1/8–1/3 as many bits while still pruning 96%–98% of exact distance computations—losing only 1–2% compared with RabitQ. Moreover, the quantized-distance optimization in MRQ compensates for the reduced bit budget by leveraging higher-precision projection distances, leading to additional efficiency gains.

Exp-4: Parameter Selection. To validate the parameter-selection strategy introduced in § 5.3, we evaluate its effectiveness in real AKNN search scenarios. Specifically, we report the search performance obtained from a grid search over the projection dimension d with a step size of 64 (Fig. 8). From the variance distribution in Fig. 2, the `GIST` dataset requires the first 78 projected dimensions to capture 80% of the total variance, whereas `OpenAI-1536` requires 354. Following our empirical rule—which aligns d to the nearest multiple of 64 for efficient SIMD execution—we set $d = 128$ for `GIST` and $d = 512$ for `OpenAI-1536`. The efficiency curves in Fig. 8 show that these empirically chosen values (solid-line markers) achieve near-optimal search performance across the grid. This empirical rule is used throughout all experiments in Fig. 5, offering a low-cost yet effective guideline for efficiency-oriented index configuration.

7 CONCLUSION

This paper proposes a novel method, MRQ, for AKNN search. MRQ decomposes each vector into quantized and residual components via projection. The quantized component operates on user-controllable dimensions, enabling arbitrary compression ratios. The residual component is modeled using lightweight statistical summaries, further reducing memory usage. By combining both components, MRQ performs a multi-stage distance computation tailored for efficient and accurate AKNN search. Extensive experiments show that MRQ greatly improves search efficiency with negligible index construction time and storage overhead. Future work will investigate distributed processing techniques for large-scale vector datasets.

REFERENCES

[1] Cecilia Aguerrebere, Ishwar Singh Bhati, Mark Hildebrand, Mariano Tepper, and Theodore Willke. 2023. Similarity Search in the Blink of an Eye with Compressed Indices. *Proceedings of the VLDB Endowment* 16, 11 (2023), 3433–3446.

[2] Cecilia Aguerrebere, Mark Hildebrand, Ishwar Singh Bhati, Theodore Willke, and Mariano Tepper. 2024. Locally-Adaptive Quantization for Streaming Vector Search. *arXiv preprint arXiv:2402.02044* (2024).

[3] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache locality is not enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. *Proceedings of the VLDB Endowment* 9, 4 (2015).

[4] Fabien Andre, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2019. Quicker adc: Unlocking the hidden potential of product quantization with simd. *IEEE transactions on pattern analysis and machine intelligence* 43, 5 (2019), 1666–1677.

[5] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems* 87 (2020), 101374.

[6] Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2023. Elpis: Graph-based similarity search for scalable data science. *Proceedings of the VLDB Endowment* 16, 6 (2023), 1548–1559.

[7] Artem Babenko and Victor Lempitsky. 2014. Additive quantization for extreme vector compression. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 931–938.

[8] Artem Babenko and Victor Lempitsky. 2015. Tree quantization for large-scale similarity search and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4240–4248.

[9] Artem Babenko and Victor S. Lempitsky. 2015. The Inverted Multi-Index. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 6 (2015), 1247–1260.

[10] Alina Beygelzimer, Sham Kakade, and John Langford. 2006. Cover trees for nearest neighbor. In *Proceedings of the 23rd international conference on Machine learning*. 97–104.

[11] Yuzheng Cai, Jiayang Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating labels and vectors: A unified approach to filtered approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 2, 6 (2024), 1–27.

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

[13] Patrick Chen, Wei-Cheng Chang, Jyun-Yu Jiang, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2023. FINGER: Fast Inference for Graph-based Approximate Nearest Neighbor Search. In *Proceedings of the ACM Web Conference 2023*. 3225–3235.

[14] Xinyan Dai, Xiao Yan, Kelvin KW Ng, Jiu Liu, and James Cheng. 2020. Norm-explicit quantization: Improving vector quantization for maximum inner product search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 51–58.

[15] Sanjoy Dasgupta and Yoav Freund. 2008. Random projection trees and low dimensional manifolds. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*. 537–546.

[16] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*. 253–262.

[17] Liwei Deng, Penghao Chen, Ximu Zeng, Tianfu Wang, Yan Zhao, and Kai Zheng. 2024. Efficient Data-aware Distance Comparison Operations for High-Dimensional Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2411.17229* (2024).

[18] Yihe Dong, Piotr Indyk, Ilya P Razenshteyn, and Tal Wagner. 2020. Learning Space Partitions for Nearest Neighbor Search. *ICLR* (2020).

[19] Karima Echihabi, Panagiota Fatourou, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2022. Hercules against data series similarity search. *Proceedings of the VLDB Endowment* 15, 10 (2022), 2005–2018.

[20] Elastic. [n.d.]. *Elasticsearch*. https://www.elastic.co/elasticsearch/

[21] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. 2000. Vector approximation based indexing for non-uniform high dimensional data sets. In *Proceedings of the ninth international conference on Information and knowledge management*. 202–209.

[22] Cong Fu, Changxu Wang, and Deng Cai. 2022. High Dimensional Similarity Search With Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 8 (2022), 4139–4150.

[23] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *Proc. VLDB Endow.* 12, 5 (2019), 461–474.

[24] Junhao Gan, Jianlin Feng, Qiong Fang, and Wilfred Ng. 2012. Locality-sensitive hashing scheme based on dynamic collision counting. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*. 541–552.

[25] Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. 2024. Practical and Asymptotically Optimal Quantization of High-Dimensional Vectors in Euclidean Space for Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2409.09913* (2024).

[26] Jianyang Gao and Cheng Long. 2023. High-Dimensional Approximate Nearest Neighbor Search: with Reliable and Efficient Distance Comparison Operations. *Proc. ACM Manag. Data* 1, 2 (2023), 137:1–137:27. https://doi.org/10.1145/3589282

[27] Jianyang Gao and Cheng Long. 2024. 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 (2024), 1–27.

[28] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2014. Optimized Product Quantization. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 4 (2014), 744–755.

[29] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. 1999. Similarity search in high dimensions via hashing. In *Vldb*, Vol. 99. 518–529.

[30] Michel X Goemans and David P Williamson. 1995. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)* 42, 6 (1995), 1115–1145.

[31] Shenghao Gong, Haobo Sun, Ziquan Fang, Liu Liu, Lu Chen, and Yunjun Gao. 2025. VStream: A Distributed Streaming Vector Search System. *Proceedings of the VLDB Endowment* 18, 6 (2025), 1593–1606.

[32] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 12 (2013), 2916–2929. https://doi.org/10.1109/TPAMI.2012.193

[33] Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. 2025. SymphonyQG: Towards Symphonious Integration of Quantization and Graph for Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–26.

[34] Robert Gray. 1984. Vector quantization. *IEEE Assp Magazine* 1, 2 (1984), 4–29.

[35] Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. 2016. Quantization based fast inner product search. In *Artificial intelligence and statistics*. PMLR, 482–490.

[36] Ben Harwood and Tom Drummond. 2016. Fanng: Fast approximate nearest neighbour graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 5713–5722.

[37] Qiang Huang, Jianlin Feng, Qiong Fang, Wilfred Ng, and Wei Wang. 2017. Query-aware locality-sensitive hashing scheme for lp norm. *The VLDB Journal* 26, 5 (2017), 683–708.

[38] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. 2015. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proceedings of the VLDB Endowment* 9, 1 (2015), 1–12.

[39] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. 604–613.

[40] Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. 2019. Diskann: Fast accurate billion-point nearest neighbor search on a single node. *Advances in Neural Information Processing Systems* 32 (2019).

[41] Mengxu Jiang, Zhi Yang, Fangyuan Zhang, Guanhao Hou, Jieming Shi, Wenchao Zhou, Feifei Li, and Sibo Wang. 2025. DIGRA: A Dynamic Graph Indexing for Approximate Nearest Neighbor Search with Range Filter. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–26.

[42] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2019), 535–547.

[43] Yannis Kalantidis and Yannis Avrithis. 2014. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2321–2328.

[44] Weihao Kong and Wu-Jun Li. 2012. Isotropic hashing. *Advances in neural information processing systems* 25 (2012).

[45] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems* 33 (2020), 9459–9474.

[46] Jinfeng Li, Xiao Yan, Jian Zhang, An Xu, James Cheng, Jie Liu, Kelvin KW Ng, and Ti-chung Cheng. 2018. A general and efficient querying method for learning to hash. In *Proceedings of the 2018 International Conference on Management of Data*. 1333–1347.

[47] Ying Liu, Dengsheng Zhang, Guojun Lu, and Wei-Ying Ma. 2007. A survey of content-based image retrieval with high-level semantics. *Pattern recognition* 40, 1 (2007), 262–282.

[48] Kejing Lu, Mineichi Kudo, Chuan Xiao, and Yoshiharu Ishikawa. 2021. HVS: Hierarchical Graph Structure Based on Voronoi Diagrams for Solving Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 15, 2 (2021), 246–258.

[49] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. 2014. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.* 45 (2014), 61–68.

[50] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[51] Julieta Martinez, Shobhit Zakhmi, Holger H. Hoos, and James J. Little. 2018. LSQ++: Lower running time and higher recall in multi-codebook quantization. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

[52] Nasser M Nasrabadi and Robert A King. 1988. Image coding using vector quantization: A review. *IEEE Transactions on communications* 36, 8 (1988), 957–971.

[53] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. Acorn: Performant and predicate-agnostic search over vector embeddings and structured data. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–27.

[54] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proc. ACM Manag. Data* 1, 1 (2023), 54:1–54:27. https://doi.org/10.1145/3588908

[55] Maxim Raginsky and Svetlana Lazebnik. 2009. Locality-sensitive binary codes from shift-invariant kernels. *Advances in neural information processing systems* 22 (2009).

[56] J Ben Schafer, Dan Frankowski, Jon Herlocker, and Shilad Sen. 2007. Collaborative filtering recommender systems. In *The adaptive web: methods and strategies of web personalization*. Springer, 291–324.

[57] Yifang Sun, Wei Wang, Jianbin Qin, Ying Zhang, and Xuemin Lin. 2014. SRS: Solving c-Approximate Nearest Neighbor Queries in High Dimensional Euclidean Space with a Tiny Index. *Proc. VLDB Endow.* 8, 1 (2014), 1–12.

[58] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2017. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence* 40, 4 (2017), 769–790.

[59] Mengzhao Wang, Haotian Wu, Xiangyu Ke, Yunjun Gao, Yifan Zhu, and Wenchao Zhou. 2025. Accelerating Graph Indexing for ANNS on Modern CPUs. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–29.

[60] Mengzhao Wang, Weizhi Xu, Xiaomeng Yi, Songlin Wu, Zhangyang Peng, Xiangyu Ke, Yunjun Gao, Xiaoliang Xu, Rentong Guo, and Charles Xie. 2024. Starling: An I/O-Efficient Disk-Resident Graph Index Framework for High-Dimensional Vector Similarity Search on Data Segment. *Proc. ACM Manag. Data* 2, 1 (2024), V2mod014:1–V2mod014:27. https://doi.org/10.1145/3639269

[61] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment* 14, 11 (2021), 1964–1978.

[62] Runhui Wang and Dong Deng. 2020. DeltaPQ: lossless product quantization code compression for high dimensional similarity search. *Proceedings of the VLDB Endowment* 13, 13 (2020), 3603–3616.

[63] Jiadong Xie, Jeffrey Xu Yu, and Yingfan Liu. 2025. Graph Based K-Nearest Neighbor Search Revisited. *ACM Transactions on Database Systems* (2025).

[64] Qian Xu, Feng Zhang, Chengxi Li, Lei Cao, Zheng Chen, Jidong Zhai, and Xiaoyong Du. 2025. Harmony: A scalable distributed vector database for high-throughput approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 3, 4 (2025), 1–28.

[65] Mingyu Yang, Liuchang Jing, Wentao Li, and Wei Wang. 2025. *Technical Report*. github.com/mingyu-hkustgz/RESQ/technical_report.pdf

[66] Mingyu Yang, Wentao Li, Jiabao Jin, Xiaoyao Zhong, Xiangyu Wang, Zhitao Shen, Wei Jia, and Wei Wang. 2025. Effective and General Distance Computation for Approximate Nearest Neighbor Search. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 1098–1110.

[67] Ziqi Yin, Jianyang Gao, Pasquale Balsebre, Gao Cong, and Cheng Long. 2025. DEG: Efficient Hybrid Vector Search Using the Dynamic Edge Navigation Graph. *Proceedings of the ACM on Management of Data* 3, 1 (2025), 1–28.

[68] Qiang Yue, Xiaoliang Xu, Yuxiang Wang, Yikun Tao, and Xuliyuan Luo. 2024. Routing-Guided Learned Product Quantization for Graph-Based Approximate Nearest Neighbor Search. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 4870–4883.

[69] Fangyuan Zhang, Mengxu Jiang, Guanhao Hou, Jieming Shi, Hua Fan, Wenchao Zhou, Feifei Li, and Sibo Wang. 2025. Efficient Dynamic Indexing for Range Filtered Approximate Nearest Neighbor Search. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–26.

[70] Ting Zhang, Chao Du, and Jingdong Wang. 2014. Composite quantization for approximate nearest neighbor search. In *International Conference on Machine Learning*. PMLR, 838–846.

[71] Bolong Zheng, Xi Zhao, Lianggui Weng, Nguyen Quoc Viet Hung, Hang Liu, and Christian S. Jensen. 2020. PM-LSH: A Fast and Accurate LSH Framework for High-Dimensional Approximate NN Search. *Proc. VLDB Endow.* 13, 5 (2020), 643–655.

[72] Xiangyu Zhi, Meng Chen, Xiao Yan, Baotong Lu, Hui Li, Qianxi Zhang, Qi Chen, and James Cheng. 2025. Towards Efficient and Scalable Distributed Vector Search with RDMA. *arXiv preprint arXiv:2507.06653* (2025).

[73] Xiaoyao Zhong, Haotian Li, Jiabao Jin, Mingyu Yang, Deming Chu, Xiangyu Wang, Zhitao Shen, Wei Jia, George Gu, Yi Xie, et al. 2025. VSAG: An Optimized Search Framework for Graph-based Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2503.17911* (2025).

[74] Wengang Zhou, Yijuan Lu, Houqiang Li, and Qi Tian. 2012. Scalar quantization for large scale image search. In *Proceedings of the 20th ACM international conference on Multimedia*. 169–178.

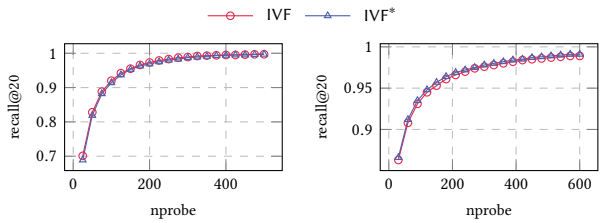
[75] Chaoji Zuo, Miao Qiao, Wenchao Zhou, Feifei Li, and Dong Deng. 2024. SeRF: segment graph for range-filtering approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 2, 1 (2024), 1–26.

A. APPENDIX

A.1. Extra Discussion

We further evaluate the variance statistics of several representative datasets. As shown in Fig. 10, vector embeddings from images, text, and audio consistently exhibit a long-tailed variance distribution after rotation by the PCA matrix. However, the cumulative variance captured by the tail dimensions differs across data modalities. For example, Word2Vec captures less than 80% of the total variance using the top 50% of its dimensions, whereas GIST reaches over 80% using only 10% of its dimensions.

These observations align with our empirical parameter-setting rule in Eq. 8. The projection dimensions required to preserve 80% of the total variance for the GIST, MSONG, OpenAI-1536, OpenAI-3072, DEEP, Word2Vec, MSMARCO, and TINY datasets are 128, 5, 354, 470, 46, 173, 273, and 51, respectively. Following this rule, we set the projected dimensionality to 128 bits for MSONG, DEEP, TINY, and GIST; 256 bits for Word2Vec; and 512 bits for OpenAI-1536, OpenAI-3072, and MSMARCO10M, achieving near-optimal search efficiency across all datasets.



(a) GIST **(b) OpenAI-1536**

Figure 9: The Study of Centroid Approximation

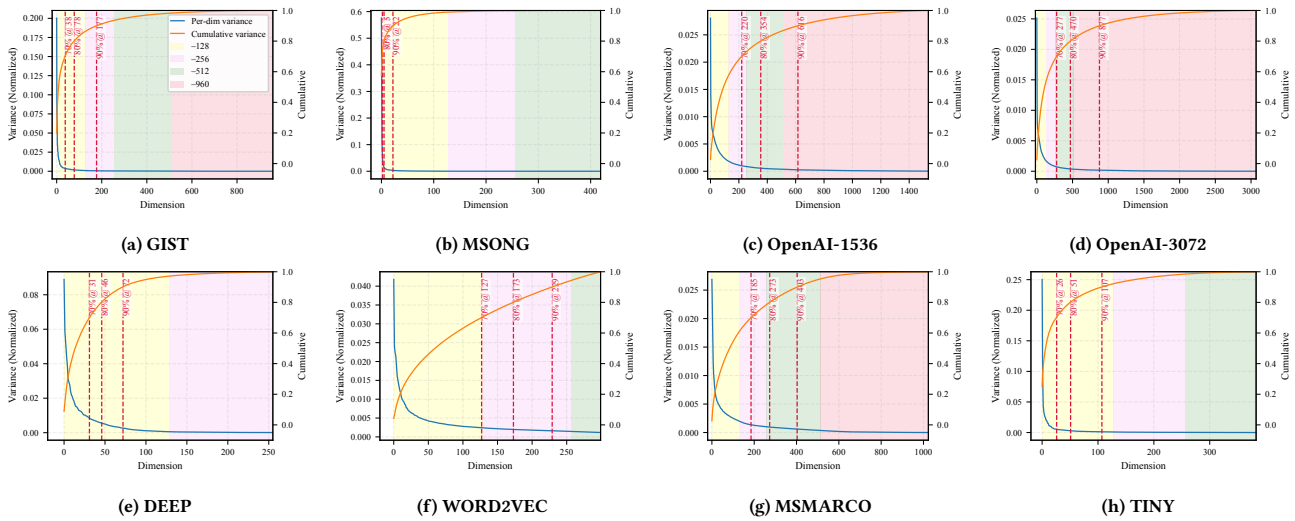
A.2. Extra Experiments

Exp-5: Study of Centroid Approximation. In Algorithm 1, we apply the k-means algorithm on the projected (approximate) vectors instead of the original full-dimensional vectors. This results in approximate centroids, as used in the IVF method. The effectiveness of this approach is shown in Fig. 9, where the x-axis indicates the number of scanned clusters. The red line labeled IVF represents the original centroids computed from the full-dimensional vectors, while the blue line labeled IVF* represents the approximate centroids computed from the projected vectors. Specifically, IVF uses all D dimensions to compute the centroids, whereas IVF* uses only 128 dimensions for the GIST dataset (1/7 of the original dimensionality) and 512 dimensions for the OpenAI-1536 dataset (1/3 of the original dimensionality).

As shown in Fig. 9, the projected centroids incur almost no recall loss on the GIST dataset and even slightly outperform the original centroids on the OpenAI-1536 dataset. Therefore, we adopt the projected centroids in our method. Additionally, this projection significantly reduces the training time for the IVF index, making it more efficient for large-scale datasets.

Exp-6: Comparison with Additional Baselines. To further validate the superiority of our proposed quantization method MRQ, we first disable the projection-distance and memory-layout optimizations. (1) We also include additional quantization techniques, such as Product Quantization (PQ), as baselines. For a fair comparison in terms of memory usage, IVF-PQ is configured with a 32× compression ratio so that its space consumption matches that of IVF-RabitQ. Note that both IVF-PQ and IVF-Truncate are evaluated without re-ranking. (2) Under the re-ranking setting, we additionally compare against the state-of-the-art Binary Quantization (BQ) method RabitQ, as well as the optimized PQ variant, OPQ-4bit-fastscan (denoted IVF-OPQfsr). (3) We also design an IVF-Truncate baseline with re-ranking, which directly truncates vectors to match the bit budget of MRQ. This method is similar in spirit to MRQ but lacks PCA, and thus cannot flexibly control the separation of leading and residual dimensions. All results are reported in Fig. 11.

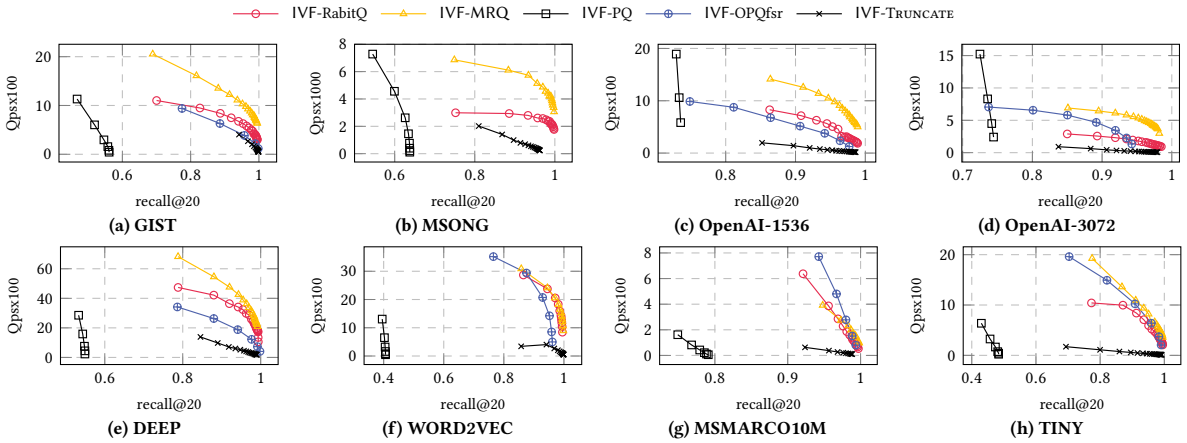
As shown in Fig. 11, without re-ranking, standard PQ-based quantization fails to achieve 90% recall across datasets. Using fewer bits—comparable to the setting of MRQ—only further degrades accuracy. With re-ranking enabled, the results show that MRQ significantly outperforms both IVF-RabitQ and IVF-OPQfsr on the vast majority of datasets. Moreover, compared with Truncate, MRQ exhibits a substantial performance advantage, demonstrating the importance of PCA-based projection. Notably, MRQ achieves this superior accuracy relying solely on quantized distances for re-ranking, even without multi-stage distance computation or memory-layout optimizations.



(a) GIST (b) MSONG (c) OpenAI-1536 (d) OpenAI-3072

(e) DEEP (f) WORD2VEC (g) MSMARCO (h) TINY

Figure 10: The Variance Distributions of Vectors After PCA



(a) GIST (b) MSONG (c) OpenAI-1536 (d) OpenAI-3072

(e) DEEP (f) WORD2VEC (g) MSMARCO10M (h) TINY

Figure 11: The Test of Additional Quantization Methods