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 security search scenarios of Ant Group 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], recommendation 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 an 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 this 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 the refinement phase. Notably, **distance computation is often the most time-consuming component of AKNN algorithms.** For example, in graph-based methods like HNSW [9], distance computation constitutes 80% of the total AKNN search time. Similarly, for inverted file-based methods 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. ADSampling first estimates an (initial) approximate distance between two points by random projection and obtains an error bound from the projection matrix for distance correction. The advantage of ADSampling lies in its ability to use error bounds to analyze whether the use of *current* approximate distances is sufficient in AKNN search [24], [23]. If not, more accurate distances are computed incrementally as a correction to the current approximation (until an exact distance is computed). Thus, ADSampling achieves a good balance between speed and accuracy in AKNN search by incremental corrections, 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 projec-**

tion typically incurs a larger error than optimized 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 approximate distances necessitate incremental correction through additional dimensionality sampling. However, deriving the error bound for ADSampling is determined by the randomness of the projection matrix employed in the distance estimation process. **This dependence constrains the applicability of** ADSampling to scenarios where approximate distances are generated by techniques other than random projection.

Our Idea. It can be observed that the excessive reliance on random projections makes ADSampling a data-free method but also limits its potential without considering the data distribution. Therefore, a natural question arises: Can we design a new distance computation method that retains the error bounds for distance correction and fully optimizes the efficiency from the data distribution?

We provide an affirmative answer in this paper. First, we analyze the root cause of the limited effectiveness of the projection distance. By decomposing the exact distance into approximate distance and estimation error. We minimize the estimation error via the orthogonal projection matrix with the database vectors. Also, by studying the distribution of estimation errors and introducing reasonable assumptions, we derive new error bounds for distance correction which are also proven 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), 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 a more effective and general method 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 PCA projection instead of random projections results in minimal approximation error. Consequently, we replace the random projections used in ADSampling with PCA projections to improve 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), 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 determine the appropriate parameter for the given problem. By learning this parameter, the proposed method avoids making assumptions about the source of the approximate distance, thus achieving generality. We also discuss how the proposed distance correction scheme can be integrated into existing AKNN algorithms to enhance their efficiency.

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 retrieval 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.

Due to space limitations, some proofs and experiments have been omitted and can be found in the technical report [25].

II. Preliminaries

Section II-A introduces the AKNN search problem and its associated algorithms. Following this, Section 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.

It is important to 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],

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

TABLE I: A Summary of Notations

Notation	Description
S	A set of points
D	The dimensionality of S
$\mathbf{R}$	The projection (rotate) matrix
$\mathbb{R}^d$	d-dimensional Euclidean space
dis, dis'	Exact and approximate distance
N^{prob}	The search parameter of IVF
N^{ef}	The search parameter of HNSW
$\|u, v\|$	The Euclidean distance between u and v
τ	The distance threshold
L	The linear classifier

[6], graph-based [9], [10], [11], [26], [13], [27], [12], tree-based [14], and hash-based [17], [18], [20], [19], [21], [22].

Inverted File-Based Algorithms. Inverted file-based algorithms, such as IVF, are often used to speed up AKNN searches. 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 Algorithms. 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) 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.

Other AKNN Algorithms. 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. However, in practice, these methods are generally 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 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-based methods, which compute distance in low-dimensional space, quantization computes the distance between the vector and 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, making it effective for distance computing.

Remark. Both projection and product 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 can result in reduced search accuracy. For example, none of the quantization methods achieve more than 60% recall without re-ranking [28], [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, the AKNN algorithm will fail to return an exact result.

C. Related Work

In the realm of distance computation acceleration between points or vectors, two notable methods are ADSampling [23] and FINGER [29]. ADSampling leverages the Johnson-Lindenstrauss (JL) lemma [30] to establish a probabilistic bound, utilizing this to expedite distance computations. This approach is grounded in a robust mathematical foundation, making it broadly applicable across various domains. Conversely, FINGER [29] is tailored specifically for graph-based approaches. It focuses on estimating angles between neighboring residual vectors during the graph search phase to enhance acceleration. FINGER achieves empirical success in accelerating over HNSW [9], but it also introduces longer index construction times and additional space requirements.

Data-driven methods have made significant contributions to the field of AKNN search. In particular, the literatures [31], [32] have applied learning techniques to predict the next node during graph traversal, enabling more efficient navigation through the search space. Methods [33], [34] use learning-based approaches to predict the hardness of the AKNN search,

enabling early termination of the search process. In contrast, our approach places a strong emphasis on distance computation, enabling seamless integration with the aforementioned methods. By addressing the computational bottlenecks inherent in distance calculations, our method provides a comprehensive solution to boost AKNN search efficiency throughout the entire search process.

III. Problem Analysis

To address the loss of accuracy that often accompanies the integration of approximate distances in existing AKNN algorithms, ADSampling has been proposed as an optimization technique for distance computations. The core idea of ADSampling is not only to use approximate distances but also to incorporate an error bound for correction purposes. Specifically, this treatment allows ADSampling to determine whether the current approximate distance is sufficient for the refinement phase of the AKNN search. If it is not, ADSampling will use more precise distance computations to compensate for any inadequacies in the approximate distances. Including these incremental distance computations ensures that ADSampling can increase the accuracy.

Distance Estimation. ADSampling first employs random projection to reduce the dimensionality from D to d, thereby estimating an approximate distance dis' for the exact distance dis. The relationship between the approximate and exact distances can be 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 introduces an innovative approach to improving the accuracy of AKNN algorithms by using error bounds (between approximate and exact distances) for distance correction. This action addresses the potential loss of accuracy inherent in approximate distance computations. Specifically, ADSampling uses a hypothesis test to assess whether the currently computed approximate distance is adequate. That is, if $dis' > (1+\epsilon)\cdot\tau$, where τ represents the maximum distance (threshold) of a queue Q, ADSampling confidently concludes that $dis > \tau$ under a given significance level $p = 2e^{-c_0\epsilon_0^2}$. Here, ϵ_0 is a parameter that requires empirical tuning. In this case, using the current approximate distance to exclude points from the queue Q is reliable.

Conversely, if the approximate distance does not satisfy this condition, it is insufficient to determine the exclusion of a point in the queue Q. In such cases, ADSampling mandates sampling additional dimensions for distance correction. This process involves computing a more precise approximate distance to conclusively determine whether $dis > \tau$ or $dis \leq \tau$. This incremental correction process continues until all dimensions are sampled and an 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 ignore a candidate point. Therefore, enhancing 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 quantized distances. Developing a generalized distance correction scheme could potentially improve the efficiency and applicability of ADSampling, especially when other approximate distances prove to be more effective than those derived from random projection.

IV. An Improved Projection-Based Distance Computation

This section primarily addresses the first limitation of ADSampling: the limited effectiveness of approximate distance estimation based on random projection. We demonstrate that using optimal orthogonal projection, rather than random projections, 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 yield approximate distances that are close to the exact distances. To do this, we decompose the exact distance to obtain the approximate distance and the estimation error after the projection/rotation. By minimizing the estimation error, we can achieve effective (projection-based) distance estimation by determining the optimal projection matrix that minimizes the estimation error. The treatment of non-projection-based distance estimation is left for the next section.

Suppose $\mathbf{x}$ and $\mathbf{q}$ are the D-dimensional database vector and query vector, respectively. We consider a simple model where data points $\mathbf{x}$ are randomly sampled from the dataset according to a fixed but unknown distribution U. We introduce a global rotation parameterized by the matrix $\mathbf{R}$. We denote the transformed (or rotated) vectors as $\mathbf{x}_D$ and $\mathbf{q}_D$, respectively.

The rotated vector $\mathbf{x}_D$, or $\mathbf{Rx}$, is then decomposed into $\mathbf{x}_d$ and $\mathbf{x}_r$. To explain the rotated vector, we think in this new coordinate system, we group the dimensions into two categories: the first d dimensions and the remaining dimensions. We then denote $\mathbf{x}_D$ as $(\mathbf{x}_d, \mathbf{x}_r)$, where $\mathbf{x}_d$ is $\mathbf{x}_D$ truncated to the first d dimensions, and x_r represents the rest. Similarly, we have $\mathbf{q}_D = (\mathbf{q}_d, \mathbf{q}_r)$. Then, the (exact) distance is decomposed as:

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

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

B. An Improved Distance Estimation

Equation 2 provides the estimation error, expressed as $\epsilon = -2\cdot\langle \mathbf{q}_r, \mathbf{x}_r\rangle$, between the exact and approximate distances. Assuming that the data follows a Gaussian distribution $\mathbf{x} \sim \mathcal{N}(0, \Sigma)$[2] when a query $\mathbf{q}$ is given, the distribution of the estimation error can be viewed as a linear combination of multiple Gaussian distributions.

Which Projection is Better? Under the constraints of orthogonal projection, our goal is to minimize the error variance. This approach aims to find the projection that provides the most effective distance estimation, making the approximate distance as close as possible to the exact distance.

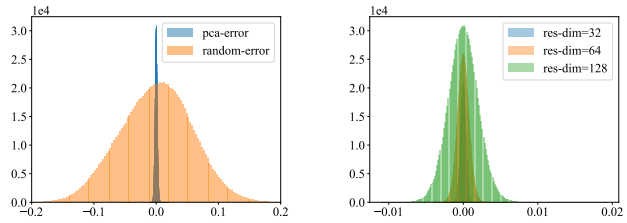
To achieve this, we first address the computation of the variance associated with the estimation error term. Let σ_i^2 represent the variance of the i-th dimension in distribution U (recall that the data are assumed to follow the distribution U). Upon receiving a query, the variance of the inner product for the residual dimension is $\sigma_i^2 q_i^2$. Consequently, the variance of the estimation error term is computed as:

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

Next, we explore all orthogonal projection matrices to find the one that brings the smallest variance of the error term and derive the following theorem:

Theorem 1 ([25]). *For a given set of vectors S, the Principal Component Analysis (PCA) projection matrix is the one that not only maximizes the variance in the projected dimensions but also minimizes the variance in the residual dimensions among all possible projection matrices.*

[2]We centralized the data to yield a mean of zero



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

Fig. 1: Distribution of Residual Error between PCA and Random Projection in the Deep1M Dataset

The PCA projection matrix is particularly well-suited for our problem, as it maximizes the projection variance (to make the approximate distances contain more information and thus closer to the exact distances). Additionally, we demonstrate that PCA also minimizes the variance in the residual dimensions (to make the estimation error as small as possible). Our investigation includes analyzing the distribution of estimation error terms on real datasets and comparing various projection matrices. 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.

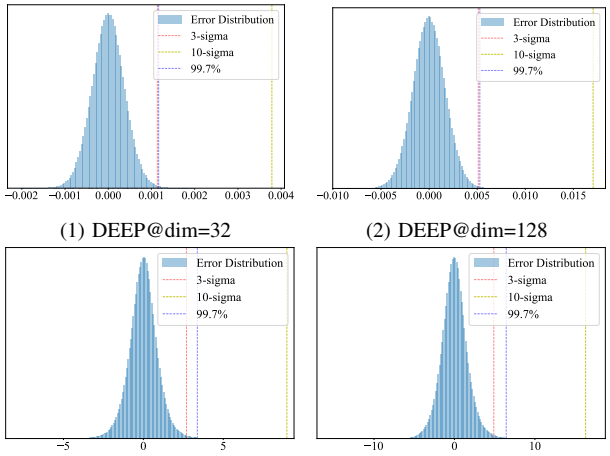
As illustrated in Fig. 1a, with a residual dimension of 128, the PCA projection matrix shows a more concentrated distribution compared to the random projection. Also, as shown in Fig. 1b, as the projection dimensions increase and the residual dimensions decrease, the error gradually converges to zero. This indicates that the PCA projection matrix is better than the random projection matrix in reducing estimation errors.

Remark of Independence. Equation 3 requires each dimension of $\mathbf{x}$ to be linearly independent. Fortunately, our implementation aligns the data through PCA, making the off-diagonal data of the covariance matrix equal to 0, thus avoiding the calculation of covariance.

C. An Improved Distance Correction

The success of ADSampling lies in utilizing Lemma 1, which provides the error bound between exact and approximate distances, for distance correction. However, Lemma 1 only applies when the matrix is random, and it prevents the use of the optimal projection (PCA) matrix. Fortunately, Equation 2 also provides the error, which can be treated as a random variable. To this end, we propose using error quantile (e.g., 99.5%) to get the error bound. The error quantile/bound can be calculated from the inverse function of the Cumulative Distribution Function (CDF), allowing a trade-off between efficiency and accuracy.

Error Quantile. ADSampling derives error bounds from random projections to determine when more precise distance calculations are necessary as a corrective measure. However, the way ADSampling derives error bounds is not applicable to the PCA projection. To obtain an error bound for PCA



(1) DEEP@dim=32 (2) DEEP@dim=128

(3) GLOVE@dim=50 (4) GLOVE@dim=100

Fig. 2: Example of Residual Error Distribution and Empirical Rule of Gaussian Distribution on DEEP and GLOVE dataset

projections, we analyze the distribution of the estimation error, denoted as $\epsilon = dis' - dis$. This error can be expressed as $\epsilon = -2 \cdot \langle \mathbf{q}_r, \mathbf{x}_r \rangle$, as shown in Equation 2, representing the difference between exact and approximate distances. Recall that the distribution of ϵ follows a Gaussian distribution, we rewrite as $\epsilon \sim \mathcal{N}(0, \sigma^2)$, where the variance σ^2 is computed using Equation 3.

Given this distribution, values corresponding to specific quantiles can be computed. For instance, according to the empirical rule for Gaussian distributions, the 99.7% quantile corresponds to three standard deviations from the mean. This quantile value can be used as the error bound, establishing a direct correspondence between the error quantile and the error bound. Specifically, given a desired quantile, the value $m \cdot \sigma$ corresponding to that quantile can be computed and treated as the error bound.

Therefore, similar to ADSampling, if $dis' - m \cdot \sigma > \tau$, where τ represents the maximum distance (threshold) of a queue Q, we conclude that $dis > \tau$ and $m \cdot \sigma$ can be replaced by the corresponding quantile calculated by numerical method. This makes excluding points from the queue Q reliable. Otherwise, if the approximate distance does not satisfy this condition, it is insufficient to determine the exclusion of a point from the queue Q. In such cases, we mandate sampling additional dimensions for distance correction.

However, PCA projection only minimizes the variance of the error. We prove the error quantile value is also minimized via minimizing error variance by Lemma 2. Lemma 2 shows that the quantile approach has proven to be optimal under the Gaussian distribution, and thus is highly effective in practice.

Lemma 2 ([25]). *Under the Gaussian distribution assumption, when the variance of the error is minimized, the right quantile of the error is also minimized.*

Empirical Analysis. We further analyzed the error distribution of a real-world dataset, as shown in Fig. 2. Specifically, we compared the Gaussian Empirical Rule ($\mu \pm 3\sigma$, indicated by the red line) and Chebyshev's inequality ($\mu \pm 10\sigma$, indicated by the yellow line) against actual data from the DEEP1M dataset. The results, depicted in Fig. 2, indicate that the Empirical Rule more accurately approximates the 99.7th percentile (blue line) of the DEEP dataset. It can also be seen that some datasets have a certain gap with the Gaussian distribution. We use the learning-based approach in Sec V to deal with this.

Algorithm 1: DDC_{res} algorithm

Input: Threshold τ, 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'

$C_1 \leftarrow \|\mathbf{x}\|^2 + \|\mathbf{q}\|^2$; // Precompute Once
$C_2 \leftarrow 2 \cdot \langle \mathbf{x}_d, \mathbf{q}_d \rangle$; // Compute On the Fly
$\sigma_r \leftarrow \sqrt{4 \cdot \langle \mathbf{q}_r^2, \sigma_r^2 \rangle}$; // Precompute Once
if $C_1 - C_2 - m \cdot \sigma_r > \tau$ **then**
  return 1 with $dis' = (C_1 - C_2)$;
else
  $C_3 \leftarrow 2 \cdot \langle \mathbf{x}_r, \mathbf{q}_r \rangle$; // Compute On the Fly
  return 0 with $dis = (C_1 - C_2 - C_3)$;

D. Implementation

We explore the integration of the improved distance estimation and correction into current AKNN algorithms to improve their efficiency.

A Basic Distance Computation Method. To minimize the residual dimension error (i.e., estimation error) and obtain the standard deviation of the error from the data distribution, we utilize training data with the same distribution as the queries to generate the data distribution. Based on this distribution, we perform PCA projection and estimate the standard deviation of the residual dimensions. To get the distance bound, we employ the approximate distance $dis' = C_1 - C_2$ minus m times the error standard deviation, resulting in a high probability that the precise distance will be greater than the distance bound within the training data-generated distribution.

The entire process of minimized residual variance-based data-driven distance computation, abbreviated as the DDC_{res} method, is summarized in Algorithm 1. To address the issue of data distribution changes under different search parameters, we perform PCA on the dataset points as an approximation. For the standard deviation, we consider the KNN distribution of the query because it does not change with the search parameters and directly affects search accuracy.

Incremental Optimization. A significant advantage of orthogonal projection lies in its capability to compute projection dimensions incrementally, thereby achieving a more accurate approximation of distance up until the exact distance is determined. Similar to ADSampling, where sample projection dimensions are progressively increased, DDC_{res} also supports the incremental addition of computational dimensions for distance correction. Specifically, for the current approximate distance dis', if DDC_{res} prunes the computation of the exact distance

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

Input: Threshold τ, Multiplier m, Incremental Project dim Δ_d, Transformed query $\mathbf{q}$, Transformed data $\mathbf{x}$
Output: Result: 0 with precise distance dis or 1 with approximate distance dis'

$C_1 \leftarrow \|\mathbf{x}\|^2 + \|\mathbf{q}\|^2$; // Precompute Once
while $d < D$ **do**
  $C_2 \leftarrow C_2 + 2 \cdot \langle \mathbf{x}_{\Delta_d}, \mathbf{q}_{\Delta_d} \rangle$; // Incremental
  $r \leftarrow D - d$;
  $\sigma_r \leftarrow \sqrt{4 \cdot \langle \mathbf{q}_r^2, \sigma_r^2 \rangle}$;
  if $C_1 - C_2 - m \cdot \sigma_r > \tau$ **then**
    return 1 with $dis' = (C_1 - C_2)$;
  else
    $d \leftarrow d + \Delta_d$;

for the current point, the computation halts. Otherwise, if DDC$_{\text{res}}$ does not prune the precise distance computation for the current point, the computation proceeds by incrementally adding dimensions. Subsequently, the new approximate distance is utilized to continue pruning the precise computation until the accumulated dimensions reach the original dimensionality or it is pruned and stopped earlier. We summarize the method of using incremental DDC$_{\text{res}}$ in Algorithm 2.

V. A General Distance Computation

This section addresses the lack of generality associated with ADSampling. In the previous section, we replaced the random projection used in ADSampling with a PCA projection to form our new method, DDC$_{\text{res}}$. However, both ADSampling and DDC$_{\text{res}}$ are limited to projection-based distance estimation. Note that other distance estimation methods, such as product quantization, can outperform projection-based methods in some cases. To make distance computation work for other estimated distances, we introduce a novel data-driven distance correction scheme. Our shemem makes no assumptions about the source of the approximate distance, ultimately providing a general method for distance computation.

A. A Data-Driven Distance Correction

Distance correction plays a crucial role in determining whether the currently computed approximate distance, dis', is sufficiently accurate. In the ADSampling algorithm, if the condition $dis' - \epsilon\tau > \tau$ holds, it is concluded that the candidate point p is unlikely to be inserted into the queue Q. Here, dis' is the approximate distance between the candidate point p and the query point q, and τ is the threshold. The term $\epsilon\tau$ quantifies the error between the approximate and exact distances.

Similarly, in the DDC$_{\text{res}}$ algorithm, if $dis' - m\sigma > \tau$ holds, it is concluded that the candidate point p is unlikely to be inserted into the queue Q because it is highly probable that the exact distance dis exceeds τ. Otherwise, both ADSampling and DDC$_{\text{res}}$ will sample additional dimensions to compute a more precise approximate distance dis' for correction.

Using $dis' - \epsilon\tau > \tau$ in ADSampling and $dis' - m\sigma > \tau$ in DDC$_{\text{res}}$ allows for early stopping, ensuring efficiency. When these conditions are not met, incremental distance correction

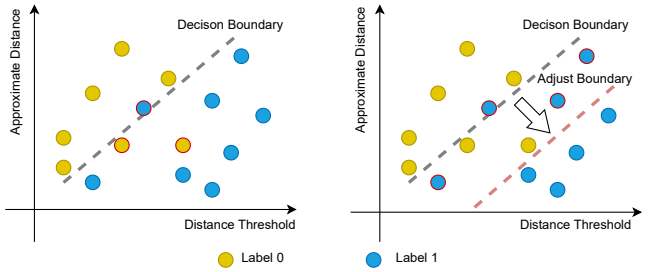
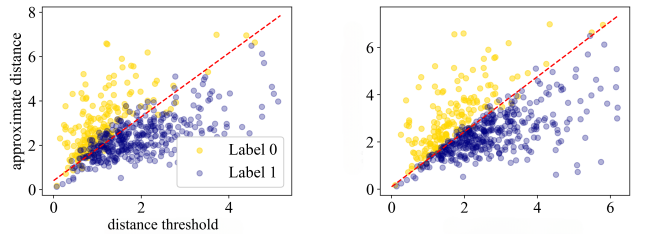


Fig. 3: The Example of Learned Decision Boundary



(1) dim=32 (2) dim=128

Fig. 4: Example of Linear Models with PCA on GIST dataset

is employed to maintain accuracy and prevent a significant drop in performance.

New Distance Correction Scheme. Although DDC$_{\text{res}}$ addresses the limited effectiveness of ADSampling using PCA projection, projection-based approximate distance is not guaranteed to outperform other approximate distances (e.g., product quantization). This raises a critical question: how can we determine the error bound (e.g., $m\sigma$ for DDC$_{\text{res}}$) for distance correction for an arbitrary approximate distance? To address this question, our key insight is to treat the error bound as a parameter. Recall that the purpose of the error bound is to adjust the approximate distance from dis' to $(dis' - \text{parameter})$. In the case of DDC$_{\text{res}}$, the parameter is $m\sigma$. Let's delve deeper into this action.

The parameter(s), or error bound, is required to ensure that $f_{\text{parameter}}(dis') > \tau$ if and only if $dis > \tau$, where $f_{\text{parameter}}$ is the modification of the approximate distance. This implies that when the exact distance is larger than τ, the updated approximate distance (i.e., $f_{\text{parameter}}(dis')$) should also be larger than τ, thereby effectively pruning irrelevant candidates. Thus, finding the error bound becomes the task of finding the parameter(s) that fulfills this condition. In light of this, we propose a data-driven approach to determine the parameter(s) for distance correction.

Data-Driven Parameter Learning. For the generation of training data, we adhere to the same process as with the DDC$_{\text{res}}$ instance. However, considering arbitrary approximate distances, we cannot solely depend on projection methods to minimize error. Instead, we employ a linear model to replace the minimization of errors. Specifically, we utilize features related to the approximate distance to learn a decision boundary for higher pruning efficiency, thereby finding the parameter. Recall that we aim to make $f_{\text{parameter}}(dis') > \tau$ if and only if $dis > \tau$. To achieve an efficient classifier, we use the approximate distance dis' and the threshold τ as

features. We then learn the weights of these two features and the intercept to classify whether dis is greater than τ. The linear model can be expressed as:

$$L = sign(w_1 dis' + w_2 \tau + b > 0)$$
$$= sign(m_1 dis' + \beta > \tau)$$

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

We utilize simple logistic regression with cross-entropy loss trained by SGD as our method for obtaining linear models due to its relatively stable performance and extremely high training efficiency. This efficient learning process yields the parameters, specifically m_1 and β, which define our function $f_{\text{parameter}}()$.

Adaptive Adjustment. With the initial learned decision boundary $m_1 \cdot dis' + \beta$, we can employ numerical methods such as binary search for the intercept term β' (similar to error quantile) to ensure that $dis > m_1 \cdot dis' + \beta'$ holds with a manageable small failure probability. Yet, the failure probability of equation $dis > m_1 \cdot dis' + \beta'$ does not directly impact search accuracy. We aim to obtain β' corresponding to the recall target for a specific AKNN search accuracy.

Using the Binary Cross Entropy (BCE) as the loss function, we can set the threshold τ as the k-NN distance of the query where Label 0 data becomes the KNN of the query. Then, we can perform a binary search on β' to ensure that the learned instance achieves the target recall r for Label 0 in the training dataset, thereby facilitating automatic parameter configuration.

Example 1. *Fig. 3 illustrates the decision boundary of the learning-based approach. Data points with red edges represent misclassified instances. The classification recall of Label 0 directly impacts the accuracy of the AKNN search, while the recall of Label 1 influences the search efficiency by determining the extent of precise distance computation required. To optimize the AKNN search accuracy, we adjusted the intercept of the linear model. This adjustment intentionally sacrifices some accuracy for Label 1 to ensure that the accuracy for Label 0 meets the necessary requirements. This trade-off is crucial for maintaining the desired performance of the AKNN search.*

Remark. The learning-based approach is comparable to the DDC_{res} approach described in Section 4 by setting $m_1 = 1$ and $\beta = m \cdot \sigma$. However, this learning-based approach is more flexible because it can handle any approximate distance, making it more general.

B. Implementation

Approximate Distance and Feature Select. For projection-based approximate distance, we use a simple PCA projection as the approximate distance without applying the decomposition by Equation 2. This general case is denoted as DDC_{pca}. For another popular approximate method, product quantization distance, we utilize the distance to quantized centroids of the query q to the database point u, known as the Asymmetric Distance Computation (ADC), as the approximate distance dis'.

For more precise distance approximation, we use OPQ [35] as our final quantized approximate distance method. OPQ employs an orthogonal matrix to rotate the space to minimize the quantization error, and we denote this method as DDC_{opq}. Additionally, for quantization methods, we can also use the distance from u itself to the quantized centroid as an **additional feature**. This multi-feature approach can further enhance the effectiveness of the linear model.

Moreover, we observed that product quantization exhibits different optimizations depending on the hardware environment [36], [6]. In our experimental process, we do not consider these hardware optimizations and focus solely on the algorithm itself.

Multiple Linear Model. We can also employ multiple linear models with the same principle as in ADSampling or Algorithm 2. These models, denoted as $L_1, L_2, \ldots, L_n$, each correspond to a unique projection dimension, as illustrated in Figure 4. Each model has an associated error probability $e_1, e_2, \ldots, e_n$ for Label 0 classification. If the projected distance dis' for the current dimension is classified as Label 0 by the current linear model L_i, we increment the projection dimension. This process continues until the projection dimension equals the original dimension, at which point we obtain an accurate distance. If this distance is less than the threshold τ, we update the result queue.

To achieve the target recall r for Label 0, it is essential to consider the number of false positives (FP) for each linear model. A straightforward strategy is to set the recall target for each of the n linear models as $r_i = 1-(1-r)/n$. This ensures that the overall recall target satisfies the recall requirement for Label 0. In line with ADSampling, we also assign a corresponding classifier L_i for every Δ_d dimension. The target recall for each classifier is set as $r_i = 1-(1-r)/(D/\Delta_d)$. If the current classifier result is 0, we continue to calculate the projection distance and use the next classifier until the exact distance is determined. Otherwise, we prune this candidate.

C. Discussion of Out-of-Distribution Query

In real-world applications, queries may differ from the existing distribution. Such out-of-distribution (OOD) queries pose a challenge for all learning-based algorithms. Fortunately, our DDC_{res} algorithm treats the query as a deterministic variable when computing the error bound, making it effective on OOD queries in our report [25]. For linear model-based methods like DDC_{opq} and DDC_{pca}, we observed poor performance on OOD queries [25]. To handle such cases, we found that retraining with approximately 100 OOD queries effectively prevents performance degradation.

VI. Time and Space Analysis

In this section, we analyze the time and space complexity of our distance computation method. Depending on the underlying approximate distance, we derive different cost analyses.

TABLE II: 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
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

A. Projection Distance

One advantage of orthogonal projection is the ability to incrementally add dimensions until the precise distance is computed or early terminated according to the error bound. Then, we can analyze the average scan dimensions for the projection-based method in our experiments to verify the efficiency. Moreover, the DDC_{res} and DDC_{pca} methods introduce additional time due to the need to rotate the space. For a single query, the time complexity to perform matrix multiplication for the projection (rotation) of the query is $O(D^2)$. In practice, such rotation only costs 3% in high recall AKNN search scenarios [25].

B. Quantization Distance

The quantization distance brings additional time cost consisting of both the opq rotation $O(D^2)$ and the construction of a lookup table. When OPQ is implemented across m subspaces, each containing 2^{nbit} quantized centroids, the time complexity of constructing a lookup table is $O(D \cdot 2^{nbit})$. With the lookup table, the calculation of the asymmetric distance requires only m table lookups. We evaluate the pruned rate by DDC_{opq} to evaluate its efficiency. The extra space cost is $n \cdot m \cdot nbit$ bits for m subspaces. Here, m is taken as $1/4$ or lower of the original dimension D, resulting in an extra space cost of $1/32$ dataset space with float32 vector.

VII. Experiments

A. Experimental Settings

Datasets. We employ eight publicly available datasets of different scales and source types, as outlined in Table II. These datasets have been extensively utilized as benchmarks for evaluating AKNN search algorithms. It is important to note that these publicly available datasets encompass base vectors and query vectors. For datasets that provide learning data, such as GIST and DEEP, we directly utilize the provided learning data. However, for datasets that do not provide learning data, we randomly sample 100,000 instances from the base data to train the linear model and then remove them from the base data. Note that all vector data in the experiment are stored in float32 format.

Performance Evaluation. We employ recall@K defined as $\frac{|T \cap G|}{K}$ as the metric to evaluate search accuracy, where G is the set of ground-truth K-nearest neighbors of the query in the dataset S and T is the AKNN algorithm search result set. To gauge efficiency, we utilize query-per-second (QPS), which measures the number of queries processed per second, including the end-to-end query time, Additionally, we evaluate the total number of dimensions scanned by the projection-based method. For OPQ we use the pruning rate to evaluate the efficiency. All the metrics mentioned are averaged over the entire query set.

Compare Method. We list the compared method below:

- HNSW: HNSW with all precise distances computed.
- HNSW++: HNSW with ADSampling method.
- HNSW-DDC_{opq} : HNSW takes OPQ as the approximate distance and learned boundary.
- HNSW-DDC_{pca} : HNSW takes PCA as the approximate distance and learned boundary.
- HNSW-DDC_{res} : HNSW takes residual dimension variance for pruning.
- FINGER [29]: Estimating angles between HNSW neighboring residual vectors for pruning.
- IVF: IVF with all precise distances computed.
- IVF++: IVF with ADSampling method.
- IVF-DDC_{opq} : IVF takes OPQ as the approximate distance and learned boundary.
- IVF-DDC_{pca} : IVF takes PCA as the approximate distance and learned boundary.
- IVF-DDC_{res} : IVF takes residual dimension variance for pruning.

Index Configuration. We mainly consider the index construction of two AKNN algorithms, HNSW, and IVF. For HNSW, two key parameters control the graph construction: M determines the number of connected neighbors, and $efConstruction$ controls the quality of the approximate nearest neighbors. Following the original HNSW work, we set $M = 16$ and $efConstruction = 500$. For IVF, as recommended in the Faiss library, the number of clusters should be around the square root of the database size. We set the number of clusters to 4,096 as in ADSampling. All C++ code compiles with g++ 11.4.0 and -O3 optimization and disables all SIMD operations with the same as ADSampling. Experiments use an Intel(R) Xeon(R) Platinum 8352V CPU @ 2.10GHz with 512GB memory, running in Ubuntu Linux.

Training Configuration. For the training of the linear models, we directly take the KNN of learning data as label 0. For label 1, we accumulated 500,000 items from the query process as negative samples. We compute all training items' approximate distance, threshold, and other features offline to train the linear classifier with BCE as the loss. Specifically, we use 10,000 learning vector data for each dataset as training queries and perform the search algorithm with a fixed configuration to get the training data. We set the recall target r as 0.995 for the time-accuracy tradeoff experiment and provide a verified recall target experiment as follows.

Approximate Distance Configuration. For the random projection approach, we set $\epsilon_0 = 2.1$ and $\Delta_d = 32$ which is recommended as the best performance in ADSampling. For

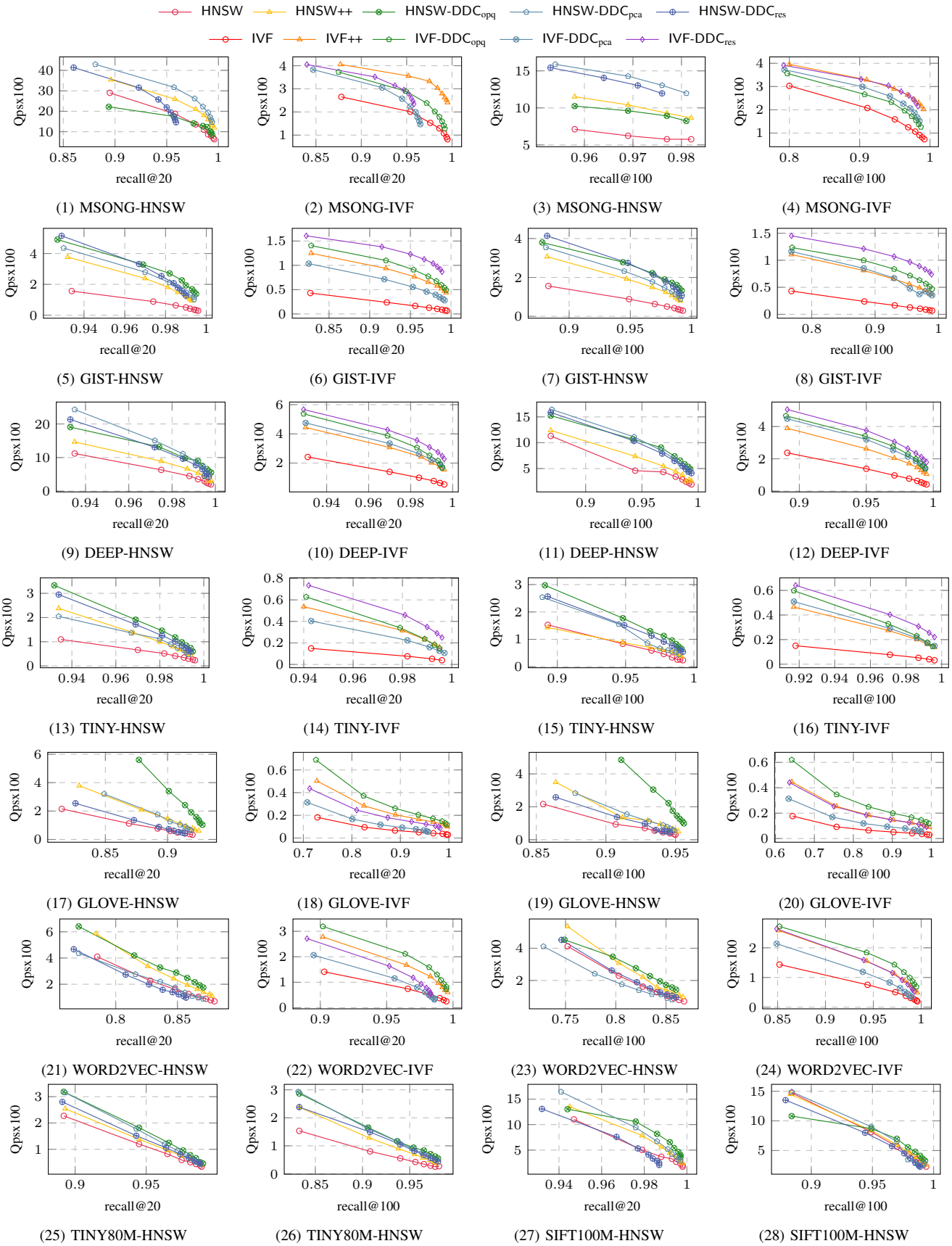


Fig. 5: Time-Accuracy Tradeoff (HNSW and IVF)

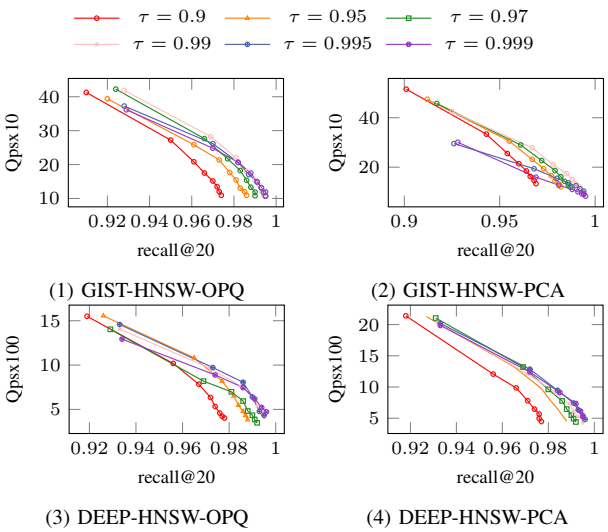


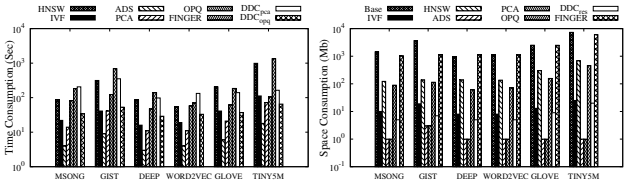
Fig. 6: Parameter Study on Target Recall τ

$\mathsf{DDC}_{\text{res}}$ and $\mathsf{DDC}_{\text{pca}}$, we also set every $\Delta_d = 32$ dimension to achieve the same condition as ADSampling. For the $\mathsf{DDC}_{\text{opq}}$ approach, we set the subspace number as $d/8$ for the GIST dataset and $d/4$ for the others since all the dataset dimensions can be divided by 4. The $nbits$ for $\mathsf{DDC}_{\text{opq}}$ is set as 8, the default parameter. The features used for the $\mathsf{DDC}_{\text{pca}}$ approach are the project distance(approximate distance) and the threshold. For the $\mathsf{DDC}_{\text{opq}}$ approach, we added an additional point-to-centroid distance as a feature. The target recall is set as 0.995 for the $\mathsf{DDC}_{\text{pca}}$ and $\mathsf{DDC}_{\text{opq}}$ methods. For the multiplier m for $\mathsf{DDC}_{\text{res}}$, we set it as 8 for SIFT, GIST, and DEEP, 12 for TINY and WORD2VEC, and 16 for the GlOVE dataset. For the case of multiple classifiers, we set the target recall for each classifier based on Δ_d as $r_i = (1-(1-r)/(D/\Delta_d))$.

B. Experimental Results

Overall Results. We plot the time accuracy curve with two popular algorithms HNSW and IVF in Fig. 5 which the upper right is better. To achieve the tradeoff between time-accuracy, we varying N^{ef} for HNSW, HNSW++, $\mathsf{HNSW\text{-}DDC}_{\text{opq}}$, $\mathsf{HNSW\text{-}DDC}_{\text{pca}}$, $\mathsf{HNSW\text{-}DDC}_{\text{res}}$ and N^{probe} for IVF, IVF++, $\mathsf{IVF\text{-}DDC}_{\text{opq}}$, $\mathsf{IVF\text{-}DDC}_{\text{pca}}$, $\mathsf{IVF\text{-}DDC}_{\text{res}}$. As the focus of our approach, we consider the high recall($> 85\%$) as the main scenario that the approximate distance methods without correction cannot achieve. We observe the following results. (1) From the overall experimental result, the data-driven distance approximation method can achieve a large margin speed-up while maintaining high recall. (2) The PCA-related method performs better on image datasets because the variance of image datasets is more skewed. For example, PCA projection to 32 dimensions can retain 67% of the variance of the GIST dataset and 82% of the variance of SIFT, while WORD2VEC and GLOVE only retain 36% and 18% respectively.

Results of Verified Target Recall. We study the parameter of target recall with different approximate methods. The actual threshold is greater than the threshold for training. The overall



(1) Index-Time (2) Index-Size

Fig. 7: Index Size and Time

performance of the recall will be better than the target recall since the update threshold will be larger than the ground truth threshold. As in the parameter study of r in Fig. 6, we found that in the case of $r = 0.995$, the search algorithm of both IVF and HNSW can achieve the best tradeoff between efficiency and recall loss(less than 0.5%) which is selected as the default target recall.

Results of Pre-Processing Time and Index Size. We present the preprocessing time and index space in Fig. 7. Here, ADS represents the additional space-time required by ADSampling , while $\mathsf{DDC}_{\text{pca}}$ and $\mathsf{DDC}_{\text{opq}}$ denote the training time for linear classifiers with HNSW. The preprocessing time for $\mathsf{DDC}_{\text{res}}$ only includes the PCA time (including variance computation). Meanwhile, the additional space required by $\mathsf{DDC}_{\text{pca}}$ and ADSampling , which is only for the projection matrix (D^2 floats), can be considered negligible. $\mathsf{DDC}_{\text{res}}$ requires additional storage for the norms of N vectors, and $\mathsf{DDC}_{\text{opq}}$ needs to store quantized vectors that grow linearly with the size of the dataset.

Results of Comparing with FINGER. We conducted an efficiency comparison between our algorithm and FINGER under the same environment (with SIMD acceleration) as shown in Fig. 8. It is observable that the $\mathsf{DDC}_{\text{res}}$ algorithm is 20% to 30% faster than FINGER on the GIST and DEEP datasets in scenarios requiring high recall. Concurrently, the ADSampling exhibits inferior performance in a hardware-accelerated environment. More experimental results can be found in our technical report [25].

Results of Scalability Test. We conducted scalability tests on the SIFT100M dataset, where PCA includes the computation of data variance. It is observed that the preprocessing time for approximate distances, such as ADS, PCA, and OPQ, only accounts for 1%-5% of the total time required for constructing large-scale data indexes. Moreover, the training of linear models, including the generation of test data, grows linearly with the size of the dataset. $\mathsf{DDC}_{\text{pca}}$ and $\mathsf{DDC}_{\text{opq}}$ require preprocessing times of 105(s) and 107(s), respectively, for a dataset size of 20 million, and 254(s), 504(s) for 100 million vectors. This indicates that our method can enhance search efficiency at a cost of less than 10% of the index construction time.

Results on Large Scale Dataset. We test the performance of our algorithm on a large-scale data set. We build the HNSW index and test ADSampling and our method on it. Although the large-scale dataset has relatively low dimensions (128, 150),

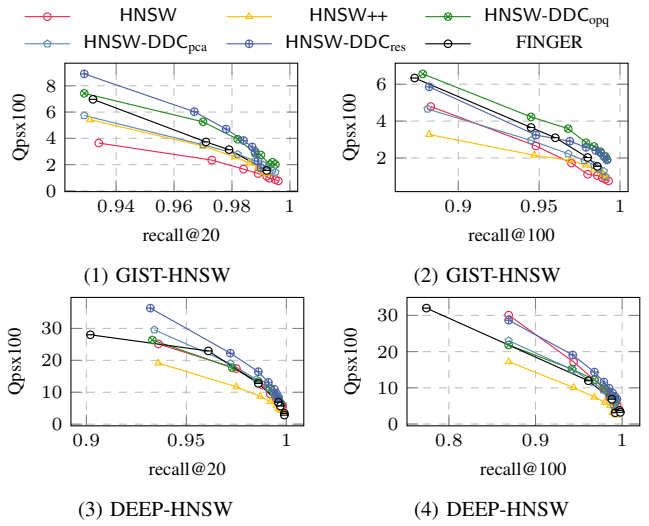


Fig. 8: Time-Accuracy Tradeoff with FINGER

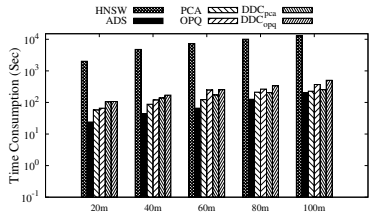


Fig. 9: The Scalability Test on SIFT100M

as shown in Fig. 5. (25-28), our approaches still achieve a 2x speedup compared to HNSW and a 1.45x speedup compared to ADSampling on the TINY80M dataset. It is worth noting that the training of PCA and OPQ, as well as linear models, is independent of the data scale. For large-scale datasets, we sample 1 million points for the training of the PCA matrix. As for the OPQ matrix, we follow the design of the Faiss library [37], which involves sampling 65,536 points from the database for OPQ training.

Results of Scan Dimensions. The scan dimensions can be evaluated using the projection-based method. We plot the average scan dimension ratio compared with the naive method plot(red line) to the ADSampling method(orange line) and DDC_{pca} method(green line) with the left axis in Fig. 10. For the DDC_{opq} method, we plot the pruning rate with the right axis and blue line to verify its efficiency. The query performance corresponds to the time-accuracy tradeoff in Fig.5. When N^{ef}=2000, DDC_{res} scans only 7% of the total dimensions, while DDC_{pca} scans 15% of the dimensions, and ADsampling scans 26% on the GIST dataset.

Results of Security Search Scenarios in Ant Group. Our method has been validated in real-world scenarios at Ant Group. On a 1 million secure dataset with 512 dimensions, the DDC method reduces retrieval time by 35% and increases throughput by 55.25% without compromising accuracy. Additionally, the linear classifier is used during the re-ranking process in DiskANN [38] to prune distance computation and it improved search efficiency by 50% under at 90% recall on

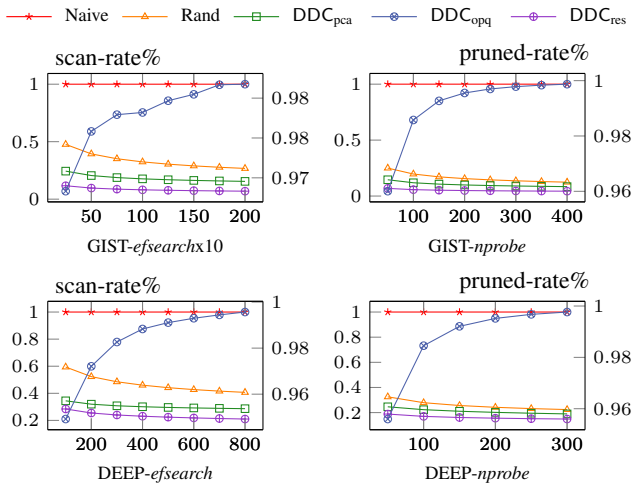


Fig. 10: Dimension Scan Rate and Pruned Rate

TABLE III: Approximate Distance 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

SIFT dataset.

Results of Distance Approximation. We test the accuracy of our approximate distance method without considering the index, by scanning the entire dataset to find the Top K approximate nearest neighbors (AKNN). Table III verifies the search accuracy of different projection methods when projecting to 32 dimensions for approximate distance. The results indicate that DDC_{res} achieves higher accuracy than random projection and is superior to PCA in most datasets. Notably, the DDC_{res} method performs almost comparably to the PCA approach on the Deep dataset, as the vectors are normalized.

VIII. Conclusion

In this paper, we propose an effective and general distance computation method for AKNN search. First, to improve the effectiveness of ADSampling, we analyze the estimation error between the approximate and the exact distances. We show that PCA with distance decomposition instead of random projection leads to more accurate distance estimates. Next, to improve the generalizability of ADSampling, we design a data-driven approach to perform distance correction that is not constrained by the source of the distance estimation. Extensive experiments on real-world datasets confirm that our proposed method outperforms ADSampling in search speed. Note that our method can be used for maximum inner product search by taking off the vector norm term of distance estimation. Future work will investigate more precise distance estimation methods.

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] 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

[26] 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.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[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.

[38] S. Jayaram Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi, "Diskann: Fast accurate billion-point nearest neighbor search on a single node," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[39] Z. Wang, H. Xiong, Z. He, P. Wang *et al.*, "Distance comparison operators for approximate nearest neighbor search: Exploration and benchmark," *arXiv preprint arXiv:2403.13491*, 2024.

APPENDIX

Proof of Lemma 2

Proof. 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

Proof. 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. Since 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 then 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,

$$\begin{aligned}
\frac{\partial J(\mathbf{w}_1)}{\partial \mathbf{w}_1} &= 2\Sigma\mathbf{w}_1 - 2\lambda_1\mathbf{w}_1 = 0 \\
\mathbf{w}_1^T\Sigma\mathbf{w}_1 &= \lambda_1
\end{aligned}$$

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. □

Verification of Empirical Rule of Gaussian distribution

We studied the actual error distribution on the DEEP1M dataset and validated the differences between the Empirical Rule of Gaussian distribution and Chebyshev's inequality on real-world datasets. As shown in Fig. 11, the blue line represents the 99.7th percentile on the right side of the mean distance (half of the data), corresponding to the interval $\mu \pm 3\sigma$ for Gaussian distributed data. The red line represents three times the standard deviation from the mean distance on the right side, corresponding to the decision boundary using the empirical rule. The yellow line represents ten times the standard deviation, corresponding to the decision boundary (cover 99% data) obtained by Chebyshev's inequality. It can be observed that on real-world data, the Empirical Rule for Gaussian distribution is closer to the actual quantile, whereas Chebyshev's inequality exhibits a larger discrepancy. Thus making the decision boundary obtained using the Empirical Rule more suitable for adjusting approximate distances

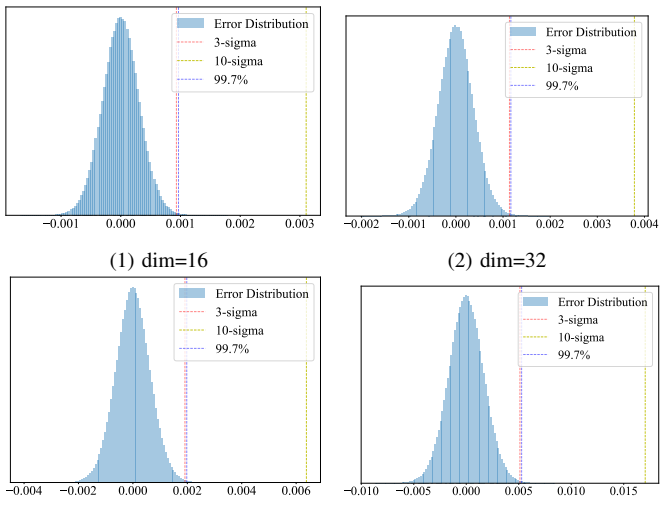


Fig. 11: Example of Residual Error Distribution and Empirical Rule of Gaussian Distribution on DEEP1M dataset

Results for Evaluating the Distance Approximation. We then study the method with only approximate distances without error quantile or bound. We take the OPQ with HNSW and IVF algorithms notes as HNSW_{opq} and IVF_{opq}. The approximate distance method with the parameter 120 subspaces for the GIST dataset and 64 subspaces for the DEEP dataset is the same parameters as $\text{HNSW-DDC}_{\text{opq}}$ and $\text{HNSW-DDC}_{\text{opq}}$. It can be seen that there is about a 7% to 9% recall gap between the OPQ-only method(HNSW_{opq}, IVF_{opq}) to the

leaned distance correction one. The approximate distance with PCA at the same setting will become even worse as the OPQ is more accurate than PCA in various data settings.

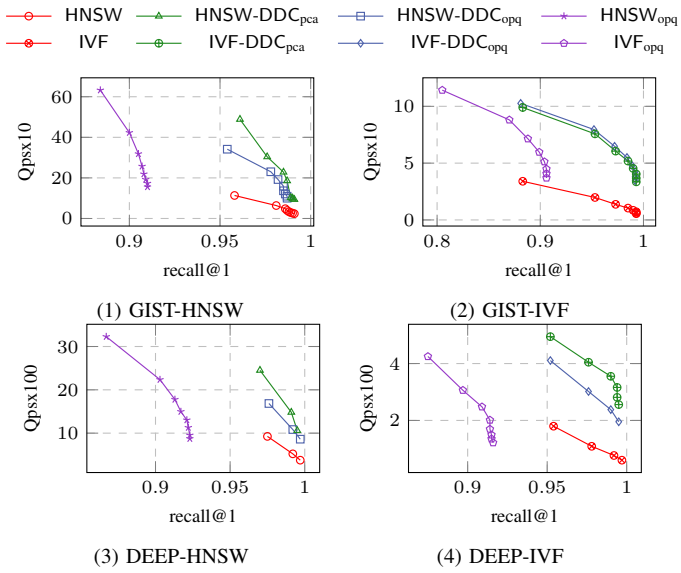


Fig. 12: Recall@1 Time-Accuracy Tradeoff

Results of Extra Time. Upon receiving a query, the search algorithm executes the transformation first. We implement this process via a matrix multiplication operation(with C++ Eigen Library), which takes 0.344ms in the GIST dataset. Specifically, on the GIST dataset, the time cost for projection accounts for only 6% of the total cost when achieving a recall rate greater than 90%, and a mere 3% at a recall rate of 95%. Additionally, the time complexity associated with computing the Look-up table constitutes only 1% to 2% of the total time.

Results of Inner Product. It is worth noting that our DDC$_{res}$ method can be trivially used for maximum inner product search by taking off the norm term of the distance estimation. Then, the (exact) inner product is decomposed as:

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

The error term is the same as the origin DDC$_{res}$ method which the approximate distance can be computed by $\langle \mathbf{q}_d, \mathbf{x}_d \rangle$ with $O(d)$ time complexity. Then we still perform PCA projection to minimize the error term variance and leverage the Gaussian assumption to prune precise distance computation. The result is shown in Fig. 13 and the DDC$_{res}$ is 1.5x to 2x faster than the origin HNSW algorithm of the DEEP dataset.

Results of Out-of-Distribution Query. Out-of-distribution (OOD) data presents a critical challenge for all data-driven (learning) methods. To evaluate the performance of our method on OOD queries, we simply added Gaussian noise to the original queries and conducted experimental tests. As shown in the experimental results in Fig. 14, our algorithms DDC$_{res}$ and ADSampling are less affected by out-of-distribution queries. This is because ADSampling is a data-free method, with the error bounds provided by the JL-lemma derived from

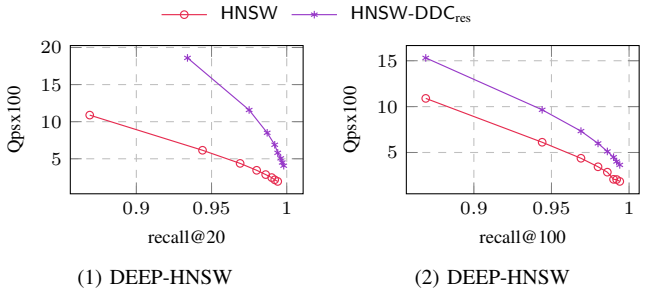


Fig. 13: Time-Accuracy Tradeoff of Inner Product

random matrices. On the other hand, the DDC$_{res}$ algorithm treats the query as a deterministic variable, which also makes it effective for OOD queries. However, the DDC$_{pca}$ and DDC$_{opq}$ algorithms, either have low efficiency or low recall in such cases. A straightforward idea is to retrain the linear classifier. Using 100 queries and approximately 10 seconds, we can retrain the model. Experimental results show that this retraining strategy can prevent the accuracy drop of the linear model in OOD scenarios.

Result of SSE and AVX. Experimental results in Fig. 16 demonstrate that, in environments with SIMD hardware acceleration instruction sets, employing OPQ with the trained classifier has yielded the best performance across nearly all datasets. The drawback of OPQ is that it requires a relatively long training time($\approx 300s$ for the GIST dataset where PCA only takes $\approx 30s$) and incurs some additional space overhead (which is not with the Projection-based method), specifically requiring an additional 1/32 to 1/64 of the space consumed by the vector dataset (excluding the index) when using 32-bit floats.

Compare with FINGER. We implemented FINGER and integrated it into the same environment as ADsampling and the results are shown in Fig. 17. Surprisingly, the performance of FINGER exhibited significant differences compared to ADsampling without specific hardware acceleration [29], which was also confirmed in recent benchmarks [39]. Additionally, from recent benchmarks, we discovered that the pruning performance of FINGER is inferior to the PCA method. On the contrary, our DDC$_{res}$ method performed better than the PCA-based method DDC$_{pca}$ on datasets such as GIST.

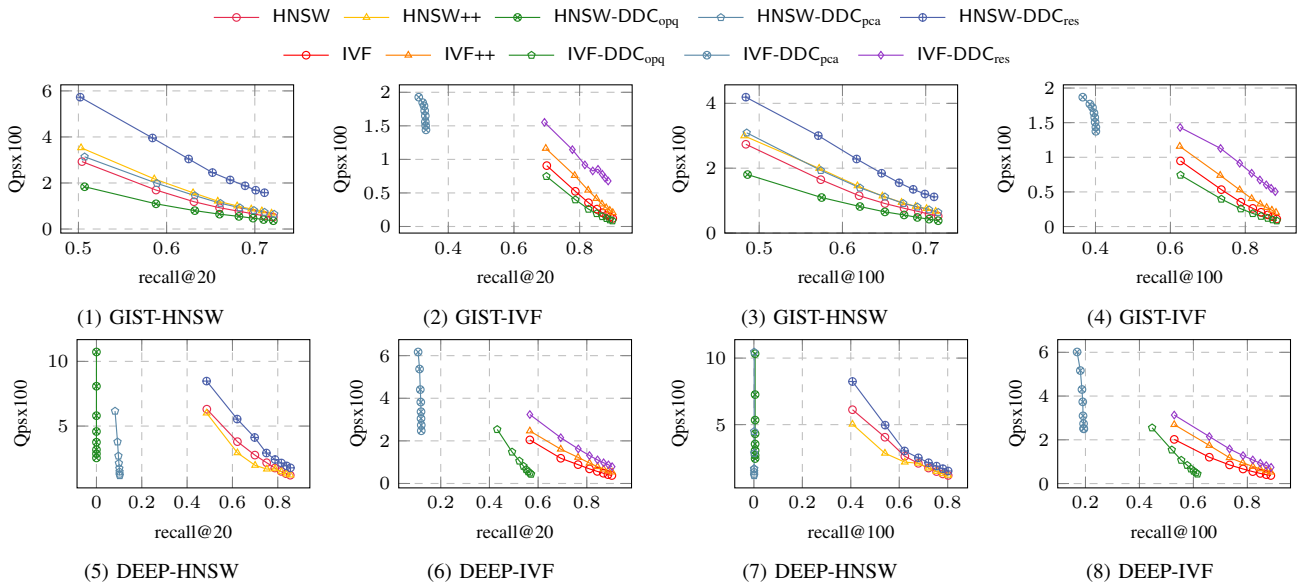


Fig. 14: Time-Accuracy Tradeoff with OOD Query

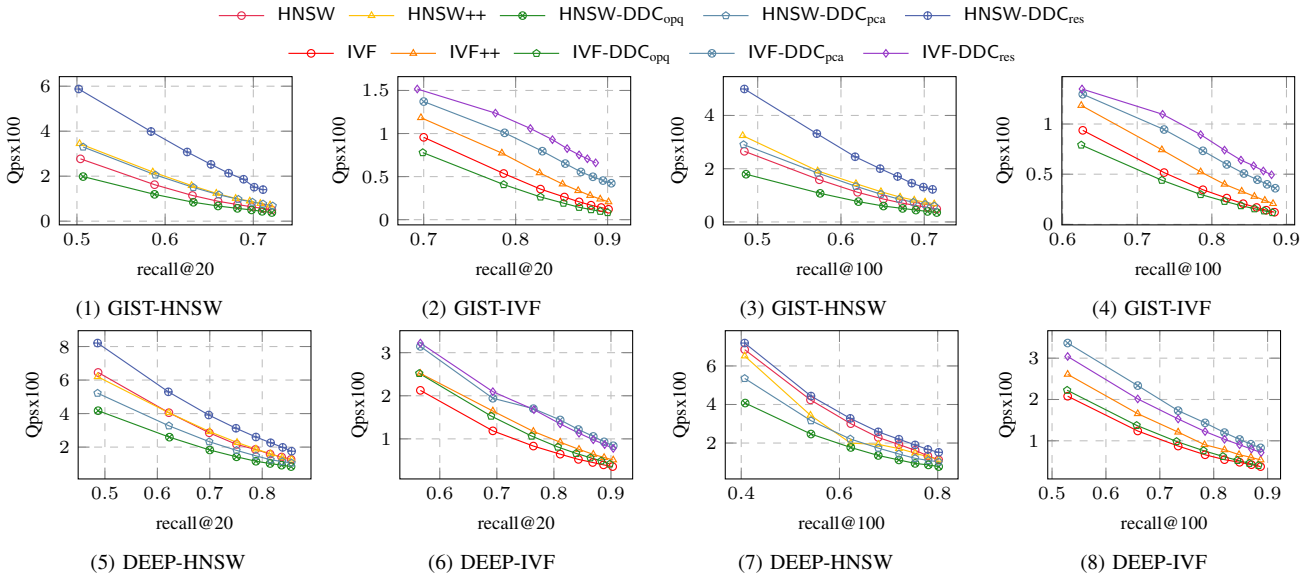


Fig. 15: Time-Accuracy Tradeoff with OOD Query and Retrain Model

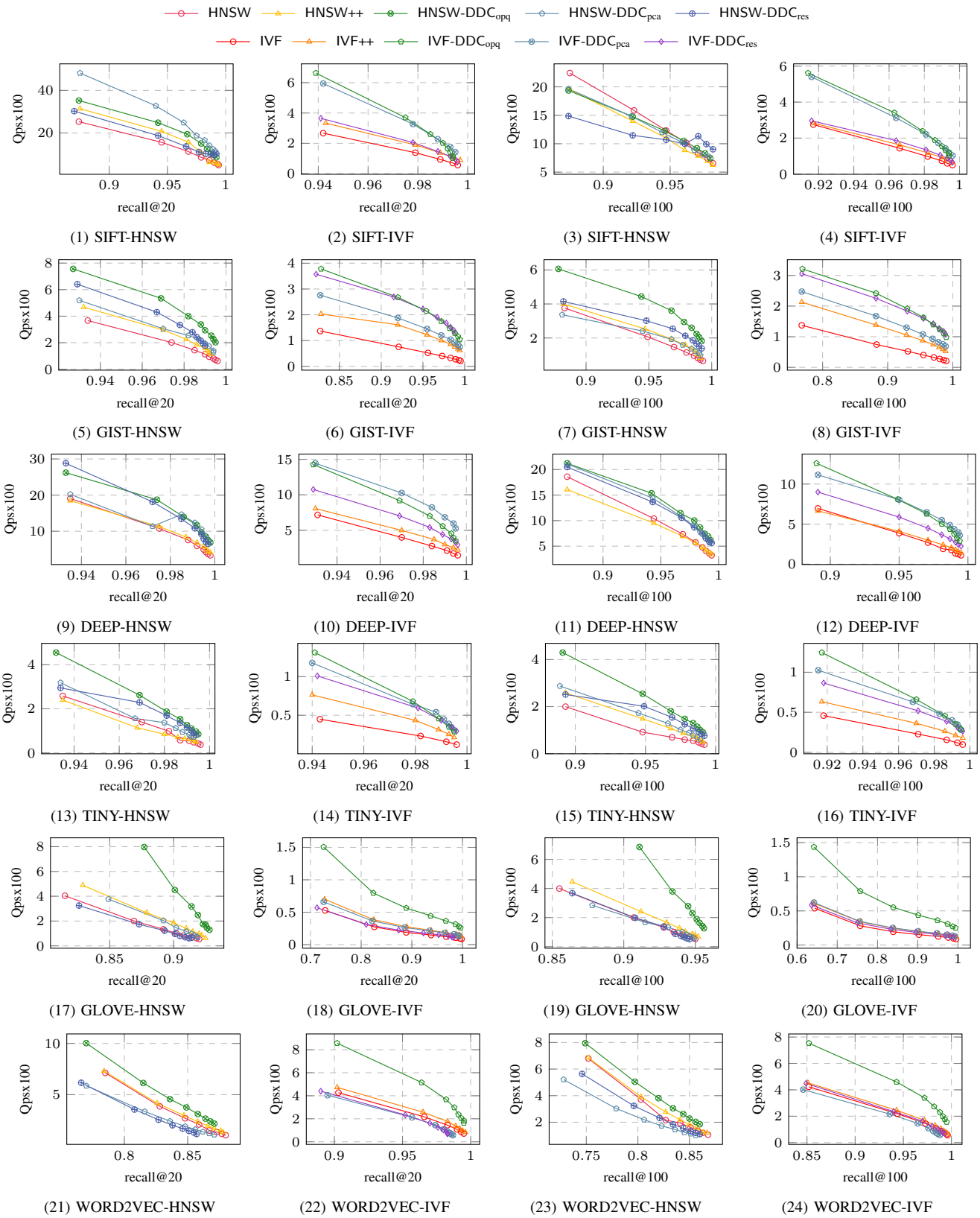


Fig. 16: Time-Accuracy Tradeoff and Dimensionality (SSE)

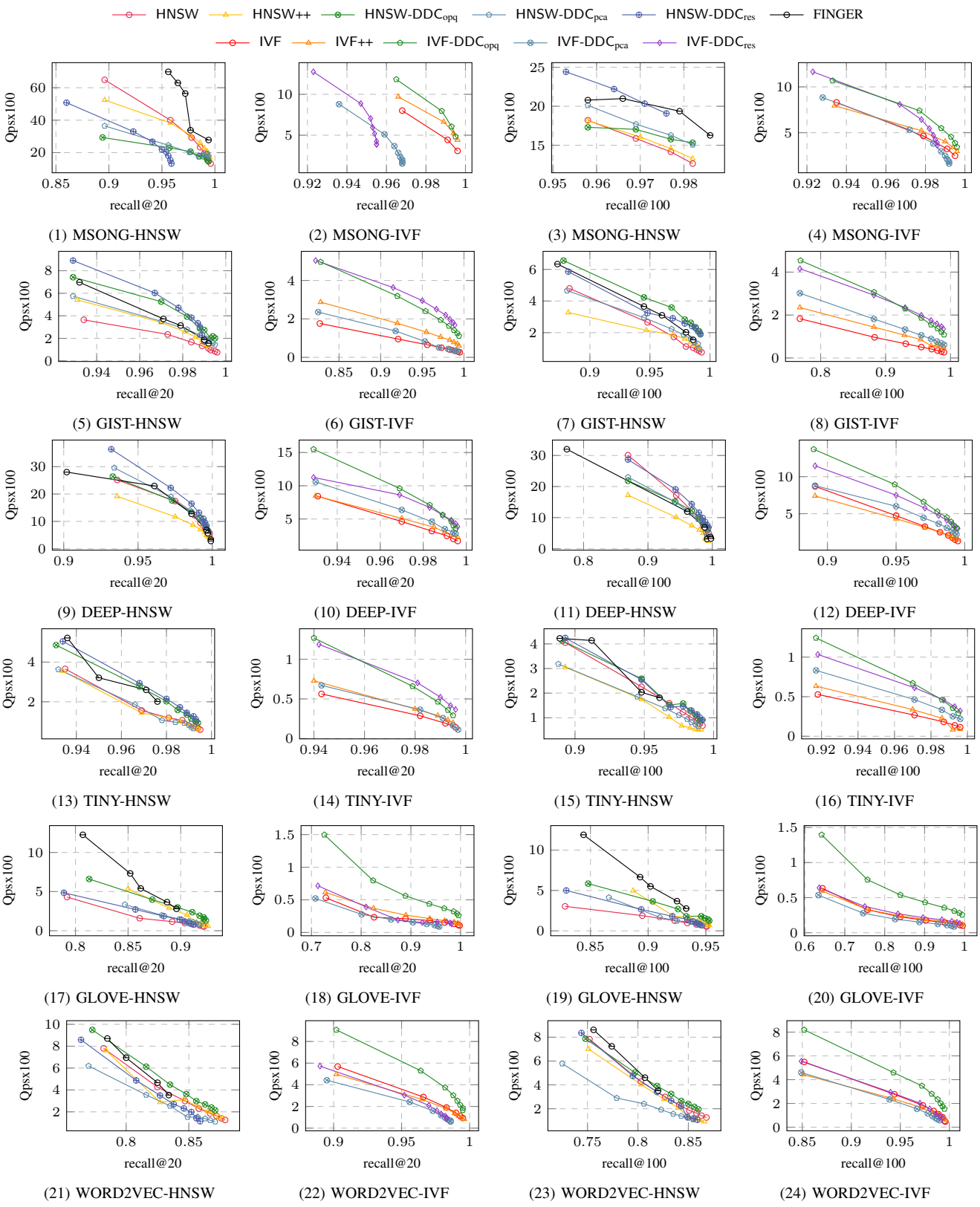


Fig. 17: Time-Accuracy Tradeoff (HNSW and IVF) with SIMD-AVX512