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

We would like to thank the reviewers for their insightful and valuable comments. We have revised the paper carefully according to the comments. A summary of major revisions is given below, followed by the point-to-point response to each reviewer.

(1) Strengthened Motivation: We have revised the Introduction (§1) to broaden the motivation beyond the limitations of a single method (RabitQ). We now frame the problem as a general challenge for high-ratio quantization methods that tie code length to vector dimensionality, making our contribution more significant. The analysis from the original §3 has been integrated into the Introduction to support this.

(2) Improved Related Work: We have expanded the Related Work section (§2) to include a critical analysis of prior methods (PQ, BQ, RabitQ), discussing their limitations and contrasting them with the advantages of MRQ.

(3) Deeper Experimental Analysis: We have enriched the experimental results (§6.2) with explanations for the observed performance phenomena, detailing why MRQ achieves superior efficiency in terms of SIMD utilization and cache locality.

(4) More Comprehensive Baselines: We have added descriptions for new baselines in our experimental setup (§6.1) as suggested, including fundamental methods like **IVF-PQ** and **IVF-OPQ**, and an optimized **IVF-RabitQ+** to ensure a fair comparison against our optimized IVF-MRQ+ variant.

(5) Clarifications and Discussions: We have added clarifications throughout the paper regarding hardware acceleration (§4.2), distributed scalability (§7), parameter selection (§6.2), and specific implementation details requested by the reviewers.

All the updates have been marked in blue in the manuscript. Please find our point-to-point responses below.

1 RESPONSE TO REVIEWER # 2

O1. The motivation of MRQ is mainly based on RabitQ's limitation that the number of quantization bits must equal the vector dimensionality, leading to a fixed compression ratio. It remains unclear whether this is a general bottleneck for quantization-based AKNN methods or just a RabitQ-specific drawback. If it is only the latter, the contribution may appear limited; if it is the former, the authors should emphasize its prevalence and practical importance to strengthen the motivation.

Response: Thank you for this insightful suggestion. We agree that the motivation needs to be strengthened by generalizing the problem. The limitation of a fixed compression ratio is not merely a drawback of RabitQ but a characteristic of a class of high-efficiency quantization methods that rely on hardware acceleration and error-bounded re-ranking. These methods often require the code length to match the vector dimensionality to enable certain mathematical properties or efficient SIMD operators, such as the unbiased inner product estimator used in RabitQ [4] and Ex-RabitQ [3] or low-precision SIMD instruction sets used by SQ methods such as LVQ [1]. This requirement creates a fundamental trade-off: to gain theoretical error bounds and high accuracy, one must sacrifice the flexibility of choosing the compression ratio, which is critical for memory-constrained environments and for optimizing hardware utilization (e.g., SIMD).

We have revised the **Introduction (§ 1)** to emphasize this general bottleneck. We now position RabitQ as the state-of-the-art example of this class of methods and frame MRQ as a solution that decouples quantization bit length from vector dimensionality, thereby resolving this broader, more prevalent challenge.

O2. The related work section only lists prior methods without discussing their limitations or contrasting them with the advantages of MRQ.

Response: Thank you for the comment. We have substantially revised the **Related Work Section (§ 2)** to provide a more analytical discussion. We now explicitly detail the limitations of fundamental techniques like Product Quantization (PQ) and Binary Quantization (BQ), focusing on their inherent trade-off between compression and accuracy. We then contrast these methods with MRQ, highlighting how our approach combines the high compression of quantization with the high accuracy of principled re-ranking, a feature previously constrained by fixed compression ratios. This revised section now better contextualizes MRQ's contribution relative to existing work.

O3. The discussion of limitations in Section 3 would fit more naturally in the Introduction, where it could clearly frame the motivation for the proposed work.

Response: Thank you for the comments. We have moved the core analysis from the original § 3 into the **Introduction (*S* 1)**. This includes the discussion of the fixed-compression-ratio limitation and our key observation about the long-tailed variance distribution of high-dimensional data. This change helps to frame the problem and motivate our solution more clearly and effectively from the outset. The original § 3 has been streamlined and repurposed to serve as a concise bridge to our proposed method.

O4. The experimental analysis is weak, as many results are reported as observed phenomena without sufficient explanation of the underlying causes.

Response: Thank you for the comments. We have revised the **Experimental Results (**§ 6.2**)** to include deeper analysis and explanations for the observed performance. For instance, when discussing MRQ's speedup over RabitQ, we now explain that this is due to two primary factors: (1) MRQ's user-controllable bit length can be aligned with SIMD register widths (e.g., 128/256 bits), leading to more efficient hardware utilization than RabitQ's often non-aligned bit lengths. (2) The smaller space cost of MRQ's codes improves cache locality, reducing latency during the inverted list scan. We have added similar causal explanations throughout the section to provide a more thorough analysis.

O5. It would be helpful to include an IVF-RabitQ+ variant (RabitQ with optimized distance computations and memory layout) to better isolate the impact of engineering optimizations independently of MRQ.

Response: Thank you for the comments. We have updated our **Experimental Setting (**§ **6.1)** to include a new baseline, **IVF-RabitQ+**, which applies the same memory layout and distance computation optimizations used in IVF-MRQ+ to the RabitQ method. This new baseline will allow for a direct comparison of the quantization schemes themselves, providing a clearer picture of MRQ's inherent advantages.

O6. As fundamental quantization methods, PQ and BQ should be included as baselines to ensure a more comprehensive and convincing experimental comparison.

Response: Thank you for the comments. We have updated our **Experimental Setting (**§ **6.1)** to include **IVF-PQ** and **IVF-OPQ** as baselines. For these experiments, we will use a bit budget comparable to that of MRQ to provide a fair assessment of the trade-off between compression, accuracy, and speed across different quantization paradigms. This will make our experimental evaluation more comprehensive and robust.

O7. The paper notes that PQ and BQ benefit from hardware acceleration, but it is unclear whether MRQ can do so; this should be clarified.

Response: Thank you for the comments. MRQ is indeed designed to leverage hardware acceleration, particularly SIMD instructions. The core of MRQ's distance computation involves an inner product on binary codes, which is efficiently calculated using bitwise operations (e.g., *popcnt*). A key advantage of MRQ is that the quantization dimension d is user-controllable. This allows us to choose a bit length (e.g., 128, 256, 512) that perfectly aligns with SIMD register widths, maximizing computational throughput. This is a significant advantage over methods like RabitQ, where the code length is tied to the original vector dimension, which is often not a power of two and leads to suboptimal SIMD performance. We have added a detailed clarification on this in § 4.2.

2 RESPONSE TO REVIEWER # 4

W1&C6. How can this method scale to distributed systems for scalability? The authors need to discuss how to parallelize their method in a distributed setting to scale to large datasets. Which components of their system are easily distributed and which ones are blockers and need centralized processing.

Response: Thank you for the comment. Our MRQ method is naturally suited for distributed systems, largely due to its foundation on the IVF index structure. The partitioning scheme of the IVF index, where the dataset is divided into independent clusters, lends itself perfectly to a distributed deployment.

During index construction, a distributed clustering algorithm (e.g., distributed k-means) can be employed to partition the data. Subsequently, the data points within each cluster (or a group of clusters) can be assigned to a specific worker node, creating data shards. At query time, a central query broker, which stores only the lightweight cluster centroids, can first identify the `nprobe` clusters closest to the query. It then dispatches the search task *only* to the worker nodes that are responsible for these target clusters. This *scatter-gather* approach enables parallel processing and significantly reduces the search space for each query. We have incorporated a discussion of this scalability aspect into § 5.3.

■

W2&C8. How do you select the system parameters (e.g., quantization bits, number of dimensions, ec.)

Response: Thank you for raising these important points. Thank you for this excellent question regarding hyperparameter selection. The optimal choice for the projection dimension d (which also defines the number of quantization bits) involves a trade-off between resource constraints and search performance. We propose two perspectives for its selection.

- **Resource-Driven Selection:** From a resource-driven perspective, d can be directly determined by a predefined memory budget. For instance, in a memory-constrained environment that allows for a maximum of 300 bits per vector, one would set $d = 300$ to maximize the information captured within that strict space limit.
- **Efficiency-Driven Selection:** From an efficiency-driven perspective, the optimal d is one that maximizes QPS. This optimum depends on a complex interplay between the costs of multi-stage distance computations and hardware capabilities, particularly SIMD width. The theoretically optimal approach would be a grid search over values of d, likely in increments of 64 to align with SIMD registers. However, this incurs a prohibitive tuning cost.

To address this, we provide a practical and effective **heuristic** in our discussion. We recommend selecting d as the smallest power of two (i.e., $d = 2^i$) such that the cumulative variance of the first d principal components exceeds a certain threshold, such as 80%. This approach provides a strong starting point that balances information preservation with computational efficiency (by choosing powers of two that are friendly to binary operations). We have validated the effectiveness of this heuristic in the Appendix, demonstrating that it consistently yields high-performing configurations across different datasets. We have added this detailed explanation to the new **Impact of Projection Dimension** subsection in our experimental analysis (§6.2).

■

3 RESPONSE TO REVIEWER # 5

W1. The claim that "RabitQ requires the binary quantization code to be of the same dimension with the original vector" is not entirely accurate, as one can pad the original vectors with 0's before generating the quantization code. Although MRQ could technically support arbitrary compression ratio, the length of the quantization code is subject to the same constraints of needing align with SIMD width.

Response: We thank the reviewer for pointing this out. We would like to clarify the distinction between zero padding and dimensionality reduction to highlight the unique advantage of MRQ.

- **Compression Flexibility:** While we agree that padding original vectors with zeros allows RabitQ to adjust dimensions, this operation is monotonic: it can only *increase* or *maintain* the code length. Padding allows the code length L to be greater than or equal to the input dimension D (i.e., low-ratio compression). However, padding does not enable RabitQ to generate a code where $L < D$. MRQ, in contrast, specifically addresses the challenge of **high-ratio compression** (where the code length is significantly smaller than the original dimension), which is impossible for standard binary quantization methods like RabitQ to achieve via zero-padding.
- **Nature of SIMD Constraints:** For MRQ, the constraint is merely an *implementation overhead*: we only need to load the final quantization code to the nearest multiple of the SIMD width (e.g., padding a 90-bit code to 128 bits on the fly without storing it). This cost is negligible and constant (< 1 SIMD block). We will revise the manuscript to clarify that while padding solves alignment for RabitQ, it does not solve the limitation regarding high-ratio compression.

■

D1. What is the length of the quantization code for RabitQ in the evaluation? Is it equal to the vector embedding dimension, or padded to be multiples of e.g., 64?

Response: Thank you for this clarifying question. We followed the implementation details from the original RabitQ paper [4] and padded the vectors with zeros to the nearest dimension that is a multiple of 64 before applying quantization. This practice is common for optimizing performance, as it ensures efficient memory alignment and better utilization of SIMD instructions. We have updated the *Configurations* subsection in § 6.1 to explicitly state this padding procedure for the RabitQ baseline.

■

D2. The difference between IVF-MRQ and IVF-MRQ+ is also not clear, as IVF-MRQ/Algorithm 2 already uses the optimized distance computation.

Response: Thank you for constructive comments. Unlike MRQ, which computes only quantized distances, MRQ+ employs a two-stage computation: it first computes quantized distances for coarse pruning and subsequently applies projection-based distances for refinement. This design choice is now clearly reflected in the revised algorithm description and experiment section.

■

D3. In Table 3, why does RabitQ sometimes have higher storage than HNSW? I was expecting that quantization method should save space overall.

Response: Thank you for pointing this out. The reason is that the total storage for IVF-RabitQ includes not only the quantized vector but also precomputed metadata required for its distance estimator. Specifically, for each D-dimensional vector, RabitQ stores a D-bit code, the vector's L2 norm (4 bytes), and the precomputed inner product $\langle \bar{\mathbf{x}}, \mathbf{x} \rangle$ (4 bytes). This sums to $(D/8 + 8)$ bytes per vector. For HNSW, the index size is primarily the space for storing neighbor links (e.g., $M \times 4$ bytes per vector). On datasets with very high dimensionality (e.g., OpenAI-1536), the per-vector overhead of RabitQ can exceed that of HNSW with a modest M (e.g., 16), since we exclude the base vector space cost. In contrast, our MRQ method drastically reduces this overhead by quantizing only a small d-dimensional projection. We have added a detailed explanation for this phenomenon in § **6.2 (Exp-3)**.

■

D4. How to determine the projection dimensions d? It might help to show a tradeoff between compression ratio vs search performance with different projection dimensions.

Response: Thank you for the suggestion. We determine the projection dimension d using a practical heuristic guided by both resource and efficiency considerations:

- **Resource-driven:** If a memory budget is given (e.g., B bits per vector), we set $d = B$ to maximize information within the budget.
- **Efficiency-driven:** We select d to maximize QPS given SIMD width and multi-stage distance costs. Rather than an expensive grid search, we adopt a strong default: choose the smallest power of two $d = 2^i$ such that the first d principal components explain at least 80% of the variance.

This balances compression and speed (powers of two align with hardware) while preserving most information. We have added a new subsection (§ 6.2, **Impact of Projection Dimension**) reporting the compression–performance trade-off across datasets, showing that this heuristic consistently achieves near-optimal recall/QPS without exhaustive tuning.

■

D5. [54] was referenced in the introduction as a recent survey, but it was from 2017.

Response: Thank you for catching this. You are correct that the reference is dated. We have updated the citation in the **Introduction (§ 1)** to point to a more recent and relevant surveys [2, 5] on AKNN search and vector quantization.

■

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] Jianyang Gao, Yutong Gou, Yuexuan Xu, Jifan Shi, Cheng Long, Raymond Chi-Wing Wong, and Themis Palpanas. 2024. High-Dimensional Vector Quantization: General Framework, Recent Advances, and Future Directions. *IEEE Data Eng. Bull.* 48, 3 (2024), 3–19.

[3] Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. 2025. Practical and asymptotically optimal quantization of high-dimensional vectors in euclidean space for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data* 3, 3 (2025), 1–26.

[4] 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.

[5] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 14, 11 (2021), 1964–1978. http://www.vldb.org/pvldb/vol14/p1964-wang.pdf

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 used to compress vectors into compact, lossy representations—often at the expense of accuracy loss. Recent state-of-the-art quantization methods mitigate this loss in two ways: (i) assign a fixed number of quantization bits to every vector dimension to enable SIMD-friendly computation; and (ii) derive error bounds that support efficient re-ranking. However, binding the code budget to dimensionality constrains both flexibility and efficiency. To overcome this, we propose MRQ, a novel AKNN method that combines projection and quantization in a principled way. The key insight is that high-dimensional vectors concentrate most information in the early dimensions after projection. Leveraging this, MRQ quantizes only a low-dimensional, user-controllable *subset* of the projected vector, decouples the compression ratio with vector dimensionality. For the residual dimensions, instead of storing them, MRQ uses lightweight pre-computed statistics to estimate their distance contributions, further enhancing search efficiency. 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. Furthermore, MRQ exhibits high scalability, enabling up to 5× faster index construction than graph-based methods.

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 vector k-Nearest Neighbor (KNN) search in high-dimensional space is critical to many applications, including recommendation systems [52], information retrieval [43], and Retrieval-Augmented Generation for Large Language Models [41]. However, the curse of dimensionality [35] renders exact KNN search computationally expensive in high-dimensional space. This has led to growing interest in approximate KNN (AKNN) search [9, 11, 14, 17–19, 22, 25, 29, 32, 34, 37, 49, 53, 57, 62, 64, 66, 68, 70], which trades off a small amount of accuracy for substantially improved efficiency in large-scale settings.

Quantization. Vectors are typically represented in floating-point format, and storing them in memory at scale can be expensive. To address this, **quantization** has emerged as a widely adopted technique for AKNN search [30, 48], aiming to reduce storage overhead. Among various quantization approaches, most prominent methods are Binary Quantization [23, 28, 40, 42, 51], Product Quantization [1–3, 6, 12, 24, 39, 47, 58, 63] and scalar quantization [1, 69]. • **Binary Quantization (BQ)** encodes float vectors into binary codes and estimates distances between vectors via bit-wise operators, offering lightweight storage and fast comparison. • **Product Quantization (PQ)** partitions vectors into subspaces and quantizes vectors in each subspace using a pre-trained codebook. Distance is then computed via efficient look-up tables. • **Scalar Quantization (SQ)** converts vectors to a lower-precision data type, such as an 8-bit integer, thereby reducing the storage size. The distance estimation is accelerated by enabling faster numerical computation. BQ, PQ, and SQ all benefit from hardware acceleration—particularly SIMD instructions [1, 3, 21, 23]—which enables high-throughput distance computations and greatly improves AKNN search performance. However, a major drawback of vector quantization is *the considerable drop in search accuracy* caused by the lossy encoding of the original vectors. As highlighted in recent studies [20, 21, 54], it is challenging for these methods to achieve high recall with a very compact code (e.g., >90% recall with 128-bit codes).

The Challenge of High-Accuracy Quantization. To mitigate the accuracy degradation, a common strategy is to use re-ranking: retrieve a large set of candidates using approximate distances and then re-rank them with exact distances. However, this often requires manual tuning and incurs significant computational overhead. A more principled approach, pioneered by methods like RabitQ [23], introduces error-bound-based re-ranking. Instead of fetching a fixed number of candidates, these methods use statistical guarantees to prune candidates whose true distance cannot possibly be within the current top-k, thus avoiding many costly exact distance computations. This high accuracy, however, comes at a cost. To enable the calculation of tight error bounds, such methods often require the number of quantization bits to exactly match the dimensionality of the original vector. This requirement, while enabling principled re-ranking, is a common constraint in methods that estimate distances from quantized codes by preserving inner products or other geometric properties, as the estimation often relies on a direct correspondence between the code bits and the original dimensions of the vector. This leads to a *fixed compression ratio* (typically 32× for float32 vectors), which presents two major problems:

- **Memory Inflexibility:** Users cannot choose a higher compression ratio for memory-constrained scenarios. A 1TB dataset compressed 32× still requires 32GB of RAM, which may be unavailable.
- **Computational Inefficiency:** The code length is dictated by the vector dimensionality (e.g., 960, 420), which often does not align with hardware SIMD register widths (e.g., 128, 256, 512 bits). This leads to suboptimal performance, as additional instructions are needed to process a single code.

Our Solution. To summarize, existing quantization techniques either sacrifice search accuracy for space reduction or compromise on compression to preserve accuracy. A natural question arises: Can we maintain high search accuracy with principled re-ranking, without being constrained by a fixed compression ratio? We argue that a principled combination of projection and quantization can effectively address this challenge. Our key insight stems from an empirical analysis of high-dimensional vector embeddings: after applying an information-aware projection like PCA, the per-dimension variance follows a pronounced long-tailed distribution (see Fig. 2). This indicates that most of the information is concentrated in a low-dimensional *subspace*, which warrants differentiated treatment 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-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:

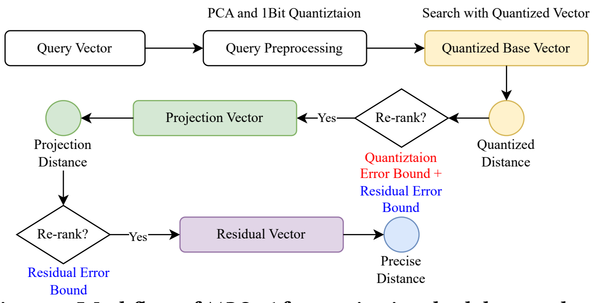


Figure 1: Workflow of MRQ. After projection, both base and query vectors are split into projected and residual components. MRQ performs distance computation in three stages: (1) The projected vectors are quantized with arbitrary bit lengths, enabling fast distance estimation with error bounds for re-ranking. (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 Flexible Quantization Mechanism. To address the limitation of the state-of-the-art method RabitQ, 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.

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 HNSW and the quantization-based RabitQ. Our method consistently outperforms existing approaches, achieving up to a 3× improvement in search efficiency while preserving comparable accuracy.

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 goal 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 [35], 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 [14, 15, 19, 22, 25, 26, 33, 34, 53, 54, 66], (2) quantization and inverted-file (IVF) methods [2, 6, 7, 23, 24, 28, 31, 39, 58, 65], (3) tree-based methods [5, 8, 13, 16], and (4) graph-based methods [5, 11, 17, 18, 32, 36, 44–46, 50, 55, 57, 59]. 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.

Many AKNN methods share a common component: a fixed-size result queue (typically implemented as a heap) to maintain the top-k results during the search process. 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 Vector Quantization

The core idea of quantization is to map continuous vectors into discrete codes. Among various techniques, product quantization (PQ) and binary quantization (BQ) are widely adopted due to their high compression ratios and efficient distance computation.

Product Quantization (PQ) partitions the original vector space into multiple low-dimensional subspaces—for example, a 128-dimensional vector may be divided into 32 subspaces of 8 dimensions each. Clustering methods, such as k-means, are then applied independently within each subspace. As a result, each vector can be approximated by the Cartesian product of the subspace centroids. During distance computation, PQ employs a lookup table that stores the distances from the query vector to all centroids. The approximate distance is then efficiently estimated by summing the corresponding centroid distances across subspaces.

Binary Quantization (BQ) encodes vectors into binary strings and approximates exact distances via binary operators during AKNN search. For example, Signed Random Projection [10] estimates angular similarity by generating binary codes via random projections. Iterative Quantization [28] formulates BQ as an optimization problem that minimizes the mean squared error between the original vectors and their binary codes using an orthogonal transformation. The recent RabitQ method [23] leverages spatial geometric properties to unbiasedly estimate vector inner products and refine distance estimation with error bounds.

Limitations. Both PQ and BQ face a fundamental trade-off: increasing the compression ratio (i.e., using fewer bits) typically leads to a significant drop in search accuracy. For example, achieving high recall on challenging datasets with a budget of just 128 or 256 bits remains an open problem for these methods. They rely on approximate distances, and improving accuracy usually requires retrieving a very large candidate set for re-ranking, which negates the speed benefits of quantization.

Remark. Both the PQ and BQ methods can benefit from hardware acceleration. For example, BQ supports bitwise operations, making it well-suited for instruction sets like SIMD. In contrast, PQ requires additional techniques such as 4-bit quantization [2, 3] to fit the entire lookup table into SIMD registers, which often results in increased quantization error. As we will show, MRQ is also designed for efficient SIMD execution.

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 [21–23, 36, 56]. 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.

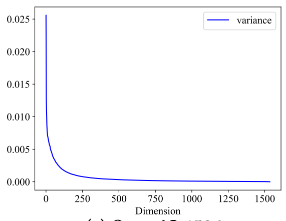
Error Bounds. Re-ranking incurs additional cost due to the need for many exact distance computations. To mitigate this, state-of-the-art methods, such as RabitQ [23], apply error bounds to prune unlikely candidates. By leveraging statistical properties of the quantization process, RabitQ can determine if the true distance of the current candidate is guaranteed to be larger than the current k-th farthest neighbor. If so, the candidate is safely pruned without an exact distance computation. However, as discussed in the Introduction, this powerful technique in RabitQ is coupled with the constraint that the code length must equal the vector dimension, limiting its applicability. Our work, MRQ, aims to preserve the benefits of error-bounded re-ranking while removing this restrictive constraint.

3 PROBLEM ANALYSIS AND OBSERVATION

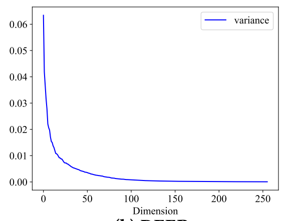
This section reiterates the core problem and presents our key observations that motivate the design of MRQ.

3.1 Problem Analysis

We begin our analysis with the SOTA quantization method, RabitQ. RabitQ introduced a re-ranking mechanism based on an unbiased estimator for the inner product. This estimator, $\langle \mathbf{x}, \mathbf{q} \rangle \approx \langle \bar{\mathbf{x}}, \mathbf{q} \rangle / \langle \bar{\mathbf{x}}, \mathbf{x} \rangle$, relies on computing the inner product between the quantized vector $\bar{\mathbf{x}}$ and the original vector $\mathbf{x}$. This formulation inherently requires the binary code of $\bar{\mathbf{x}}$ to have the same length as the dimensionality of $\mathbf{x}$. When $\mathbf{x}$ is stored in float32, this yields a fixed compression ratio of 32×. As motivated in the Introduction, this fixed ratio introduces challenges for both memory-constrained scenarios and computational efficiency due to potential misalignment with SIMD widths.



(a) OpenAI-1536



(b) DEEP

Figure 2: Variance Distribution of Vectors After PCA

3.2 Observation

To benefit from the significant space reduction offered by quantization methods while overcoming the limitation of RabitQ's fixed compression ratio, we present the following two key observations 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.** From an information-theoretic perspective, higher variance in a dimension indicates greater information content. Consequently, only a small number of dimensions after PCA are required to preserve most of the information in the original vector. This suggests that treating all dimensions equally during quantization is suboptimal.

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. From the perspective of reconstruction error, 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 vector quantization, we found that the residual part still contains useful information worth exploring. The vector norm of the residual part can be integrated into the overall vector norm to reduce the error. Secondly, the variance of the residual part can help us analyze the error distribution and bounds. In summary, this perspective clarifies why focusing on high-variance components is both information-efficient and reconstruction-aware, while also motivating targeted use of residual statistics for further refinement.

4 OUR QUANTIZATION METHOD

Based on the above observations, 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.

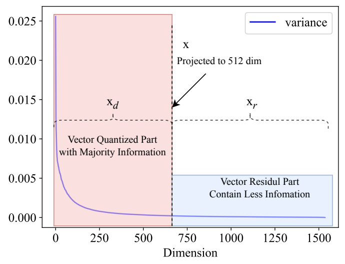


Figure 3: The Example of Vector Split and Quantization

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.

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., $\mathbf{x}_d = \mathbf{R}\mathbf{x}$. We then apply a quantization method (similar to RabitQ) to the projected vector $\mathbf{x}_d$. The exact squared Euclidean distance dis between a database vector $\mathbf{x}$ and a query vector $\mathbf{q}$ can then be expressed as:

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

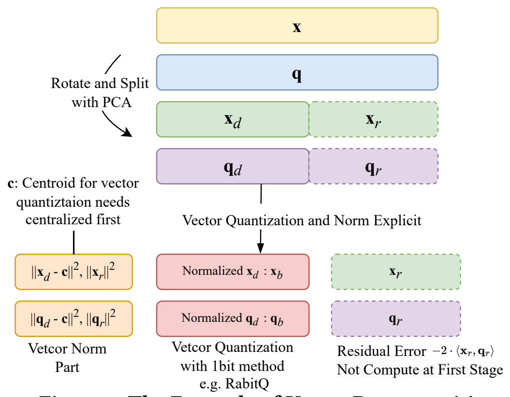
where $\mathbf{x}_r$ is the residual dimension ($r + d = D$). Similar to RabitQ, we normalize and center the vectors after projection. Let $\mathbf{c}$ denote the centroid of the projected data $\mathbf{x}_d$. The normalized projected vector is defined as $\mathbf{x}_b := \frac{\mathbf{x}_d - \mathbf{c}}{||\mathbf{x}_d - \mathbf{c}||}$, and the query vector is similarly normalized as $\mathbf{q}_b := \frac{\mathbf{q}_d - \mathbf{c}}{||\mathbf{q}_d - \mathbf{c}||}$. With this normalization, the projected distance $||\mathbf{x}_d - \mathbf{q}_d||^2$ can be expressed in terms of the inner product and vector norms. For the inner product term $\langle \mathbf{x}_d, \mathbf{q}_d \rangle$, we further apply a quantization method such as RabitQ to obtain a d-bit binary representation $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$.

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

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.

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

Example 4.1. Fig.4 illustrates an example of Equation 3. We first apply PCA to transform the original vectors $\mathbf{x}$ and $\mathbf{q}$, resulting in



$\mathbf{x}$
$\mathbf{q}$
Rotate and Split with PCA
$\mathbf{x}_d$
$\mathbf{x}_r$
$\mathbf{q}_d$
$\mathbf{q}_r$
$\mathbf{c}$: Centroid for vector quantiztaion needs centralized first
Vector Quantization and Norm Explicit
$||\mathbf{x}_d - \mathbf{c}||^2, ||\mathbf{x}_r||^2$
$||\mathbf{q}_d - \mathbf{c}||^2, ||\mathbf{q}_r||^2$
Normalized $\mathbf{x}_d : \mathbf{x}_b$
Normalized $\mathbf{q}_d : \mathbf{q}_b$
$\mathbf{x}_r$
$\mathbf{q}_r$
Vetcor Norm Part
Vetcor Quantization with 1bit method e.g. RabitQ
Residual Error $-2 \cdot \langle \mathbf{x}_r, \mathbf{q}_r \rangle$ Not Compute at First Stage

Figure 4: The Example of Vector Decomposition

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 $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$.

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: (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 $\langle \mathbf{x}_r, \mathbf{q}_r \rangle$.

Then, for the multi-stage computation, we first estimate the distance using the quantized inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ (details provided below). This approximation enables efficient filtering: if the estimated distance is already too large, we skip the candidate. Otherwise, we re-rank using the original projected inner product $\langle \mathbf{x}_d, \mathbf{q}_d \rangle$ (which can be directly computed). If even higher precision is required, we further compute $\langle \mathbf{x}_r, \mathbf{q}_r \rangle$ for exact distance calculation. We focus only on the first stage: using the quantized inner product $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$ for distance estimation and applying error bounds for re-ranking, as the second and third stages follow similar logic.

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 approach of RabitQ. Specifically, the ratio $\frac{\langle \bar{\mathbf{x}}_b, \mathbf{q}_b \rangle}{\langle \bar{\mathbf{x}} b, \mathbf{x}_b \rangle}$ serves as an effective unbiased estimator of $\langle \mathbf{x}_b, \mathbf{q}_b \rangle$, where $\bar{\mathbf{x}}_b$ is the quantized vector of $\mathbf{x}$ using a randomly rotated codebook $\mathcal{C}$rand. The codebook is defined as $\mathcal{C} := \{-\frac{1}{\sqrt{D}}, +\frac{1}{\sqrt{D}}\}^D$, and the random rotation is applied via an orthogonal matrix P, yielding $\mathcal{C}_{\text{rand}} := \{P\mathbf{x} \mid \mathbf{x} \in \mathcal{C}\}$.

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 omit the residual inner product and obtain the final approximate distance:

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

A key advantage of MRQ is that the computation of the quantized inner product $\langle \bar{\mathbf{x}}_b, \mathbf{q}_b \rangle$ is highly amenable to hardware acceleration. The binary codes $\bar{\mathbf{x}}_b$ can be processed using SIMD instructions (e.g., POPCNT on AVX2/AVX-512 registers), which are then converted to inner products. Crucially, the projection dimension d can be chosen to align with SIMD vector widths (e.g., 128, 256, 512 bits), ensuring optimal instruction throughput. This contrasts with methods where the code length is fixed to the original vector dimension, which may not be a multiple of the SIMD width, leading to 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 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 [23]:

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

where ϵ_0 is a tunable parameter controlling the failure probability, and c_0 is a constant [23]. *(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. 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 $\sigma_i$
1 Train PCA matrix $P_p$ with database vector set $S$;
2 Derive residual variance $\sigma_i$ 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 $\langle \bar{\mathbf{x}}, \mathbf{x} \rangle$;
```

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 implement 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

```
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 $\epsilon_0$ 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 $\sigma_i$
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 $\sigma$ by Equation (6) with $\mathbf{q}_r$ and $\sigma_i$;
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 $\epsilon_0$;                // 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;
```

precomputed values (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. Recall from Fig.1 that we adopt 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 incorporate stage 2: instead of using the quantized distance, we compute the original projected distance as

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

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 [22] to enhance cache efficiency.

5.3 Discussion

Parameter Selection. The projection dimension d directly affects the search performance. The choice of d mainly comes from the following two aspects. Specifically, with limited space constraints, the parameter d can be derived from the space budget to utilize resources fully. On the other hand, the projection dimension d can be set to achieve the best search efficiency. Since the overall search latency is determined by multi-stage distance computation, and each component heavily depends on the hardware settings, such as SIMD bandwidth. The straightforward method is to grid search the parameter d with a step size of 64, which may cost a lot with a large dataset. Fortunately, we provide an empirical formula as follows

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

to set d as the first $2^{i^\star}$ that accumulates 80% total variance and at least 128 for better SIMD operation to achieve the best search

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

performance. This method can achieve a near-optimal performance without explicitly building the index multiple times. We verify this experimentally in § 6.2.

Scale to Distribute System. To handle large-scale vector data, our method can be extended with a distributed system. Since our algorithm heavily relies on IVF partitioning, we can directly apply the MRQ and multi-stage distance computation to distributed IVF systems such as [27, 60, 67]. Furthermore, our pre-processing, such as PCA, can be trained on only a fraction of the total data (1 million slices), unconstrained by data scale. Future work will explore integration with diverse distributed vector indexes such as graphs.

6 EXPERIMENT

6.1 Experiment Setting

Datasets. We evaluate our method on eight 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 of our method 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 with the SOTA quantization method RabitQ;
- IVF-RabitQ$^+$: An optimized version of IVF-RabitQ where we apply the same memory layout optimizations as in IVF-MRQ$^+$.
- IVF-MRQ: IVF integrated with our MRQ method with only quantization for distance computation without memory layout and multi-stage optimization;
- IVF-MRQ$^+$: IVF-MRQ enhanced with multi-stage distance computations and optimized memory layout;
- IVF-OPQfsr: We also compare against the widely-used PQ or BQ methods within an IVF framework. We selected the most competitive 4-bit OPQ with *fastscan* SIMD [2, 38] optimized as the most competitive baseline from various options. Additional experiments can be found in our technical report [61].
- HNSW [46]: the most widely used graph-based method.

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 [38]. For the IVF-RabitQ baseline, the quantization bit length is set to match the original vector dimension, as required by its design. For all datasets except WORD2VEC and MSONG, this dimension is used directly. For WORD2VEC and MSONG, the vectors are zero-padded to lengths of 320 and 448, respectively, to facilitate more efficient SIMD operations. The quantization bit settings for MRQ are configured completely based on our empirical formula in § 5.3 as follows: 128 bits for the MSONG, DEEP, TINY, and GIST datasets; 256 bits for WORD2VEC; and 512 bits for OpenAI-1536, OpenAI-3072, and MSMARCO10M. The IVF-RabitQ+ uses the same projection dimension as MRQ$^+$ with PCA rotated vectors for fair comparison. For IVF-OPQsfr, we set the rerank range as 1000 from the best search performance setting report in [23]. For the baseline HNSW algorithm, we use M = 16 and efConstruction = 500, consistent with the recommended settings and recent benchmarks [4, 46].

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 with 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.

(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

[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

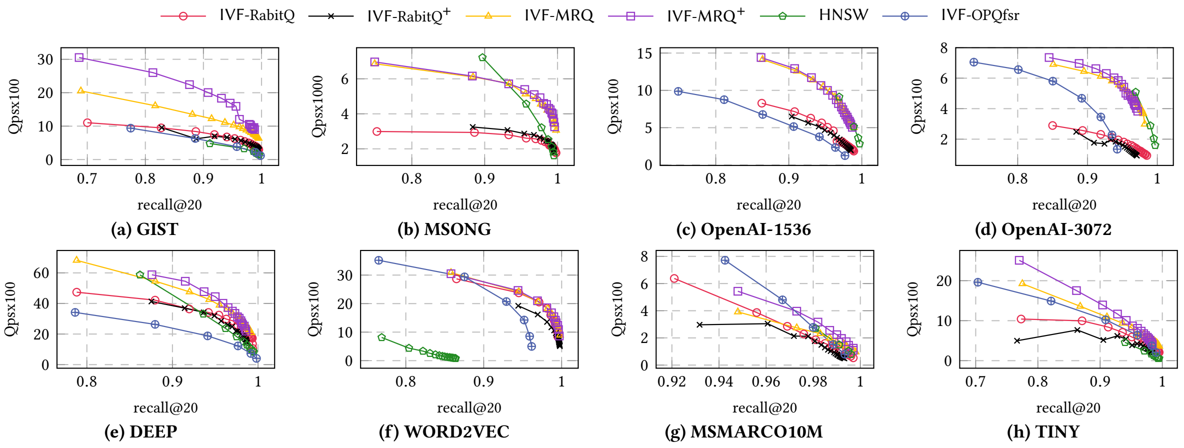


Figure 5: The Performance Test Among Various Method

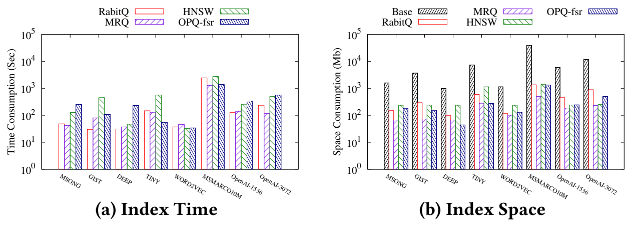


Figure 6: The Comparison of the Time Cost and Index Space

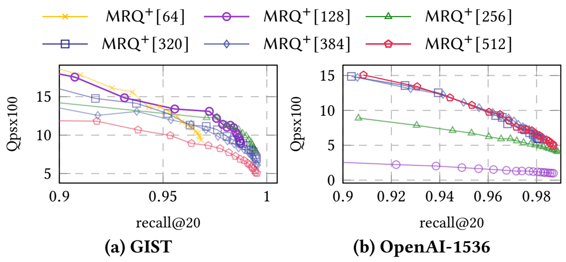


Figure 8: The Validation of Parameters Selection

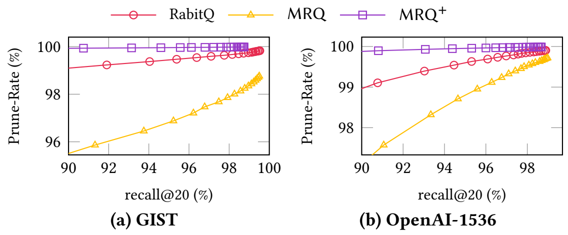


Figure 7: In-Depth Analysis of Our Methods

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.

Exp-3: In-depth Analysis We deeply analyze the efficiency of our algorithm by comparing the ratio of distance computation pruning to precise distance computations at different stages. Fig. 7 shows that RabitQ, with its error bounds, can prune >99% of precise distance computations, where the y-axis represents the prune rate of precise distance computation with origin vector. However, a fixed compression ratio makes it difficult to achieve higher efficiency with fewer bits. Furthermore, since RabitQ uses more bits, only a small portion of the distance computation requires higher precision. This results in limited speedup when combining multi-stage computation with RabitQ (RabitQ+). Unlike RabitQ, our algorithm, MRQ, uses only 1/8 to 1/3 the number of bits to prune 96%-98% of precise distance computations (losing only 1-2%) from Fig. 7. Furthermore, our multi-stage distance computation can compensate for the efficiency loss of fewer bits with higher-precision projection distances, further improving efficiency.

Exp-4: Parameter Select We experimentally verify our empirical parameter selection method in § 5.3. Specifically, we report the search performance under grid search with a step length of 64 in Fig. 8. From the variance distribution plot in Fig. 2, the GIST dataset needs the first 78 dimensions to accumulate 80% total variance, while OpenAI-1536 needs 354. Then the projection dimension d is set as 128 for GIST and 512 for the OpenAI-1536 dataset according to our empirical formula. The efficiency results in Fig. 8 show that the empirical setting (line with solid mark) can achieve near-optimal search performance. This empirical formula is also used in all tests in Fig. 5, providing low-cost parameter tuning guidance for efficiency-driven index design needs.

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 significantly improves search efficiency with negligible index construction time and storage overhead. Future work will explore external processing for large-scale vector datasets.

REFERENCES

[1] 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).

[2] 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).

[3] 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.

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

[5] 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.

[6] 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.

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

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] Jianyang Gao, Yutong Gou, Yuexuan Xu, Jifan Shi, Cheng Long, Raymond Chi-Wing Wong, and Themis Palpanas. 2024. High-Dimensional Vector Quantization: General Framework, Recent Advances, and Future Directions. *IEEE Data Eng. Bull.* 48, 3 (2024), 3–19.

[21] 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).

[22] 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

[23] 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.

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

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

[26] 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.

[27] 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.

[28] 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

[29] 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.

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

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

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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).

[37] 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.

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

[39] 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.

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

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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)*.

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

[49] 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.

[50] 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

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

[52] 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.

[53] 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.

[54] 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.

[55] 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.

[56] 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

[57] 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.

[58] 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.

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

[60] 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.

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

[62] 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.

[63] 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.

[64] 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.

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

[66] 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.

[67] 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).

[68] 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).

[69] 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.

[70] 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.

APPENDIX

Extra Discussion

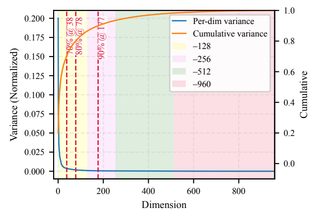
We further evaluate the data variance statistic of various source datasets. From the result in Fig. 9, we found that all vector embeddings from images, text, and audio align with our observation that the data, after being rotated by the PCA matrix, exhibits a long-tailed distribution. The cumulative variance of the tail dimension differs under various data distributions. Specifically, the WORD2VEC accumulates less than 80% of total variance using only the top 50% of its dimensions. However, the GIST dataset accumulates over 80% of total variance using only 10% of its dimensions. The variance distribution in Fig. 9 also corresponds to our empirical parameter settings formula 8. The projection dimension that accumulates the 80% total variance of GIST, MSONG, OpenAI-1536, OpenAI-3072, DEEP, WORD2VEC, MSMARCO, and TINY datasets is 128, 5, 354, 470, 46, 173, 273, and 51, respectively. According to our empirical formula, we set 128 bits for the MSONG, DEEP, TINY, and GIST datasets; 256 bits for WORD2VEC; and 512 bits for OpenAI-1536, OpenAI-3072, and MSMARCO10M to achieve the best search efficiency.

Extra Experiments

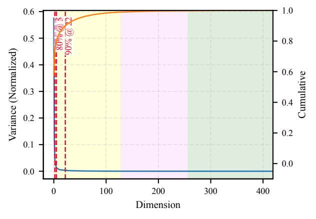
Exp-5: Study of Centroid Approximation. In Algorithm 1, we apply the k-means algorithm on the projected (approximate) vectors instead of the original full-dimensional vectors. This results in approximate centroids, as used in the IVF method. The effectiveness of this approach is shown in Fig. 10, where 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).

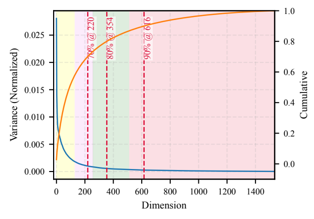
As shown in Fig. 10, the projected centroids incur almost no recall loss on the GIST dataset and even slightly outperform the original centroids on the OpenAI-1536 dataset. Therefore, we adopt the projected centroids in our method. Additionally, this projection significantly reduces the training time for the IVF index, making it more efficient for large-scale datasets.

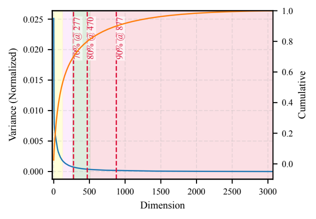
Exp-6: Compare with Various Quantization Methods We evaluate the performance of the base MRQ quantization—excluding multi-stage distance computation and memory layout optimizations—against fundamental techniques such as PQ, as shown in Fig.11. To ensure a fair comparison of memory usage, we configure PQ with a 32× compression ratio to match the space consumption of RabitQ. As illustrated in Fig.11, without re-ranking, standard PQ-based quantization fails to achieve a recall of 90% across various datasets. While high-compression regimes (like those used in MRQ) typically incur accuracy losses, we observe distinct advantages in our approach. When considering the re-ranking scenario, we compare MRQ against the state-of-the-art Binary Quantization (BQ) method, RabitQ, and the Optimized PQ method, OPQ-4bit-fastscan (denoted as OPQ-fsr). Experimental results demonstrate that MRQ significantly outperforms both RabitQ and OPQ-fsr on the vast majority of datasets. Notably, MRQ achieves this superior performance relying solely on quantized distance for re-ranking, even without the aid of multi-stage distance computation or memory layout optimizations.

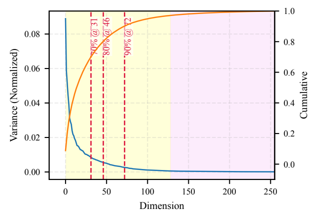


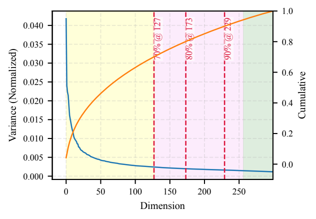
(a) GIST

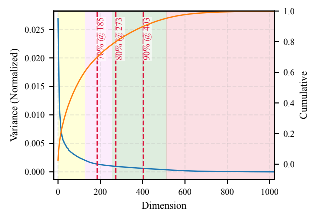

(b) MSONG


(c) OpenAI-1536


(d) OpenAI-3072


(e) DEEP


(f) WORD2VEC


(g) MSMARCO

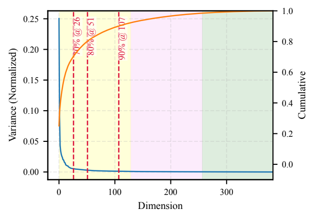
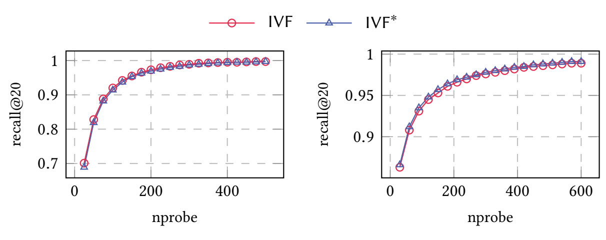

(h) TINY

Figure 9: Variance Distribution of Vectors After PCA



(a) GIST (b) OpenAI-1536

Figure 10: The Study of Centroid Approximation

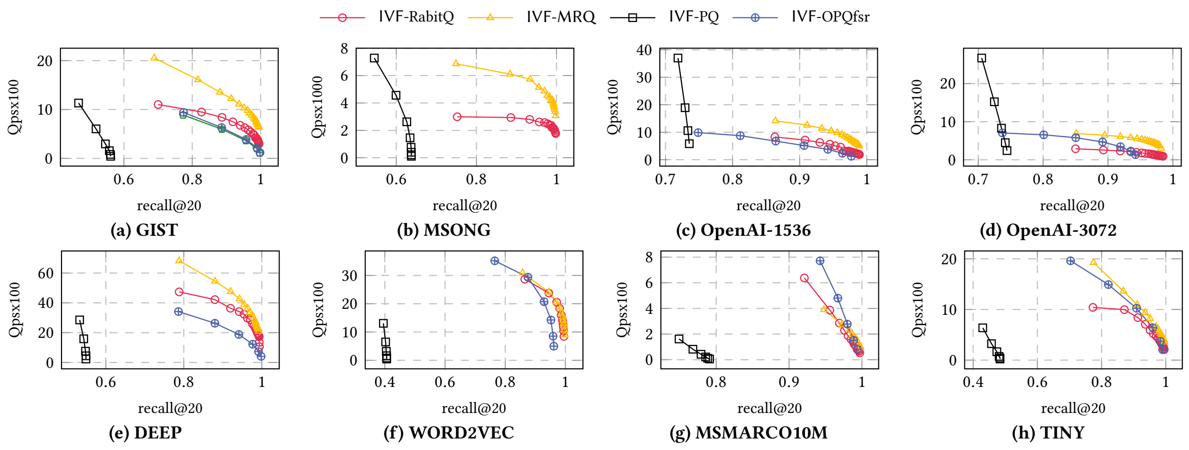


Figure 11: The Performance Test of Vairous Quantization Method