Effective and General Distance Computation for Approximate Nearest Neighbor Search

Mingyu Yang[1], Wentao Li[1], Jiabao Jin[2], Xiaoyao Zhong[2], Xiangyu Wang[2], Zhitao Shen[2], Wei Jia[2], and Wei Wang[1,3]
[1]The Hong Kong University of Science and Technology (Guangzhou); [2]Ant Group;
[3]The Hong Kong University of Science and Technolog
myang250@connect.hkust-gz.edu.cn; wentaoli@hkust-gz.edu.au;
{jinjiabao.jjb, zhongxiaoyao.zxy, wxy407827, zhitao.szt, jw94525}@antgroup.com; weiwcs@ust.hk

***Abstract*—Approximate K Nearest Neighbor (AKNN) search in high-dimensional spaces is a critical yet challenging problem. In AKNN search, distance computation is the core task that dominates the runtime. Existing approaches typically use approximate distances to improve computational efficiency, often at the cost of reduced search accuracy. To address this issue, the state-of-the-art method, ADSampling, employs random projections to estimate approximate distances and introduces an additional distance correction process to mitigate accuracy loss. However, ADSampling has limitations in both *effectiveness* and *generality*, primarily due to its heavy reliance on random projections for distance approximation and correction.**

Motivated by this, we leverage data distribution to improve distance approximation via orthogonal projection, thereby addressing the effectiveness limitation of ADSampling; we also adopt a data-driven approach to distance correction, decoupling the correction process from the distance approximation process, thereby overcoming the generality limitation of ADSampling. Extensive experiments demonstrate the superiority and effectiveness of our method. In particular, compared to ADSampling, our method achieves a speedup of 1.6 to 2.1 times on real-world datasets while providing higher accuracy. In addition, our method shows superior performance in Ant Group image search scenarios and has been integrated into their search engine.

***Index Terms*—Approximate Nearest Neighbor Search, Distance Computation, Data-Driven, Vector Databases**

I. Introduction

The problem of K Nearest Neighbor (KNN) search involves identifying the top-K data points in a database S that are closest to a query point q. KNN search is crucial in various domains, including information retrieval [1], data mining [2], recommender systems [3], and vector databases [4]. Effective solutions, such as R-trees, exist for KNN search in low-dimensional spaces. However, the curse of dimensionality [5] renders exact KNN search prohibitively time-consuming in high-dimensional spaces. Consequently, researchers have developed the approximate variant known as **Approximate K Nearest Neighbors (AKNN) search** [5], which is more suitable for real-time responses in large-scale data.

Given the critical role of AKNN search, numerous algorithms have been developed. These algorithms mainly fall into four categories: inverted file-based [6], [7], graph-based [8], [9], [10], [11], [12], [13], tree-based [14], [15], [16], and hash-based [17], [18], [19], [20], [21], [22] methods. To find the AKNN of a query point q in a database S, these AKNN algorithms can often be abstracted into a **candidate generation and refinement framework**: (1) Candidate generation: In this phase, a subset of points from S is selected to form a superset of the final AKNN. (2) Refinement: In this phase, the algorithm identifies the top points closest to q among the candidates, which are then returned as the AKNN.

The distinction between various AKNN algorithms primarily lies in the candidate generation phase, while the refinement phase is almost identical. In the refinement phase, a result queue Q, often implemented as a max-heap, is maintained to store the data points closest to the query point q, ultimately producing the final result. Specifically, for a candidate point p, if the distance to the query point q is less than the maximum distance τ recorded in Q, the result queue is updated; otherwise, the point is disregarded. Therefore, distance computation is crucial and computationally intensive during this phase. Notably, **distance computation is often the most time-consuming component of AKNN algorithms.** For example, in graph-based algorithms like HNSW [9], distance computation constitutes 80% of the total AKNN search time. In inverted file-based algorithms such as IVF [7], it accounts for 90% of the total time cost [23]. Thus, accelerating distance computation is essential for expediting AKNN search.

The State-of-the-art. To accelerate distance computation, ADSampling [23] was introduced as a general plug-in. ADSampling first estimates an (initial) approximate distance between two points by random projection and obtains an error bound based on the random projection matrix used. The advantage of ADSampling lies in its ability to use error bounds for an additional distance correction process: it corrects the approximate distance by error bounds, and analyzes whether the use of corrected approximate distances is sufficient in AKNN search [24], [23]. If not, more accurate distances are calculated, and an incremental correction is applied to the approximation (until an exact distance is computed). ADSampling achieves a good balance between speed and accuracy in AKNN search by the correction process, and experimental results confirm its efficiency. Yet, there is still much room for improvement in its effectiveness and generality.

Effectiveness. ADSampling employs a projection method to approximate distances. Specifically, ADSampling utilizes a

random projection matrix to compute these approximate distances. However, within projection methods, **random projection typically incurs a larger error than optimal orthogonal projection** between approximate and exact distances. It is important to note that if the approximate distance is sufficiently accurate, ADSampling can avoid the need for the more time-consuming incremental distance correction process. Therefore, a more accurate approximate distance is crucial.

Generality. The error bound is critical for ADSampling, as it informs whether the current (corrected) approximate distances are sufficient for the refinement phase of AKNN search. Yet, deriving the error bound for ADSampling is determined by the random projection matrix employed in the distance estimation process. **This dependence constrains the applicability of** ADSampling to scenarios where distances are approximated by techniques other than random projection.

Our Idea. The limitation of ADSampling in both effectiveness and generality stems from its over-reliance on random projections. This dependency limits its potential as it overlooks the underlying properties of the data points in the database S — ADSampling relied only on the projection matrix, not the database itself, for distance correction. This raises a natural question: Can we develop a new method for distance computation that not only provides the error bounds for distance correction to maintain accuracy, but also optimizes efficiency by exploiting the properties of the database S?

In this paper, we provide an affirmative answer to this problem. First, we investigate the cause of the limited effectiveness of ADSampling. By decomposing the exact distance into approximate distance and estimation error, we prove that the optimal orthogonal projection, when applied to database points, minimizes the estimation error. This indicates that the random projection used by ADSampling is suboptimal. We also analyze the distribution of estimation errors, and by introducing reasonable assumptions, we derive new error bounds for distance correction, which are shown to be minimized.

Furthermore, we introduce a novel data-driven distance correction scheme to accommodate the approximate distances generated by *arbitrary* distance approximation methods. Our method is unique in that it makes no assumptions about the source of these approximate distances. Instead, it parameterizes the error bound used in the distance correction process and learns this parameter directly from the data, adopting a data-driven approach. This flexibility allows our method to adapt to any approximate distance methods, such as those obtained from product quantization (PQ) [6], thereby providing a level of generality that ADSampling lacks.

Contributions. We summarize our contributions as follows:

Analysis of the SOTA Method (§ III). We introduce the state-of-the-art distance computation method, ADSampling, and analyze its limitations. This analysis motivates us to propose more effective and general methods for distance computation.

More Effective Distance Estimation (§ IV). To address the limited effectiveness of ADSampling, we decompose the exact distance into the approximate distance and the estimation error. We demonstrate that using optimal orthogonal (i.e., PCA) projections instead of random projections results in minimal estimation error. Consequently, we replace the random projections used in ADSampling with PCA projections to improve the accuracy of distance estimation. In addition, by analyzing the distribution of estimation errors and assuming a Gaussian distribution, we derive the corresponding optimal error bounds for distance correction.

More General Distance Correction (§ V). To accommodate non-projection-based distance estimation methods (such as PQ [6]), we propose a data-driven distance correction scheme. This scheme parameterizes the error bound used for distance correction and employs a data-driven approach to learn the appropriate parameter for a given database. By determining this parameter/error bound using the learning-based technique, the proposed distance correction scheme avoids making assumptions about the source of the approximate distance, thus achieving the generality that ADSampling lacks.

Extensive Experimental Analysis (§ VII). We have conducted extensive experiments on a large number of real-world datasets, ranging from 1 million to 100 million entries, to validate our method. The experimental results demonstrate that the proposed method improves the search efficiency of existing AKNN algorithms by 1.6 to 2.1 times, significantly outperforming ADSampling. Furthermore, our algorithm is scalable to larger datasets, further confirming its efficiency.

Related Work. In the area of speeding up distance computations between points or vectors, two prominent methods are ADSampling [23] and FINGER [25]. ADSampling uses the Johnson-Lindenstrauss (JL) lemma [26] to establish a probabilistic bound between approximate and exact distances to speed up distance computations. This method, based on a solid mathematical foundation, is widely applicable to various AKNN algorithms. On the other hand, FINGER [25] is specifically designed for graph-based algorithms such as HNSW. While FINGER shows empirical success in improving search speed over HNSW [9], it comes with increased index construction time and higher space requirements. Thus, in this paper we focus primarily on improving the performance of ADSampling, while providing a comparative analysis of FINGER in § VII.

Data-driven approaches have greatly advanced the field of AKNN search. In particular, recent work [27], [28] uses learning-based techniques to predict the next node during graph traversal, allowing for more efficient navigation of the search space. Other methods [29], [30] employ learning-based strategies to estimate the difficulty of the AKNN search, allowing early stops to improve the efficiency. In contrast, our work focuses on optimizing distance computations, making it compatible with various AKNN algorithms. By addressing the bottlenecks associated with distance computation, our work provides a comprehensive solution to improve the efficiency of the entire AKNN search process.

Due to space limitations, some proofs and experiments have been omitted and can be found in the technical report [31].

TABLE I: A Summary of Notations

Notation	Description
S	A set of points/vectors
D	The dimensionality of S
$\mathbf{R}$	The projection (rotation) matrix
$\mathbb{R}^d$	d-dimensional Euclidean space
dis, dis'	Exact and approximate distance
$\|u, v\|$	The Euclidean distance between u and v
τ	The distance threshold in the queue Q
ϵ	The estimation error
L	The linear classifier

II. Preliminaries

§ II-A presents the AKNN search problem and its associated AKNN algorithms. Then, § II-B discusses the issue of distance computation, an essential component of AKNN search.

A. The AKNN Search

Given a dataset S containing n points/vectors in D-dimensional space, i.e., $S = \{p_1, p_2, \ldots, p_n\}$, where $p_i \in \mathbb{R}^D$, we use the squared Euclidean distance[1] to compute the distance $dis(p, q)$ between two points p and q, where $dis(p, q) = \|p - q\|^2$. The time complexity of computing $dis(p, q)$ is $O(D)$ by scanning each dimension sequentially.

The problem of **K Nearest Neighbor (KNN)** search is to find the data points in S that have the top-K smallest distances to a query point $q \in \mathbb{R}^D$. Due to the complexity of KNN search, a relaxed version of the problem, known as **Approximate K Nearest Neighbors (AKNN)** search, has been proposed. Given a query point q, AKNN search allows the returned points to be close to, but not necessarily the exact, K closest points to q. This approach sacrifices some accuracy in favor of improved computational efficiency.

Note that there are other widely adopted distance metrics, such as cosine similarity and inner product, which can be transformed into Euclidean distance through simple transformations [23]. Therefore, our discussion will focus solely on AKNN search under the Euclidean distance metric. Table I summarizes the commonly used notations.

AKNN Algorithms. Currently, AKNN algorithms can be mainly divided into four categories: inverted file-based [7], [6], graph-based [9], [10], [11], [32], [13], [33], [12], tree-based [14], and hash-based [17], [18], [20], [19], [21], [22].

Inverted File-Based. Inverted file-based algorithms, such as IVF [7], are often used to speed up AKNN search. The core idea of IVF is to cluster the points in a data set S into multiple clusters, which helps to speed up the search process. During the **indexing** phase, IVF uses the k-means algorithm to cluster the data points in S. It then constructs a bucket for each cluster and assigns the data points within that cluster to the corresponding bucket. In the **query** phase, given a query point q, IVF first selects the nearest top-N^{Probe} clusters based on the distance from q to the cluster centroids. It then retrieves all data points in the corresponding buckets of these nearest clusters as candidates and identifies the K nearest neighbors among these candidates.

Graph-Based. Graph-based algorithms for AKNN search construct a navigable graph where nodes represent data points and edges connect nodes that are nearest neighbors. Hierarchical Navigable Small World (HNSW) [9] is a prime example of such algorithms, known for its superior search speed and accuracy. During the **indexing** phase of HNSW, data points are inserted into a multi-layered graph structure, with each layer representing data at increasingly fine-grained levels. Each point is connected to a fixed number of closest neighbors, ensuring each layer maintains a navigable small-world network property. In the **query** phase, the search begins from the top layer, leveraging the hierarchical small-world structure to efficiently navigate towards the region closest to the query point. Upon reaching the base layer, the algorithm navigates precisely through the neighborhood graph to identify the approximate nearest neighbors to the query point.

Others. Other AKNN algorithms include tree-based and hash-based methods. Tree-based methods use tree structures to route queries and identify candidate points, while hash-based methods generate candidate points through hash codes and then determine the AKNN from these candidates. Yet, in practice, these methods are less appealing than inverted file- and graph-based methods due to performance limitations.

B. Existing Distance Computation Methods

The time to compute distances dominates the runtime of AKNN search, accounting for 80% of the time complexity in IVF and 90% in HNSW. To improve efficiency, an intuitive idea is to use approximate distances instead of exact distances for the refinement phase of the AKNN search. Two types of methods are proposed for computing approximate distances: projection and quantization.

Projection. Projection methods, such as random projection, map high-dimensional data to a lower-dimensional space, mitigating the curse of dimensionality and facilitating efficient data processing and storage. Typically, dimensionality reduction is achieved by multiplying the points in the original space by an orthogonal projection matrix. The advantage of projection methods is their relative ease of implementation, which allows for the processing of large, high-dimensional datasets in a comparatively short amount of time.

Quantization. Quantization methods, such as Product Quantization (PQ) and Residual Quantization (RQ), map the original vector into discrete short quantized codes with a pre-trained codebook. Unlike projection methods, which compute distance in low-dimensional space, quantization computes the distance between the query vector and base quantized codes as the approximate distance. Due to its quantized nature and the use of a codebook, quantization can speed up distance computation via distance look-up.

Remark. Both projection and quantization methods can speed up the computation of distances. However, using approximate distances as a direct substitute for exact distances in the refinement phase of ANN algorithms (without distance correction)

[1]Squaring does not affect the order of distances.

can result in reduced search accuracy. For example, none of the quantization methods achieve more than 60% recall without re-ranking [34], [23]. To illustrate, consider the scenario where $\mathrm{K} = 1$ and we want to find the nearest neighbor of a query point q. If the approximate distance of any candidate point p to query q is less than the approximate distance of the true nearest neighbor of q, we can not return an exact result.

III. Problem Analysis

To mitigate the loss of accuracy often observed when directly integrating approximate distances in AKNN algorithms, ADSampling has been introduced as an optimization technique for distance computations. The key idea behind ADSampling is not only to leverage approximate distances but also to incorporate an error bound for correction purposes. This treatment also allows ADSampling to determine if the corrected approximate distance is adequate for the refinement phase of the AKNN search. When the current approximate distance falls short, ADSampling triggers more precise distance computations to compensate for the deficiencies. By incorporating these incremental computations for correction, ADSampling enhances the overall accuracy of AKNN search.

Distance Estimation. ADSampling first employs random projection to reduce the dimensionality from D to d for points, thereby estimating an approximate distance dis' for the exact distance dis. The relationship between any pair of approximate and exact distances is given by the following lemma:

Lemma 1 ([23]). *For a given point $x \in \mathbb{R}^D$, a random projection $P \in \mathbb{R}^{d \times D}$ preserves its Euclidean norm with a multiplicative error ϵ bound with the probability of*

$$\mathbb{P}\left\{\left|\sqrt{\frac{D}{d}}\|P\mathbf{x}\| - \|\mathbf{x}\|\right| \leq \epsilon\|\mathbf{x}\|\right\} \geq 1 - 2e^{-c_0 d\epsilon^2} \quad (1)$$

From Lemma 1, it follows that the error between the approximate distance dis' and the exact distance dis is bounded by $\epsilon \cdot dis$ with a small failure probability ($2e^{-c_0 d\epsilon^2}$).

Distance Correction. ADSampling improves the accuracy of AKNN search by leveraging error bounds between approximate and exact distances for correction. Then, ADSampling employs a hypothesis test to determine whether the (corrected) approximate distance between candidate points and the query point is sufficient to exclude candidates during the refinement phase. That is, if $dis' > (1+\epsilon) \cdot \tau$, or $dis' - \epsilon \cdot \tau > \tau$, where τ is the maximum distance (threshold) in the queue Q, ADSampling infers that $dis > \tau$ at a given significance level $p = 2e^{-c_0\epsilon_0^2}$. Here, ϵ_0 is a parameter that requires empirical tuning. In this case, excluding candidate points from Q based on the corrected approximate distance $dis' - \epsilon \cdot \tau$ is reliable.

Conversely, if the approximate distance does not satisfy the exclusion condition, it is insufficient to determine whether a candidate point should be removed from the queue Q. In such cases, ADSampling requires using additional dimensions to refine the distance estimation. This process entails calculating a more accurate approximate distance to decisively conclude whether $dis > \tau$ or $dis \leq \tau$. The correction procedure continues incrementally until all dimensions have been sampled and the final accurate distance is obtained.

Limitations. While ADSampling demonstrates superior performance compared to methods relying solely on approximate distances, it also presents two notable limitations:
(1) Limited Effectiveness. ADSampling employs random projection to compute approximate distances. However, among various projection methods, random projection does not ensure the minimization of error between approximate and exact distances. This discrepancy suggests that the approximate distances derived from random projection may significantly deviate from the exact distances. Notably, ADSampling requires incremental calculations of approximate distances until it can conclusively determine whether to exclude a candidate point. Therefore, finding a way to enhance the accuracy of approximate distance estimation could enable ADSampling to stop calculating distances for a candidate point earlier, thus accelerating the computation process.
(2) Lack of Generality. The error bound provided by Lemma 1 is applicable only to scenarios where the projection matrix is random. This limitation underscores the absence of a more general distance correction scheme that can adapt to other approximate distances, such as quantization distances. Developing a generalized distance correction scheme could potentially improve the efficiency and applicability of ADSampling, especially when other approximate distances are shown to be more effective than those derived from random projection [24].

IV. An Improved Projection-Based Distance Computation

This section primarily addresses the first limitation of ADSampling: the limited effectiveness of distance estimation based on random projection. We demonstrate that using optimal orthogonal projection, rather than random projection, leads to more effective distance estimation. Additionally, we propose a corresponding distance correction method tailored to this newly developed distance estimation technique.

A. Distance Decomposition

We investigate which projection matrices produce approximate distances that are close to the exact distances. To achieve this, we decompose the exact distance into its corresponding approximate distance and the estimation error introduced by the projection or rotation. By minimizing this estimation error, we can determine the optimal projection matrix that provides the most accurate projection-based distance estimation. The discussion of non-projection-based distance estimation is left to the next section.

Let $\mathbf{x}$ and $\mathbf{q}$ represent the D-dimensional data vectors and query vector, respectively. We consider a simple model where data vectors $\mathbf{x}$ are randomly sampled from the dataset following an unknown fixed distribution U. To capture global transformations, we introduce a rotation parameterized by the matrix $\mathbf{R}$. The transformed (rotated) vectors are denoted as $\mathbf{x}_D = \mathbf{R}\mathbf{x}$ and $\mathbf{q}_D = \mathbf{R}\mathbf{q}$. We partition the rotated data vector $\mathbf{x}_D$ into two components: $\mathbf{x}_d$, consisting of the first d **projected dimensions**, and $\mathbf{x}_r$, which represents the **residual dimensions**. Similarly, we decompose the query vector as $\mathbf{q}_D = (\mathbf{q}_d, \mathbf{q}_r)$. The exact distance between the transformed vectors can then be expressed as a decomposition:

$$\begin{aligned}\|\mathbf{x}-\mathbf{q}\|^2 = \|\mathbf{x}_D-\mathbf{q}_D\|^2 &= \|\mathbf{x}\|^2 + \|\mathbf{q}\|^2 - 2\cdot\langle\mathbf{q},\mathbf{x}\rangle \\ &= \|\mathbf{x}_d\|^2 + \|\mathbf{q}_d\|^2 + \|\mathbf{x}_r\|^2 + \|\mathbf{q}_r\|^2 \\ &\quad - 2\cdot(\langle\mathbf{q}_d,\mathbf{x}_d\rangle + \langle\mathbf{q}_r,\mathbf{x}_r\rangle).\end{aligned} \tag{2}$$

Let $C_1 = \|\mathbf{x}_d\|^2 + \|\mathbf{q}_d\|^2 + \|\mathbf{x}_r\|^2 + \|\mathbf{q}_r\|^2$ and $C_2 = -2\cdot\langle\mathbf{q}_d,\mathbf{x}_d\rangle$. For the C_1 term, $\|\mathbf{x}_d\|^2 + \|\mathbf{x}_r\|^2$ can be precomputed and stored, while $\|\mathbf{q}_d\|^2+\|\mathbf{q}_r\|^2$ only needs to be computed once for each query. The C_2 term, $-2\cdot\langle\mathbf{q}_d,\mathbf{x}_d\rangle$, can be calculated with an $O(d)$ cost. Thus, with $O(d)$ computation, the approximate distance can be computed as $dis' = C_1 - C_2$, with the **estimation error** term compared to the exact distance being $\epsilon = -2\cdot\langle\mathbf{q}_r,\mathbf{x}_r\rangle$.

B. An Improved Distance Estimation

Equation 2 presents the estimation error, expressed as $\epsilon = -2\cdot\langle\mathbf{q}_r,\mathbf{x}_r\rangle$, which captures the difference between the exact and approximate distances. Assuming the data vector follows a Gaussian distribution[2], i.e., $\mathbf{x} \sim \mathcal{N}(0,\Sigma)$ for a given query $\mathbf{q}$, the distribution of the estimation error can be viewed as a linear combination of multiple Gaussian distributions. Under this assumption, we show why random projection yields suboptimal results.

Which Projection is Better? We aim to determine which projection matrix provides the most accurate distance estimation. Under the constraint of orthogonal projection (which preserves distance after rotation), we plan to minimize the variance in estimation error. Let σ_i^2 denote the variance of the i-th dimension in the distribution U (where the data vectors are assumed to follow the distribution U). Upon receiving a query, the variance of the inner product for the residual dimension is given by $\sigma_i^2 q_i^2$. Therefore, the variance of the estimation error term can be computed as follows:

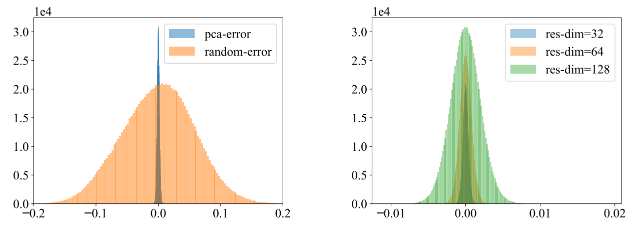
$$Var(-2\cdot\langle\mathbf{q}_r,\mathbf{x}_r\rangle) = 4\cdot\sum_{i=d+1}^{i\le D}(\mathbf{q}_i\sigma_i)^2 \tag{3}$$

We are ready to show that the principal component projection matrix, as opposed to the random projection matrix used in ADSampling, results in the smallest variance in the estimation error term.

Theorem 1. *Given a set of vectors S, the Principal Component Analysis (PCA) projection matrix maximizes the variance along the projected dimensions while simultaneously minimizing the variance in the residual dimensions. This optimization is achieved over all possible orthogonal projection matrices.*

The PCA projection matrix is well suited for distance estimation as it maximizes the projection variance, allowing the approximate distances to capture more information and thus become closer to the exact distances. Theorem 1 further shows that PCA projection minimizes the variance in the residual dimensions, effectively reducing the estimation error as much as possible. To gain deeper insight into Theorem 1, we analyze the distribution of the estimation error terms using real data sets and compare different projection matrices. Specifically, for the DEEP1M dataset (256 dimensions) and a given query q, we plot the distribution of $\langle\mathbf{q}_r,\mathbf{x}_r\rangle$ in Fig. 1.

[2]The data vectors have been centralized to have a mean of zero.



(1) Comparing PCA and Random (2) Varying dim of PCA Projection

Fig. 1: The Distribution of Estimation Error

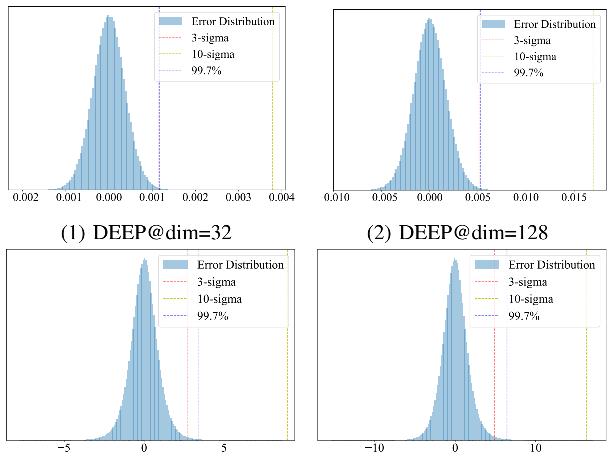
As shown in Fig. 1.1, the PCA projection matrix with a residual dimension of 128 has a more concentrated distribution compared to the random projection, which can be attributed to the smaller variance. Also, as shown in Fig. 1.2, the error associated with the PCA projection gradually converges to zero as the projection dimensions increase and the residual dimensions decrease. This shows that the PCA projection matrix is more effective than the random projection matrix in reducing estimation errors.

Remark. Equation 3 assumes that each dimension of $\mathbf{x}$ is linearly independent. In our implementation, we address this requirement by applying PCA matrix to align data vectors. This treatment reduces the off-diagonal elements of the covariance matrix to zero, ensuring independence among dimensions.

C. An Improved Distance Correction

The effectiveness of ADSampling stems from leveraging Lemma 1, which provides an error bound between exact and approximate distances, for distance correction. However, Lemma 1 is limited to scenarios where the projection matrix is random, thereby excluding the use of optimal (PCA) projections. Fortunately, Equation 2 also quantifies the error between exact and approximate distances, which can be modeled as a random variable. We then propose using the error quantile (e.g., the 99.5% quantile) of this random variable to establish an error bound, which can be efficiently derived from the inverse of its Cumulative Distribution Function (CDF).

Error Quantile. To establish an error bound for PCA projections, we analyze the distribution of the estimation error, denoted as $\epsilon = dis' - dis$. This error can be further expressed as $\epsilon = -2\cdot\langle\mathbf{q}_r,\mathbf{x}_r\rangle$, as shown in Equation 2. Here, we treat ϵ as a random variable, and note that its distribution follows a Gaussian distribution, i.e., $\epsilon \sim \mathcal{N}(0,\sigma^2)$, where the variance σ^2 is computed using Equation 3.



(1) DEEP@dim=32 (2) DEEP@dim=128 (3) GLOVE@dim=50 (4) GLOVE@dim=100

Fig. 2: The Empirical Analysis of the New Error Bound

Since the error ϵ follows a Gaussian distribution, the value of ϵ corresponding to specific quantiles can be calculated. This allows us to predefine a probability (i.e., quantile) and determine the corresponding value with a given probability as the error bound. For example, by setting the error bound to be three standard deviations from its mean, we can achieve a 99.7% quantile guarantee, as dictated by the empirical rule for Gaussian distributions. Conversely, for a given quantile, the corresponding error bound can be expressed as $m \cdot \sigma$, where m is the multiplier derived from the quantile.

After deriving the error bound, similar to ADSampling, we use this bound to correct the approximate distance to $dis' - m \cdot \sigma$ during the refinement phase. Also, if $dis' - m \cdot \sigma > \tau$, where τ is the maximum distance (threshold) for a queue Q, we can conclude that $dis > \tau$ with a corresponding quantile/probability related to $m \cdot \sigma$. This ensures that excluding candidate points from the queue Q is a reliable decision. Instead, if the condition fails, excluding a point from the Q queue is not sufficient. In such cases, additional dimensions must be sampled for the next round of distance correction.

However, the PCA projection minimizes the variance of the error, rather than directly minimizing the error corresponding to a given quantile. We then prove in Lemma 2 that minimizing the error variance also minimizes the error quantile. This shows that our approach is optimal under the Gaussian distribution, making it highly effective in practice.

Lemma 2. *Assuming a Gaussian distribution, minimizing the variance of the error also minimizes the corresponding quantile of the error.*

Empirical Analysis. To gain a deeper understanding of our new error bound, we analyzed real-world datasets and recorded the empirical error distribution for the DEEP and GLOVE datasets. The results, illustrated in Fig. 2, show error distributions with projected dimensions set to 32 and 128. In the DEEP dataset, the empirical error distribution aligns well with the Gaussian distribution: our Gaussian Empirical Rule ($\mu \pm 3\sigma$, depicted by the red line) corresponds closely to the 99.7th percentile of the observed dataset (indicated by the blue line). In contrast, the bound provided by ADSampling ($\mu \pm 10\sigma$, represented by the yellow line) diverges significantly from the 99.7th percentile (purple line). For the GLOVE dataset, despite minor deviations between our bound and the 99.7th percentile, this gap can be effectively managed using the learning-based approach discussed in § V.

Algorithm 1: DDC$_{\text{res}}$ algorithm

```
Input: Threshold $\tau$, Multiplier $m$, Project dim $d$,
       Transformed query $\mathbf{q}$, Transformed data $\mathbf{x}$
Output: Result: 0 with precise distance $dis$ or 1 with
        approximate distance $dis'$
1 $C_1 \leftarrow \|\mathbf{x}\|^2 + \|\mathbf{q}\|^2$; // Precompute Once
2 $C_2 \leftarrow 2 \cdot \langle \mathbf{x}_d, \mathbf{q}_d \rangle$; // Compute On the Fly
3 $\sigma \leftarrow \sqrt{4 \cdot \langle \mathbf{q}_r^2, \sigma_r^2 \rangle}$; // Precompute Once
4 if $C_1 - C_2 - m \cdot \sigma > \tau$ then
5     return 1 with $dis' = (C_1 - C_2)$;
6 else
7     $C_3 \leftarrow 2 \cdot \langle \mathbf{x}_r, \mathbf{q}_r \rangle$; // Compute On the Fly
8     return 0 with $dis = (C_1 - C_2 - C_3)$;
```

D. Implementation

We implement the proposed distance estimation and correction techniques that make up our distance computation method. Also, we propose an optimization to improve this method.

A Basic Method. Given a query vector $\mathbf{q}$ and a data vector $\mathbf{x}$ after the PCA projection, our novel distance computation approach DDC$_{\text{res}}$ is detailed in Algorithm 1. We begin by calculating C_1, defined as $\|\mathbf{x}\|^2 + \|\mathbf{q}\|^2$, in Line 1. Next, we determine C_2 in Line 2, based on the projected dimension d. For the residual dimensions, we pre-compute σ_i^2 for each data vector along dimension i, followed by computing σ as the standard deviation of the error (Line 3). We then estimate the approximate distance, denoted as $dis' = C_1 - C_2$, and apply a correction using $m \cdot \sigma$. If the corrected distance exceeds the threshold τ of the queue, the candidate is pruned, and we return 1 (Line 4-5). If not, we compute the exact distance and return 0 (Line 6-8).

Optimization. One of the advantages of orthogonal projection is its ability to incrementally increase the projected dimensions until all dimensions are used to obtain an exact distance. Inspired by ADSampling, we adopt the incremental correction to form the optimized algorithm in Algorithm 2. Specifically, if the current corrected distance, denoted as $C_1 - C_2 - m \cdot \sigma$, is sufficient for pruning, we can return immediately (Line 6–7). Otherwise, we increase the projected dimension by Δ_d and continue the correction process incrementally (Line 8–9), rather than computing the exact distance directly.

Example 1. *Fig. 3 explains how our methods work. First, we apply $m \cdot \Delta_0$ to correct the current approximate distance dis'_0. If this corrected distance does not exceed the queue threshold τ, it indicates that the candidate cannot be pruned.*

Algorithm 2: Incremental-DDC$_{\text{res}}$

```
Input: Threshold $\tau$, Multiplier $m$, Incremental Project dim $\Delta_d$, Transformed query $\mathbf{q}$, Transformed data $\mathbf{x}$
Output: Result: 0 with precise distance $dis$ or 1 with approximate distance $dis'$
1 $C_1 \leftarrow \|\mathbf{x}\|^2 + \|\mathbf{q}\|^2$; // Precompute Once
2 while $d < D$ do
3     $C_2 \leftarrow C_2 + 2 \cdot \langle \mathbf{x}_{\Delta_d}, \mathbf{q}_{\Delta_d} \rangle$; // Incremental
4     $r \leftarrow D - d$;
5     $\sigma \leftarrow \sqrt{4 \cdot \langle \mathbf{q}_r^2, \sigma_r^2 \rangle}$;
6     if $C_1 - C_2 - m \cdot \sigma > \tau$ then
7         return 1 with $dis' = (C_1 - C_2)$;
8     else
9         $d \leftarrow d + \Delta_d$;
```

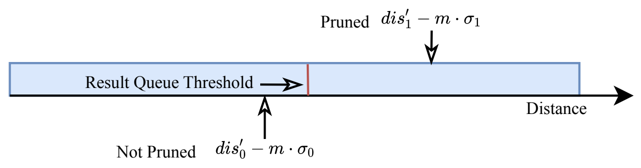

Pruned $dis_1' - m \cdot \sigma_1$
Result Queue Threshold
Distance
Not Pruned $dis_0' - m \cdot \sigma_0$

Fig. 3: The Example of How Our Methods Work

Algorithm 1 will then compute the exact distance. In contrast, Algorithm 2 proceeds by incrementing the projected dimension and recomputing the corrected distance as $dis_1' - m \cdot \delta_1$. If this recomputed distance exceeds τ, the candidate can be pruned.

V. A General Distance Computation

This section addresses the lack of generality associated with ADSampling. In the previous section, we proposed a new method, DDC$_{\text{res}}$, by replacing the random projection in ADSampling with a PCA projection. However, both ADSampling and DDC$_{\text{res}}$ are limited to projection-based distance estimation. Note that other distance estimation methods, such as product quantization, may outperform projection-based methods in certain scenarios. To accommodate a broader range of distance estimation techniques, we introduce a novel, data-driven distance correction scheme that is agnostic to the source of the approximate distances. This approach also provides a generalized framework for distance computation, extending its applicability beyond projection-based approximation.

A. A Data-Driven Distance Correction

Distance correction is essential to overcome the loss of accuracy when the approximate distance dis' is used alone. Specifically, ADSampling uses the condition $dis' - \epsilon \cdot \tau > \tau$ to determine whether the candidate point p is unlikely to be added to the queue Q, where $dis' - \epsilon \cdot \tau$ is the corrected approximate distance after using the bound $\epsilon \cdot \tau$; DDC$_{\text{res}}$ similarly uses the condition $dis' - m \cdot \sigma > \tau$ to exclude candidate point p, where $dis' - m \cdot \sigma$ is the corrected approximate distance. For both methods, if the condition is not satisfied, additional dimensions are sampled to compute a refined approximate distance dis' for the next-round correction. In summary, the conditions $dis' - \epsilon \cdot \tau > \tau$ in ADSampling and $dis' - m \cdot \sigma > \tau$ in DDC$_{\text{res}}$ facilitate early stopping, thereby improving the

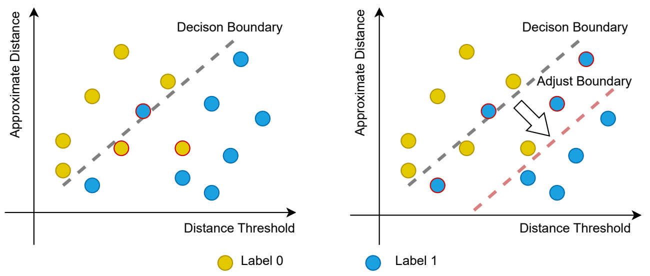


Fig. 4: The Example of the Learned Decision Boundary

efficiency. If the conditions are not met, incremental distance correction can still ensure sustained accuracy.

Data-Driven Distance Correction. Both ADSampling and DDC$_{\text{res}}$ require the error bounds $\epsilon \cdot \tau$ and $m \cdot \sigma$ for correction, with these bounds working for projection distances. This raises a question: how can we determine the error bound for distance correction for an arbitrary approximate distance, which may not originate from a projection? To address this, our key insight is to treat the error bound as the parameter(s). Recall that the purpose of the error bound is to adjust the approximate distance dis' to $(dis' - \text{parameter})$. The parameter(s), or error bound, should ensure that $f_{\text{parameter}}(dis') > \tau$ if and only if $dis > \tau$, where $f_{\text{parameter}}(dis')$ represents the corrected approximate distance. This observation implies that if the exact distance exceeds τ, the corrected approximate distance (i.e., $f_{\text{parameter}}(dis')$) should also exceed τ, effectively pruning irrelevant candidates. Thus, identifying the error bound reduces to determining the parameter(s) that satisfy this pruning condition. To address this, we propose a data-driven approach to calibrate the parameter(s) for distance correction.

Parameter Learning. The next consideration is how to determine the parameter(s). Recall that our objective is to ensure that $f_{\text{parameter}}(dis') > \tau$ when $dis > \tau$. Equivalently, we need $f_{\text{parameter}}(dis') - \tau > 0$ when $dis > \tau$, and $f_{\text{parameter}}(dis') - \tau \leq 0$ otherwise. If we label the first case (where $dis > \tau$) as label 1, and the second case (where $dis \leq \tau$) as label 0, this formulation leads naturally to a classification problem. Here, the features are dis' and the threshold τ, and the task is to obtain the weights of the features for classification.

We implement the above classification model using a linear model, selecting logistic regression with cross-entropy loss trained via stochastic gradient descent (SGD) for its stable performance and high training efficiency. Empirical results indicate similar outcomes across various linear models, so we omit further discussion on model selection. Formally, we can express this linear model as:

$$\begin{aligned} L &= sign(w_1 dis' + w_2 \tau + b > 0) \\ &= sign(m_1 dis' + \beta > \tau) \end{aligned}$$

where Label 0: $dis \leq \tau$ and Label 1: $dis > \tau$.

To summarize, we recast the error-bound estimation process as a parameter learning task (where parameters are weights m_1 and β), and we directly use the learned classification model for distance correction, thereby determining whether a candidate point should be excluded or not. The training data is sampled from vector points and queries, and the whole process works without any assumptions about the estimated distance dis'.

Adaptive Adjustment. After obtaining the initial parameters, we derive the decision boundary $m_1 \cdot dis' + \beta$. One advantage of our data-driven correction approach is the flexibility it provides in adjusting the parameter β to β', thereby modifying the decision boundary. This adjustment enables a balance between accuracy (specifically recall) and efficiency, as achieving high recall for Label 1 (i.e., when $dis > \tau$) is essential for optimizing efficiency: In extreme scenarios where Label 1 instances are absent, no pruning occurs, potentially impacting performance. Conversely, high recall for Label 0 is critical for the accuracy of the AKNN search, as it necessitates additional dimension sampling to obtain a more precious distance for the corresponding candidate point in this case.

To implement the above idea, we can set a target recall on the training set, and then identify the β' value that corresponds to the desired recall level for a given AKNN search accuracy. In particular, using Binary Cross Entropy (BCE) as the loss function, we define the threshold τ as the k-NN distance of a query where the nearest neighbor with Label 0 first appears. Subsequently, a binary search on β' can be conducted to ensure that the learned model achieves the target recall r for Label 0 on the training dataset. This process enables automatic and adaptive parameter configuration to meet specific recall targets.

Example 2. *The left figure in Fig. 4 shows the decision boundary of our data-driven approach, with parameters m_1 and β learned through training data. Data points with red borders indicate misclassified instances. The right figure in Fig. 4 shows how adjusting the boundary by updating β to β' can affect classification results. This adjustment is designed to trade off a small amount of accuracy for label 1 to ensure that the accuracy for label 0 meets essential performance requirements (i.e., recall targets).*

Remark. The data-driven approach can be aligned with DDC$_{\text{res}}$ presented in § IV by setting $m_1 = 1$ and $\beta = m \cdot \sigma$. Also, this approach offers greater flexibility as it can accommodate any approximate distance, enhancing its generality.

B. Implementation

This section provides details on the implementation of our proposed data-driven distance correction. We also show how to implement incremental distance correction.

Approximate Distances. Our new distance correction method is designed to be flexible across various types of approximate distances. For projection distances, we use a straightforward PCA projection as an approximate distance measure without applying the decomposition given by Equation 2. This approach is referred to as DDC$_{\text{pca}}$. The features for the model include the approximate distance and the threshold.

For quantization distances, we calculate the distance from quantized centroids of query q to the data point u, known as the asymmetric distance, as the approximate distance dis'. We also use OPQ [35] as our quantization distance method, and thus form the final distance computation method denoted as DDC$_{\text{opq}}$. Besides the approximate distance and threshold, we incorporate the distance from u to its quantized centroid as an additional feature. This additional feature further enhances the effectiveness of the linear model. Also, we forgo hardware-specific optimizations, concentrating instead on the core distance computation algorithm itself [36], [6].

Incremental Correction. We now discuss the incremental approach for data-driven distance correction. Similar to Algorithm 2, we start by training an initial classifier, after the initial approximate distance is given. Each time the classifier fails to confirm that $dis > \tau$ (i.e., it assigns label 0), we incrementally sample additional dimensions to compute a refined approximate distance, dis, and train a new classifier. This process is repeated until the projected dimension matches the data dimension, at which point we obtain an exact distance.

C. Discussion of Out-of-Distribution Query

For our proposed methods, we need to obtain the error distribution to establish an error bound for DDC$_{\text{res}}$ and to form the training data of the linear models for DDC$_{\text{opq}}$ and DDC$_{\text{pca}}$. However, in real-world applications, query diversity may cause the error distribution to deviate from the original one. Such out-of-distribution (OOD) queries pose a challenge to all of our methods. We analyze the impact of OOD queries on each method as follows: First, the DDC$_{\text{res}}$ algorithm treats the query as a deterministic variable when computing the error bound, making it less sensitive to OOD queries. We validate the robustness of DDC$_{\text{res}}$ through empirical studies (see Exp-A.2 in our technical report [31]). Second, for the linear model-based methods (DDC$_{\text{opq}}$ and DDC$_{\text{pca}}$), OOD queries have a significant impact because they rely on the query to generate training data. Our experimental studies also confirm this sensitivity (see Exp-A.2 of [31]). To address this limitation, we propose to retrain the model with approximately 100 OOD queries. Experimental results in Exp-A.3 of [31] show that this approach effectively mitigates the performance degradation.

VI. Analysis of Proposed Methods

This section presents an empirical analysis of our distance computation methods. This discussion is divided based on the approximate distance employed in the computation.

A. Analysis Under Projection Distance

Orthogonal projection is flexible in incrementally adding dimensions during distance approximation. This process continues either until the exact distance is calculated (using all dimensions) or until an early termination is reached based on an error bound to exclude candidate points. Thus, the average number of scanned dimensions determines the time cost of our method when using projection-based approximate distances. In Exp-6 of § VII, we measured the average number of dimensions used and observed that our methods require

TABLE II: The Description of Datasets

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
WORD2VEC	300	1,000,000	1000	Text
GLOVE	300	2,196,017	1000	Text
TINY5M	384	5,000,000	1000	Image
TINY80M	150	79,302,017	1000	Image
SIFT100M	128	100,000,000	1000	Image

scanning only a small number of dimensions. This result demonstrates the efficiency of our method.

Moreover, the DDC$_{res}$ and DDC$_{pca}$ methods incur additional time costs due to the need to rotate the query vectors. For a single query, the time complexity of performing matrix multiplication for query projection (rotation) is $O(D^2)$. However, this cost is negligible, as experiments (in Exp-3 of § VII) confirm that rotation accounts for only 3% of the time in high-recall AKNN search scenarios.

B. Analysis Under Quantization Distance

For our method DDC$_{opq}$ using quantization distance, we evaluate the rate of pruned candidate points (i.e., pruned rate) to assess its efficiency. Experimental results in Exp-6 show that DDC$_{opq}$ maintains a high pruned rate, indicating its high efficiency. The quantization method incurs additional computational cost mainly due to the OPQ (Optimized Product Quantization) rotation, which has a time cost of $O(D^2)$, and the construction of a lookup table, which has a time cost of $O(D \cdot 2^{nbit})$, since in the OPQ process D is divided into m subspaces, each containing 2^{nbit} quantized centroids. However, by leveraging the lookup table, asymmetric distances can be computed with only m table lookups, significantly reducing computational requirements.

Moreover, these methods entail extra storage, specifically requiring $n \cdot m \cdot nbit$ bits to store the quantized representations across the m subspaces. In typical configurations, m is set to a fraction, such as $1/4$ or lower, of the original dimension D. Consequently, this setup results in an additional storage cost of approximately $1/32$ of the dataset size when using float32 vectors.

VII. Experiments

A. Experimental Settings

Datasets. We utilize eight publicly available datasets of varying scales and sources, as detailed in Table II. These datasets have been widely adopted as benchmarks for assessing AKNN algorithms, encompassing both data and query vectors. For datasets that provide designated training data, such as GIST and DEEP, we employ this pre-defined training data directly. For datasets without designated training data, we randomly sample from the data vectors as the query vectors to form a training set, then remove these samples from the dataset to ensure a clean evaluation set. All data in our experiments are stored in the float32 format.

Evaluation Metrics. To assess search accuracy, we use recall@K, defined as $\frac{|T \cap G|}{K}$, where G represents the ground-truth KNN set for a given query within dataset S, and T denotes the result set obtained by AKNN algorithms. For efficiency evaluation, we employ queries-per-second (QPS), which is the number of queries processed per second, including the end-to-end query time. Additionally, for projection-based distance computation methods, we measure the total number of dimensions scanned; for quantization-based methods, we evaluate efficiency using the pruned rate. All metrics are reported as averages over the entire query set.

Algorithms. Our distance computation methods include DDC$_{res}$ (see § IV.D), DDC$_{pca}$ (see § V.B), and DDC$_{opq}$ (see § V.B). We integrate these methods into AKNN algorithms to form new variants, and we also compare with the original AKNN algorithms. All tested algorithms include:

- HNSW: HNSW with all exact distances computed;
- HNSW++: HNSW with ADSampling for distance computing;
- HNSW-DDC$_{opq}$: HNSW with DDC$_{opq}$ for distance computing;
- HNSW-DDC$_{pca}$: HNSW with DDC$_{pca}$ for distance computing;
- HNSW-DDC$_{res}$: HNSW with DDC$_{res}$ for distance computing;
- IVF: IVF with all exact distances computed;
- IVF++: IVF with ADSampling for distance computing;
- IVF-DDC$_{opq}$: IVF with DDC$_{opq}$ for distance computing;
- IVF-DDC$_{pca}$: IVF with DDC$_{pca}$ for distance computing;
- IVF-DDC$_{res}$: IVF with DDC$_{res}$ for distance computing;
- FINGER: HNSW with FINGER for distance computing.

Implementation Details. All C++ code was compiled using g++ version 11.4.0 with -O3 optimization, and all SIMD operations were disabled, similar to the setup for ADSampling. Experiments were conducted on an IIntel(R) Xeon(R) Platinum 8352V CPU @2.10GHz with 512GB memory, running on Ubuntu Linux. We first provide details on the AKNN algorithms used. For HNSW, the parameter M specifies the number of connected neighbors, while $efConstruction$ controls the quality of AKNN. Following [9], we set $M = 16$ and $efConstruction = 500$. For IVF, as advised in the Faiss library [37], we set the number of clusters to $4,096$.

To implement our data-driven distance computation methods, we use a simple labeling approach during model training. Specifically, we select 10,000 vectors from training vectors as training queries. Then, the KNNs of each training query are assigned as positive samples (label 0). For negative samples (label 1), we collect 500,000 items through a query process. To train the linear classifier, we precompute approximate distances, thresholds, and additional features offline. The classifier is then trained using Binary Cross-Entropy (BCE) loss. We also set a recall target r of 0.995 for the time-accuracy trade-off experiment. For further implementation details, please refer to our technical report [31].

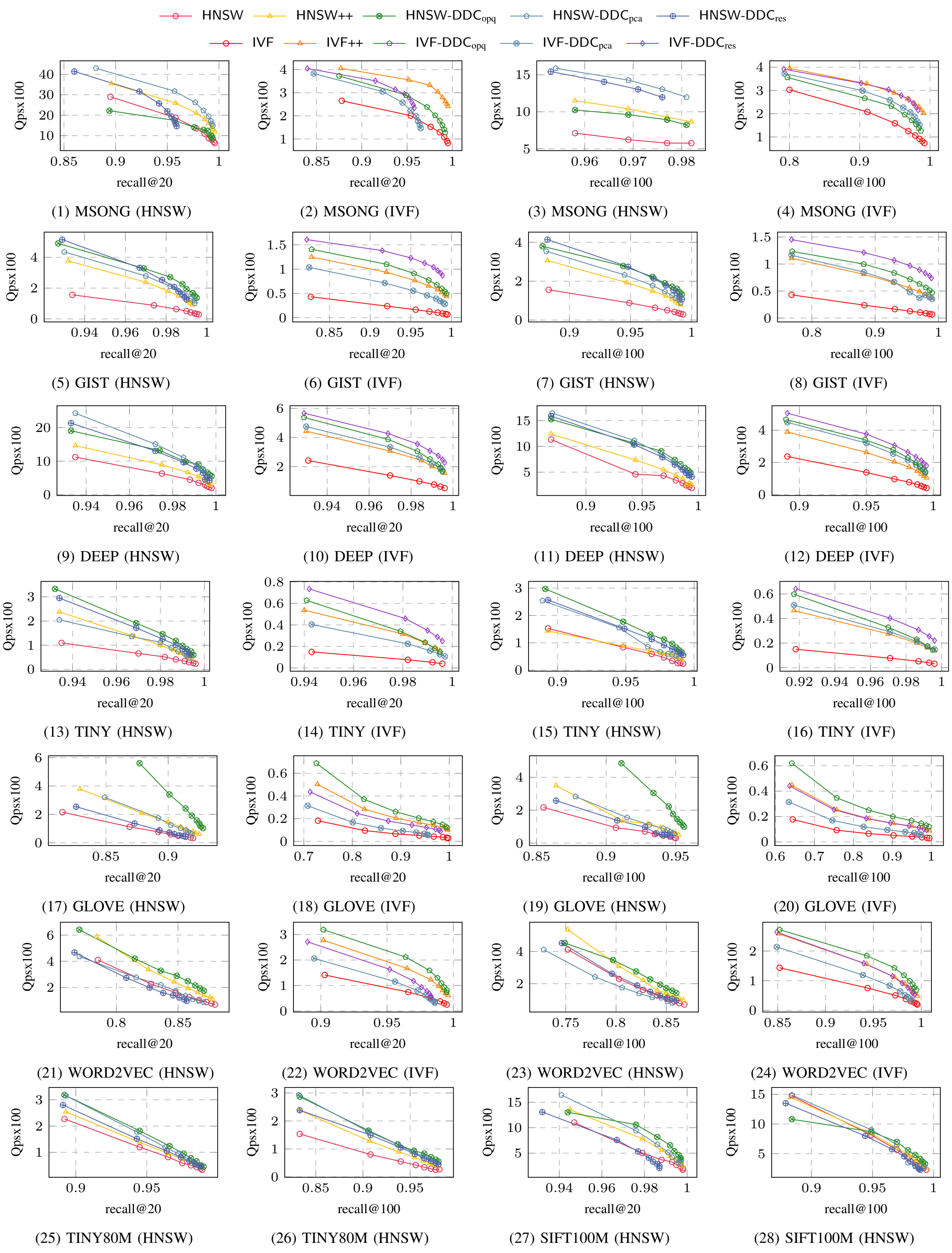
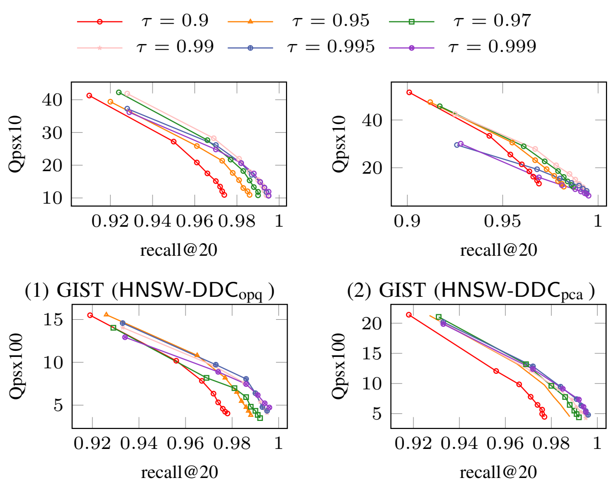


Fig. 5: The Test of Performance Among Various Methods



(1) GIST (HNSW-DDC$_{\text{opq}}$) (2) GIST (HNSW-DDC$_{\text{pca}}$)

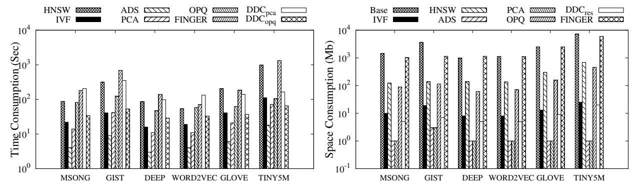
(3) DEEP (HNSW-DDC$_{\text{opq}}$) (4) DEEP (HNSW-DDC$_{\text{pca}}$)

Fig. 6: Varying the Target Recall

B. Experimental Results

Exp-1: Performance Test. To investigate the tradeoff between time and accuracy for various methods, we vary N^{ef} for HNSW, HNSW++, HNSW-DDC$_{\text{opq}}$, HNSW-DDC$_{\text{pca}}$, and HNSW-DDC$_{\text{res}}$, as well as N^{probe} for IVF, IVF++, IVF-DDC$_{\text{opq}}$, IVF-DDC$_{\text{pca}}$, and IVF-DDC$_{\text{res}}$. Fig. 5 shows the time-accuracy curve for all algorithms, where the upper right indicates better performance. Our findings reveal the following: (1) Our distance computation methods demonstrate high efficiency. Specifically, our method, DDC$_{\text{res}}$, achieves a 2x speedup over HNSW and a 1.45x speedup over ADSampling when applied to the TINY80M dataset in combination with HNSW. Notably, the computation of PAC or OPQ matrix on the database, along with the training of linear models, remains unaffected by data scale due to the sampling strategy employed. Specifically, for large datasets, following the empirical findings of the Faiss library [37], we sample 1 million data points to derive the PCA matrix; we sample 65,536 points from the database to obtain the OPQ matrix. (2) Our PCA-based distance computation methods, DDC$_{\text{res}}$ and DDC$_{\text{pca}}$, when used with AKNN algorithms such as HNSW , typically outperform quantization-based methods such as DDC$_{\text{opq}}$ on image datasets, including GIST and SIFT. This advantage is due to the skewed variance in these datasets. For example, a PCA projection to 32 dimensions preserves 67% of the variance in the GIST dataset and 82% in the SIFT dataset. In contrast, DDC$_{\text{opq}}$ outperforms PCA-based methods on datasets such as WORD2VEC and GLOVE. In these datasets, the variance is more evenly distributed, with a 32-dimensional PCA retaining only 36% and 18% of the variance for WORD2VEC and GLOVE, respectively. This observation suggests that analysis of variance skewness can effectively guide the selection of our proposed methods.

Exp-2: Varying Target Recall. The target recall is used to train the linear model by adaptively adjusting the decision boundary (see § V). We examine the effect of target recall on the data-driven methods DDC$_{\text{opq}}$ and DDC$_{\text{pca}}$ when applied to the AKNN algorithm HNSW and present results on the GIST and DEEP datasets in Fig. 6. We observe that when the target recall is set to 0.995, the search algorithm achieves the best trade-off between efficiency and recall loss (less than 0.5%). Thus, we select this value as the default target recall.



(1) Pre-Processing Time (2) Pre-Processing Space

Fig. 7: The Test of the Pre-Processing Time and Space

Exp-3: Test of Pre-Processing Time and Space. As introduced in § VI, distance computation methods require varying levels of pre-processing time and space. Specifically, ADSampling and DDC$_{\text{res}}$ incur additional time and space costs due to projection, while DDC$_{\text{pca}}$ and DDC$_{\text{opq}}$, require linear classifier training. For comparison, we also include HNSW and IVF as they need pre-processing time and space for indexing. We evaluate these pre-processing costs across different methods, with the results on various datasets presented in Fig. 7.

Fig. 7 shows that both ADSampling and PCA exhibit low pre-processing times compared to the indexing times of HNSW and IVF. In contrast, our methods DDC$_{\text{pca}}$ and DDC$_{\text{opq}}$ demand more pre-processing time as they involve training a model. Yet, this time remains comparable to the indexing times of HNSW and IVF. Moreover, we note that the space required by DDC$_{\text{pca}}$ and ADSampling , which is solely for the projection matrix (D^2 floats), is negligible when compared to the index sizes of HNSW and IVF. Also, DDC$_{\text{res}}$ requires storage for the norms of N vectors, while DDC$_{\text{opq}}$ needs to store quantized vectors, which increase linearly with the dataset size. Nonetheless, this space requirement is still comparable to the index space of HNSW and IVF.

Exp-4: Comparison with FINGER. We also compare our methods with FINGER in the same setting. Since FINGER operates exclusively with HNSW, we focus on evaluating HNSW under various distance computation methods. We present the results on the GIST and DEEP datasets in Fig. 8, and the results on other datasets are available in our technical report [31]. The results demonstrate that our DDC$_{\text{res}}$ method is 20% to 30% faster than FINGER. Additionally, Fig. 7 in Exp-3 shows that FINGER requires much more preprocessing time and memory than our method, limiting its feasibility in memory-constrained settings.

Exp-5: Scalability Test. To evaluate the scalability of our methods, we conducted a test on the SIFT100M dataset. This dataset was divided into five groups, each containing 20 million, 40 million, 60 million, 80 million, and 100 million entries. We used HNSW as the underlying AKNN algorithm. In Fig. 9, we present the time required by HNSW to construct

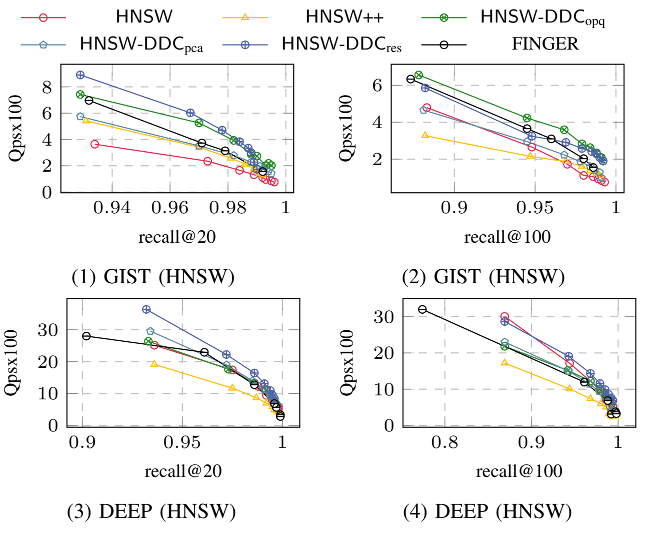


Fig. 8: The Comparison with FINGER

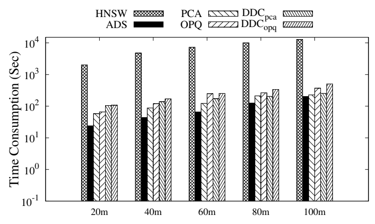


Fig. 9: The Test of Scalability on the SIFT100M dataset

the index, as well as the distance computation time for other methods. We first observe that the pre-processing time for distance methods, such as ADSampling, PCA, and OPQ, accounts for only 1%-5% of the indexing time of HNSW, regardless of dataset size. Our methods, DDC_{pca} and DDC_{opq} , also show small pre-processing time compared to the indexing time of HNSW. Also, we observe that the training time of linear models for DDC_{pca} and DDC_{opq} increases linearly with dataset size. For example, DDC_{pca} and DDC_{opq} require 105 seconds and 107 seconds, respectively, for preprocessing with a dataset of 20 million vectors, and 254 seconds and 504 seconds for a dataset of 100 million vectors.

Exp-6: Empirical Analysis of Our Methods. In § VI, we proposed using the scan dimension and pruned rate metrics to analyze the performance of projection-based methods (DDC_{res} and DDC_{pca}), and quantization-based method (DDC_{opq}), respectively. We thus compute the average scan dimension ratio (relative to the entire dimension) for various methods on the GIST and DEEP datasets, shown in the left panels of Fig. 10. Our methods show superior performance compared to baselines. For instance, when $N^{ef} = 2000$, DDC_{res} scans only 7% of the total dimensions, DDC_{pca} scans 15%, and ADsampling scans 26% on the GIST dataset. Similarly, we compute the average pruned rate in the right panels of Fig. 10. The results show that our methods achieve a higher pruned rate, further confirming their superiority.

Exp-7: Test of Approximation Accuracy. To evaluate the accuracy of our distance computation method, we directly

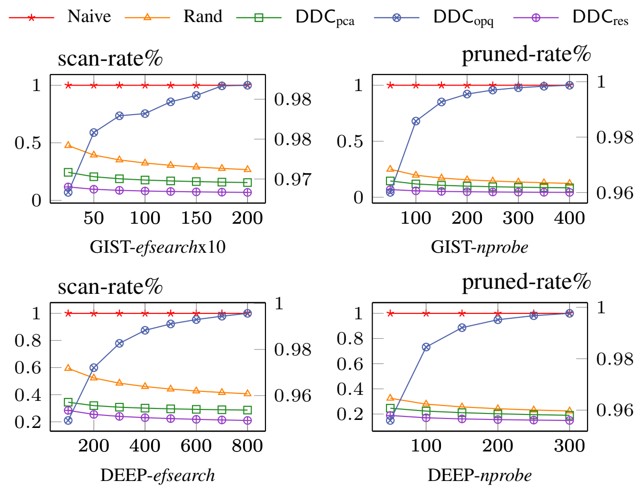


Fig. 10: The Empirical Analysis of Our Methods

TABLE III: The Test of Approximation Accuracy (Recall@100)

Dataset	PCA	Rand	DDC_{res}
DEEP	44.5	16.6	46.6
GIST	34.3	7.3	51.6
TINY	32.5	7.8	43.5
GLOVE	7.1	4.6	41.7
WORD2VEC	18.6	8.4	29.0

apply our method, DDC_{res} , to scan the points in the database, without relying on existing AKNN algorithms for finding AKNN. Table III presents the accuracy of different methods when projecting to 32 dimensions. The results indicate that DDC_{res} achieves higher accuracy than random projection (denoted as Rand) and outperforms PCA across most datasets. These findings confirm the superiority of our method.

Exp-8: Applications in Ant Group. Our proposed methods have been applied in the security search applications of Ant Group. Specifically, facial images are widely used in digital payment, where AKNN search plays a critical role in security during transactions. We conducted experiments on a private dataset from Ant Group, containing 1 million images with 512-dimensional embeddings. Our results show that the proposed DDC_{opq} method reduces retrieval time by 35% and increases throughput by 55.25% without sacrificing accuracy, greatly enhancing the efficiency of related applications.

VIII. Conclusion

In this paper, we propose novel methods for distance computation in AKNN search. We first analyze the estimation error between approximate and exact distances, and prove that using PCA projection can improve the effectiveness of ADSampling. To further improve the generalizability of ADSampling, we introduce a data-driven distance correction scheme that operates independently of the distance estimation source. Extensive experiments show that our methods greatly outperform ADSampling in search speed. In future work, we plan to explore advanced methods for distance computation to further refine the efficiency of AKNN search.

References

[1] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, “A survey of content-based image retrieval with high-level semantics,” *Pattern recognition*, vol. 40, no. 1, pp. 262–282, 2007.

[2] T. Cover and P. Hart, “Nearest neighbor pattern classification,” *IEEE transactions on information theory*, vol. 13, no. 1, pp. 21–27, 1967.

[3] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative filtering recommender systems,” in *The adaptive web: methods and strategies of web personalization*. Springer, 2007, pp. 291–324.

[4] J. J. Pan, J. Wang, and G. Li, “Vector database management techniques and systems,” in *Companion of the 2024 International Conference on Management of Data*, 2024, pp. 597–604.

[5] P. Indyk and R. Motwani, “Approximate nearest neighbors: towards removing the curse of dimensionality,” in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 1998, pp. 604–613.

[6] H. Jégou, M. Douze, and C. Schmid, “Product quantization for nearest neighbor search,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, 2011.

[7] A. Babenko and V. S. Lempitsky, “The inverted multi-index,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 6, pp. 1247–1260, 2015.

[8] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov, “Approximate nearest neighbor algorithm based on navigable small world graphs,” *Inf. Syst.*, vol. 45, pp. 61–68, 2014.

[9] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, 2020.

[10] C. Fu, C. Wang, and D. Cai, “High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 44, no. 8, pp. 4139–4150, 2022.

[11] C. Fu, C. Xiang, C. Wang, and D. Cai, “Fast approximate nearest neighbor search with the navigating spreading-out graph,” *Proc. VLDB Endow.*, vol. 12, no. 5, pp. 461–474, 2019.

[12] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, “Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement,” *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 8, pp. 1475–1488, 2020.

[13] Y. Peng, B. Choi, T. N. Chan, J. Yang, and J. Xu, “Efficient approximate nearest neighbor search in multi-dimensional databases,” *Proc. ACM Manag. Data*, vol. 1, no. 1, pp. 54:1–54:27, 2023.

[14] S. Dasgupta and Y. Freund, “Random projection trees and low dimensional manifolds,” in *Proceedings of the fortieth annual ACM symposium on Theory of computing*, 2008, pp. 537–546.

[15] A. Beygelzimer, S. Kakade, and J. Langford, “Cover trees for nearest neighbor,” in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 97–104.

[16] P. Ram and K. Sinha, “Revisiting kd-tree for nearest neighbor search,” in *Proceedings of the 25th acm sigkdd international conference on knowledge discovery & data mining*, 2019, pp. 1378–1388.

[17] Y. Sun, W. Wang, J. Qin, Y. Zhang, and X. Lin, “SRS: solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index,” *Proc. VLDB Endow.*, vol. 8, no. 1, pp. 1–12, 2014.

[18] B. Zheng, X. Zhao, L. Weng, N. Q. V. Hung, H. Liu, and C. S. Jensen, “PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search,” *Proc. VLDB Endow.*, vol. 13, no. 5, pp. 643–655, 2020.

[19] J. Gan, J. Feng, Q. Fang, and W. Ng, “Locality-sensitive hashing scheme based on dynamic collision counting,” in *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, 2012, pp. 541–552.

[20] M. X. Goemans and D. P. Williamson, “Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming,” *Journal of the ACM (JACM)*, vol. 42, no. 6, pp. 1115–1145, 1995.

[21] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, “Query-aware locality-sensitive hashing for approximate nearest neighbor search,” *Proceedings of the VLDB Endowment*, vol. 9, no. 1, pp. 1–12, 2015.

[22] Q. Huang, J. Feng, Q. Fang, W. Ng, and W. Wang, “Query-aware locality-sensitive hashing scheme for lp norm,” *The VLDB Journal*, vol. 26, no. 5, pp. 683–708, 2017.

[23] J. Gao and C. Long, “High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations,” *Proc. ACM Manag. Data*, vol. 1, no. 2, pp. 137:1–137:27, 2023.

[24] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, “A survey on learning to hash,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 769–790, 2017.

[25] P. Chen, W.-C. Chang, J.-Y. Jiang, H.-F. Yu, I. Dhillon, and C.-J. Hsieh, “Finger: Fast inference for graph-based approximate nearest neighbor search,” in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 3225–3235.

[26] W. B. Johnson, J. Lindenstrauss, and G. Schechtman, “Extensions of lipschitz maps into banach spaces,” *Israel Journal of Mathematics*, vol. 54, no. 2, pp. 129–138, 1986.

[27] D. Baranchuk, D. Persiyanov, A. Sinitsin, and A. Babenko, “Learning to route in similarity graphs,” in *International Conference on Machine Learning*. PMLR, 2019, pp. 475–484.

[28] C. Feng, D. Lian, X. Wang, Z. Liu, X. Xie, and E. Chen, “Reinforcement routing on proximity graph for efficient recommendation,” *ACM Transactions on Information Systems*, vol. 41, no. 1, pp. 1–27, 2023.

[29] Z. Wang, Q. Wang, X. Cheng, P. Wang, T. Palpanas, and W. Wang, “Steiner-hardness: A query hardness measure for graph-based ann indexes,” *arXiv preprint arXiv:2408.13899*, 2024.

[30] C. Li, M. Zhang, D. G. Andersen, and Y. He, “Improving approximate nearest neighbor search through learned adaptive early termination,” in *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, 2020, pp. 2539–2554.

[31] M. Yang, J. Jin, X. Wang, Z. Shen, W. Jia, W. Li, and W. Wang. (2024) Technical report. [Online]. Available: github.com/mingyu-hkustgz/Res-Infer/blob/nolearn/technical_report.pdf

[32] K. Lu, M. Kudo, C. Xiao, and Y. Ishikawa, “HVS: hierarchical graph structure based on voronoi diagrams for solving approximate nearest neighbor search,” *Proc. VLDB Endow.*, vol. 15, no. 2, pp. 246–258, 2021.

[33] B. Harwood and T. Drummond, “Fanng: Fast approximate nearest neighbour graphs,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5713–5722.

[34] J. Wang, T. Zhang, N. Sebe, H. T. Shen *et al.*, “A survey on learning to hash,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 769–790, 2017.

[35] T. Ge, K. He, Q. Ke, and J. Sun, “Optimized product quantization,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 744–755, 2014.

[36] F. André, A.-M. Kermarrec, and N. Le Scouarnec, “Cache locality is not enough: High-performance nearest neighbor search with product quantization fast scan,” *Proceedings of the VLDB Endowment*, vol. 9, no. 4, 2015.

[37] J. Johnson, M. Douze, and H. Jégou, “Billion-scale similarity search with GPUs,” *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

APPENDIX

A. *The Proofs*

Proof of Lemma 2. We get $\epsilon \sim \mathcal{N}(0, \sigma^2)$ under the Gaussian distribution assumption. The error quantile can be expressed by the inverse CDF (quantile function) of the Gaussian distribution with probability q:

$$F^{-1}(p) = \mu + \sigma\sqrt{2}\text{erf}^{-1}(2p-1)$$

The $\text{erf}^{-1}(x)$ is the inverse function of the error function and is positive for $x > 0$. When the probability p is fixed, $\mu = 0$ by centralized, only σ affects the value. Then the variance of the error is minimized, and the right quantile($p > 0.5$) of the error is also minimized

Proof of Theorem 1. First, we consider the simplest case where the projection dimension $d = 1$. We define a vector $\mathbf{w}_1 \in \mathbb{R}^D$ as the direction of the lower dimensional space. As we are only interested in the direction of the space, we set $\mathbf{w}_1$ to be of unit length where $\mathbf{w}_1^T\mathbf{w}_1 = 1$. Then the vector $\mathbf{x}_n$ can be projected onto this new space as $\hat{\mathbf{x}}_n = \mathbf{w}_1^T\mathbf{x}_n$. We define $\bar{\mathbf{x}}$ as the mean of the $\mathbf{x} \in S$ in the original space, and the mean of the vectors in the projected space is given by $\hat{\bar{\mathbf{x}}} = \mathbf{w}_1^T\bar{\mathbf{x}}$. We can write the variance of the projected data as:

$$\begin{aligned}\sigma^2(\hat{\mathbf{x}}) &= \frac{1}{N}\sum_{n=1}^{N}(\hat{\mathbf{x}}_n - \hat{\bar{\mathbf{x}}})^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}_1^T\mathbf{x}_n - \mathbf{w}_1^T\bar{\mathbf{x}})^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{w}_1^T(\mathbf{x}_n - \bar{\mathbf{x}})(\mathbf{x}_n - \bar{\mathbf{x}})^T\mathbf{w}_1 \\ &= \mathbf{w}_1^T\Sigma\mathbf{w}_1\end{aligned}$$

where Σ is the covariance matrix. To minimize the residual dimension variance, we consider the stationary point of the above equation. Whether maximizing or minimizing the above formulas, we introduce a Lagrange multiplier λ_1 and formulate our optimization objective as follows:

$$J(\mathbf{w}_1) = \mathbf{w}_1^T\Sigma\mathbf{w}_1 + \lambda_1(1 - \mathbf{w}_1^T\mathbf{w}_1)$$

with the unit norm constraint $\mathbf{w}_1^T\mathbf{w}_1 = 1$. Setting the derivative of the above equation, we get a stationary point when,

$$\frac{\partial J(\mathbf{w}_1)}{\partial \mathbf{w}_1} = 2\Sigma\mathbf{w}_1 - 2\lambda_1\mathbf{w}_1 = 0$$

$$\mathbf{w}_1^T\Sigma\mathbf{w}_1 = \lambda_1$$

This shows that at the stationary point, $\mathbf{w}_1$ must be an eigenvector of Σ and λ_1 the eigenvalue. we can see that the maximum or minimum variance is equal to the eigenvalue λ_1. We can identify additional principal components by choosing directions that maximize variance while being orthogonal to the existing ones. For the general case of a lower dimensional space with d dimensions with $d < D$, the principal components are the eigenvectors $\mathbf{w}_1, \mathbf{w}_2...\mathbf{w}_d$ corresponding to the d largest eigenvalues $\lambda_1, \lambda_2...\lambda_d$ and the residual dimension corresponding the r smallest eigenvalues $\lambda_{d+1}, \lambda_{d+2}...\lambda_r$. Then we can prove the PCA matrix $[\mathbf{w}_1...\mathbf{w}_D]^T$ maximizes the projection dimension variance which also minimizes the residual dimension variance.

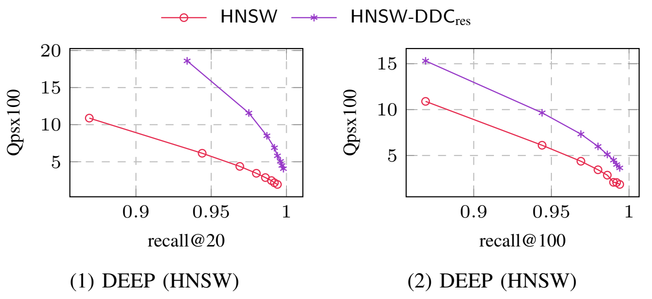


Fig. 11: The Test of the Inner Product

B. *Additional Experiments*

We first present more implementation details, followed by a presentation of additional experimental results.

More Implementation Details. For the random projection approach, we set $\epsilon_0 = 2.1$ and $\Delta_d = 32$, following the recommendations for ADSampling. For both DDC$_{res}$ and DDC$_{pca}$, each Δ_d was also set to 32 to maintain the same setting as ADSampling. In the DDC$_{opq}$ approach, the subspace number was set to $d/8$ for the GIST dataset and $d/4$ for the others, since each dataset dimension is divisible by 4. The parameter *nbits* for DDC$_{opq}$ was set to 8 as per the default. The target recall for both DDC$_{pca}$ and DDC$_{opq}$ methods was configured to 0.995. For the multiplier m in DDC$_{res}$, values were set to 8 for the SIFT, GIST, and DEEP datasets, 12 for TINY and WORD2VEC, and 16 for the GlOVE dataset. In cases involving adaptive adjustment of classifiers, the target recall for each classifier was determined based on Δ_d, calculated as $r_i = \left(1 - \frac{1-r}{D/\Delta_d}\right)$. Upon receiving a query, the algorithm first applies a transformation through matrix multiplication, implemented with the C++ Eigen library.

Exp-A.1. Test of Inner Product. Note that our DDC$_{res}$ method can be easily adapted for maximum inner product search by omitting the norm term in the distance estimation. In this case, the (exact) inner product can be decomposed as follows:

$$\langle \mathbf{q}, \mathbf{x} \rangle = \langle \mathbf{q}_D, \mathbf{x}_D \rangle = \langle \mathbf{q}_d, \mathbf{x}_d \rangle + \langle \mathbf{q}_r, \mathbf{x}_r \rangle, \quad (4)$$

where the error term remains consistent with the original DDC$_{res}$ method. This allows the approximate distance to be computed as $\langle \mathbf{q}_d, \mathbf{x}_d \rangle$ in $O(d)$ operations.

We continue to apply PCA projection to minimize the variance of the error term and use the Gaussian assumption to prune distance computations effectively. The results, shown in Fig. 11, indicate DDC$_{res}$ achieves a 1.5x to 2x speedup compared to the original HNSW algorithm on DEEP.

Exp-A.2. Effect of Out-of-Distribution Queries. Out-of-distribution (OOD) data poses a big challenge for data-driven learning methods. To assess the performance of our approach on OOD queries, we introduce Gaussian noise to

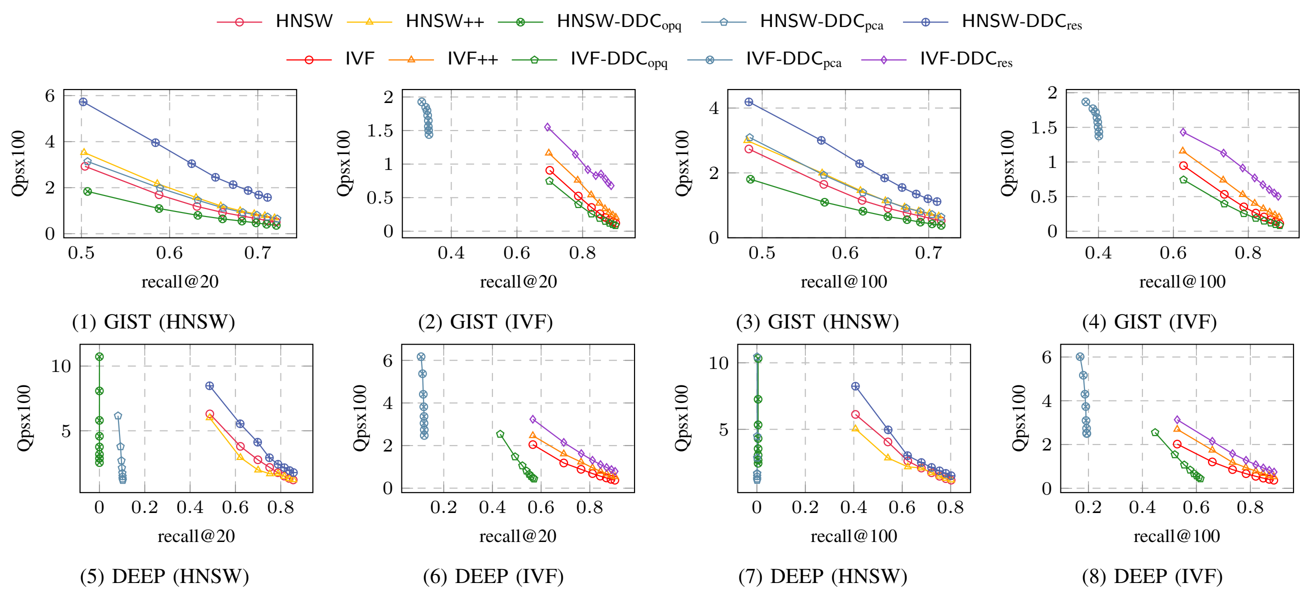


Fig. 12: The Effect of Out-of-Distribution Queries

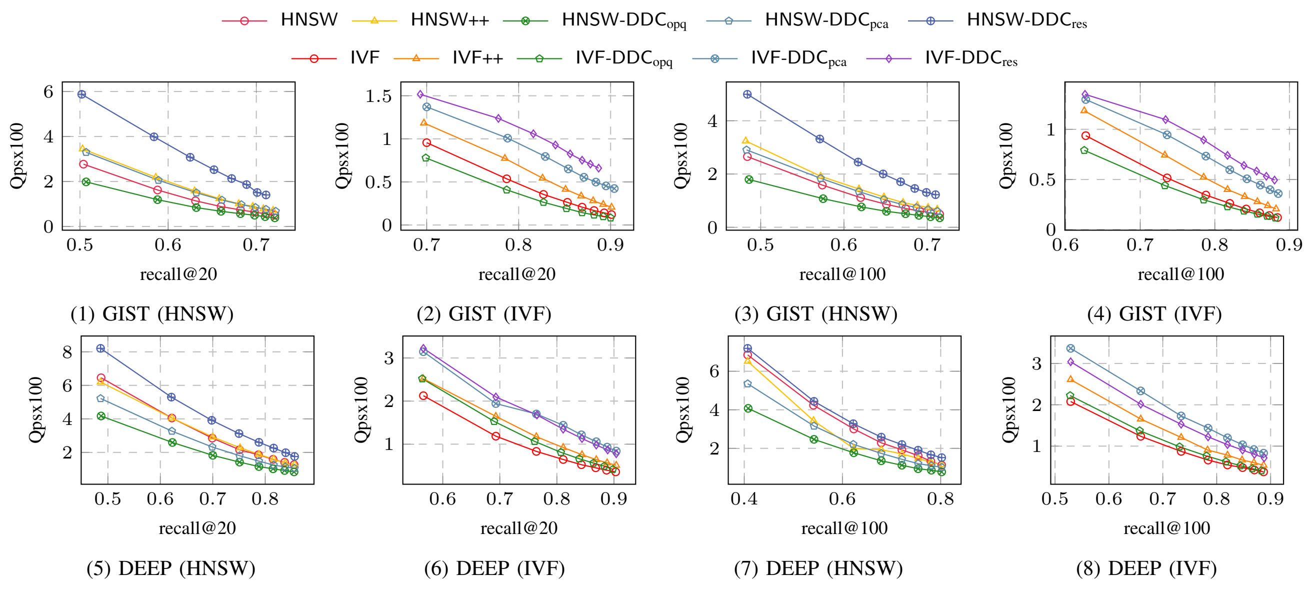


Fig. 13: The Test of Handling Out-of-Distribution Queries

the original queries and perform experimental evaluations. The results, presented in Fig. 12, demonstrate that our algorithms, $\mathrm{DDC}_{\mathrm{res}}$ and ADSampling , are relatively robust against OOD queries. This robustness arises because ADSampling operates as a data-free method, leveraging error bounds based on the Johnson-Lindenstrauss (JL) lemma derived from random matrices. In contrast, the $\mathrm{DDC}_{\mathrm{res}}$ algorithm considers the query as a deterministic variable, enhancing its effectiveness for handling OOD queries. In comparison, the $\mathrm{DDC}_{\mathrm{pca}}$ and $\mathrm{DDC}_{\mathrm{opq}}$ algorithms exhibit limitations under OOD conditions, as they tend to suffer from either lower efficiency or reduced recall in these scenarios.

Exp-A.3. Test of Handling Out-of-Distribution Queries. To address the challenge of out-of-distribution (OOD) queries, a practical approach involves retraining the linear classifier. By utilizing 100 queries and requiring approximately 10 seconds, this method enables rapid model adaptation. As demonstrated in Fig. 13, our experimental results indicate that this retraining strategy effectively mitigates accuracy degradation of the linear model in OOD scenarios.

Exp-A.4: Additional Results on Comparison with FINGER. In Exp-4 of § VII, we presented a comparison with FINGER on the GIST and DEEP datasets. In this section, we provide additional results on other datasets, which are displayed in Fig. 14. The results are consistent with those in Exp-4, so we omit further discussion for brevity.

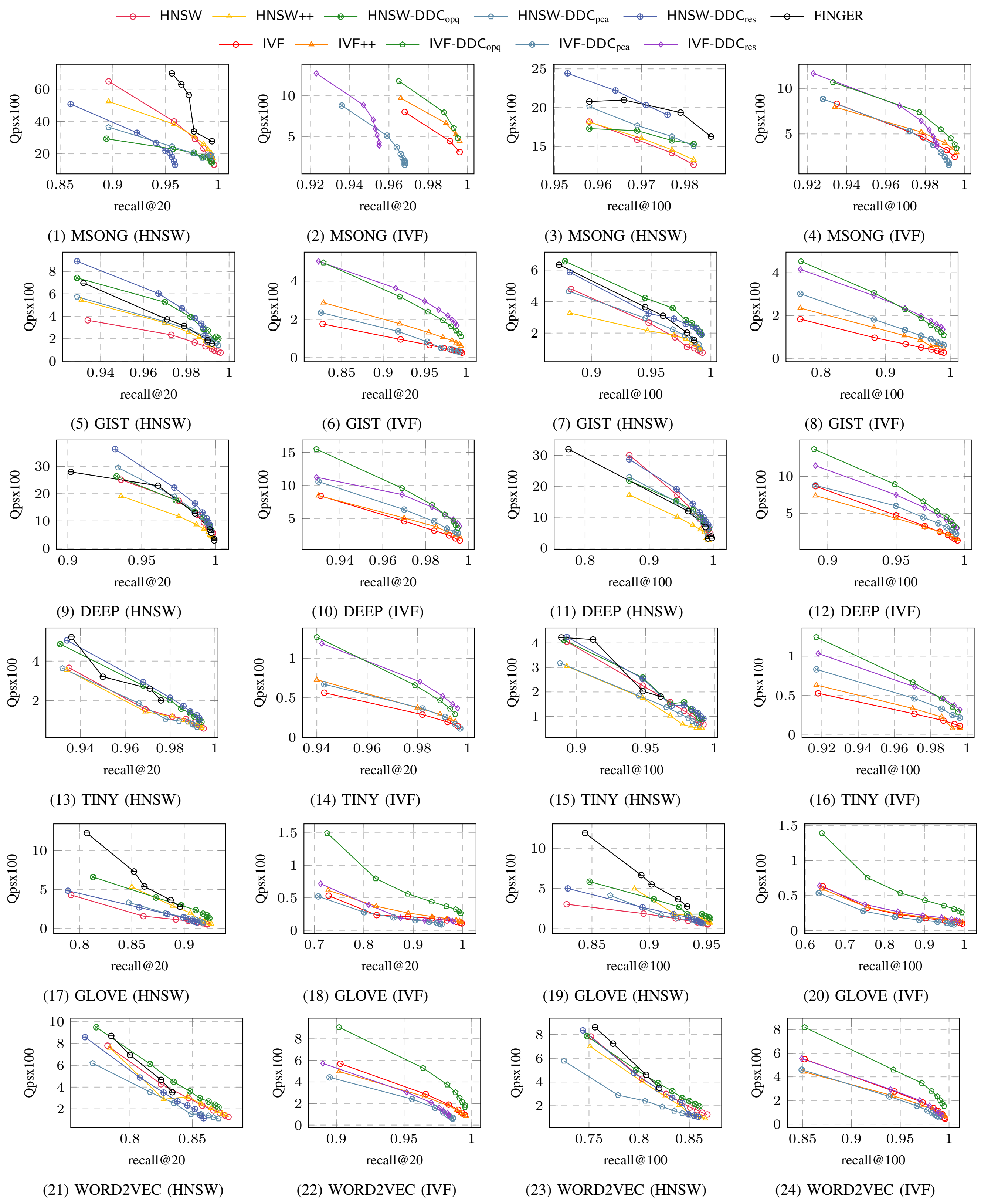


Fig. 14: The Additional Results on the Comparison with FINGER