# ATLAS: The Landscape of Approximate Similarity Search — Two Decades of Algorithmic Advances: [Experiments & Analysis]

Anonymous Author(s)

## ABSTRACT

*Similarity search* methods enable efficient retrieval of vectors similar to a given query and play a central role in wide applications, including retrieval-augmented generation, pattern recognition, and recommendation systems. Among the variants, *approximate similarity search* methods have gained prominence over the past decades, achieving high accuracy and greatly improved efficiency compared to exact methods, while occasionally retrieving non-true neighbors. Despite substantial progress, existing studies suffer from major limitations: (i) omission of key algorithmic families; (ii) overlooking recent methodological advances; (iii) lack of rigorous statistical validation; and (iv) limitation of datasets reflecting neural network representations. To address these gaps, and motivated by the growing interest in vector databases—we introduce **ATLAS**, the most comprehensive study of approximate nearest neighbor search methods, covering more than two decades of advancements. Specifically, our contributions are fourfold: (i) a systematic review of five major algorithmic categories; (ii) a large-scale evaluation of 45 methods across 58 datasets; (iii) the introduction of a new measure to quantify the speedup–recall trade-off; and (iv) robust statistical analysis to ensure the reliability of the findings. Our findings reveal four key insights: (i) modern quantization-based methods achieve query efficiency comparable to graph-based algorithms while requiring substantially less memory; (ii) across four method categories, previously unreported top performers emerge, with two showing statistically significant improvements; (iii) hardware-accelerated methods exhibit strong processor-specific dependencies, limiting generalization across architectures; and (iv) both indexing and hardware acceleration yield substantial throughput gains but at the cost of reduced accuracy compared to their vanilla counterparts. These findings reshape the ANNS landscape, reveal previously unexplored insights, and lay the foundation for future research directions.

## KEYWORDS

Similarity Search, Approximate Nearest Neighbor Search

**ACM Reference Format:**

Anonymous Author(s). 2018. ATLAS: The Landscape of Approximate Similarity Search — Two Decades of Algorithmic Advances: [Experiments & Analysis]. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX)*. ACM, New York, NY, USA, 15 pages. https://doi.org/XXXXXXX.XXXXXXX



*Conference acronym 'XX, June 03–05, 2018, Woodstock, NY*



ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XXXXXXX.XXXXXXX

**Table 1: Comparison of existing approximate nearest neighbor search studies with ATLAS, which provides the most extensive evaluation to date by encompassing the widest range of methods, datasets, and rigorous statistical analyses.**

| Benchmarks | # Methods | # Datasets | Statistical Analysis | Methodology | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Quantization | Partition | Hash | Index | Graph | HW-Accelerated |
| Aumüller *et al.* [7] | 14 | 7 | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| Li *et al.* [68] | 19 | 20 | ✗ | ✓ | ✓ | ✓ | ✗ | ✓ | ✗ |
| Echihabi *et al.* [30] | 10 | 4 | ✗ | ✓ | ✓ | ✓ | ✓ | ✓ | ✗ |
| Wang *et al.* [101] | 15 | 20 | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| Azizi *et al.* [8] | 13 | 10 | ✗ | ✗ | ✗ | ✗ | ✗ | ✓ | ✗ |
| **ATLAS (Ours)** | **45** | **58** | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

## 1 INTRODUCTION

*Similarity* or *nearest neighbor search* (NNS) is the task of retrieving database vectors most similar to a given query. NNS serves as a core building block across diverse applications such as retrieval-augmented generation [50, 65, 89], pattern recognition [11, 24, 62, 98, 112], classification [10, 13, 47, 83], clustering [14, 28, 84, 85], anomaly detection [15–17, 25, 71, 111], and recommendation systems [77, 88]. The rapid growth of large databases underscores the need for scalable, low-latency similarity search. NNS methods fall into two categories: exact or approximate [29]. *Exact* methods retrieve the true nearest neighbors but rarely meet real-time constraints, whereas *approximate* methods trade marginal accuracy for substantially higher efficiency. This accuracy–latency trade-off has driven widespread adoption of approximate solutions in real-time and interactive systems [66, 86, 90].

As approximate methods have gained prominence, researchers have proposed a wide spectrum of algorithms designed to address distinct challenges and application requirements. To bring structure to this diversity, these approaches can be broadly grouped into five categories: partition-based, indexing-based, hash-based, quantization-based, and graph-based methods. Within this taxonomy, *partition-based* algorithms [32, 45, 78] divide high-dimensional space into smaller subspaces. When a query arrives, the algorithm locates the corresponding subspace and examines its contents, as well as neighboring regions, to identify candidate neighbors. In contrast, *indexing-based* algorithms [19, 102, 105] adopt a hierarchical structure, organizing database vectors into trees and guiding search traversal through compressed query representations.

To mitigate excessive memory requirements, hash-based methods [48, 63, 64, 74, 97, 115] exploit the property that similar items map to proximate hash values, restricting candidate searches to vectors with matching codes. Despite theoretical guarantees, hash-based methods often require multiple hash tables to achieve competitive performance, increasing computational and storage costs. As the demand for large-scale similarity search continues to grow, quantization-based methods [5, 12, 40, 43, 58, 82] have emerged, substantially reducing memory requirements by encoding high-dimensional vectors, albeit at the expense of increased computational complexity relative to hash-based approaches. Complementing these advances, graph-based algorithms [36–38, 54, 68, 76, 95, 99] enhance search efficiency by connecting similar vectors within

a graph structure. Furthermore, hybrid approaches have emerged that combine the strengths of multiple categories. Notably, quantization has been integrated with graph-based [4] and indexing-based algorithms [67] to enhance efficiency and scalability.

In the past decade, several studies have examined Approximate Nearest Neighbor Search (ANNS) methods [7, 8, 30, 68, 101]. However, prior studies exhibit critical limitations by (i) omitting major algorithmic families; (ii) overlooking recent methodological advances; (iii) using limited datasets designed to reflect the requirements and properties of emerging AI applications; and (iv) lacking rigorous statistical analysis to validate performance differences. Specifically, current benchmarks often fail to encompass the full scope of the ANNS landscape. While some omit indexing-based approaches [7, 68, 101], others focus narrowly on specific categories, such as graph-based [8, 101] or indexing-based methods [29], neglecting significant progress in the field. In addition, most studies exclude evaluations of hardware-accelerated variants. This limited scope hinders a comprehensive understanding of the relative strengths and weaknesses of ANNS techniques.

Although prior benchmarks [7, 29, 30, 68] were conducted between 2019 and 2021, the landscape of ANNS has evolved considerably since then. Notably, approximately 33% of the methods evaluated in this study were introduced after 2021, underscoring the rapid and ongoing innovation within this field. As a result, such evaluations become less relevant for researchers and practitioners seeking to identify the most effective approaches. Consequently, existing benchmarks are constrained by the lack of evaluations on datasets capturing modern AI applications. Finally, the lack of statistical analysis sometimes creates misconceptions about the effectiveness of methods, as often noticeable differences in performance appear statistically negligible. In particular, no existing benchmark offers a statistically rigorous evaluation of observed performance differences. In summary, current ANNS benchmarks suffer from incompleteness, failure to incorporate key categories, and lack of recency. This underscores the urgent need for a comprehensive, modern, and statistically robust benchmark that encompasses all major ANNS categories and developments of the past two decades.

To address the limitations of prior benchmarks, we introduce **ATLAS**, a large-scale evaluation spanning all five major categories of ANNS and covering ***45 methods***, including hardware-accelerated and hybrid variants that combine quantization with indexing or graph-based methods. The evaluation spans ***58 diverse datasets***, including datasets reflecting neural network representations, as well as traditional datasets from video, image, text, audio, and other modalities. From Table 1, it is evident that this study more than doubles both the number of datasets and models evaluated compared to prior benchmarks, thereby establishing ATLAS as the most comprehensive ANNS evaluation to date. ATLAS goes beyond conducting an extensive study and advocates for a fair and comprehensive paradigm for evaluating ANNS methods. Prior studies typically reported performance at a single (speedup, recall) operating point [68], which obscures the full accuracy–efficiency trade-off and can misrepresent relative performance under varying latency or recall requirements. To address this, we introduce a new evaluation measure based on the speedup–recall curve, which integrates speedup across the entire recall range to provide an unbiased assessment of query efficiency. This approach removes the need to set arbitrary thresholds that often introduce bias or conceal important trends. In addition to proposing a novel evaluation measure for this era, this study applies a comprehensive statistical analysis to examine the significance of ranking differences across datasets, thereby providing a more holistic and robust understanding of performance. For transparency and reproducibility, we make all code and datasets publicly available [1]. Furthermore, ATLAS ensures rigor through a two-phase evaluation procedure. In *Phase I*, methods are assessed *within* their respective algorithmic families, and the top performers are identified. In *Phase II*, these family-level winners are compared globally to provide a holistic cross-family perspective.

Our analysis provides several key insights into the performance of ANNS methods. VAQ [82] is the best quantization-based method, offering substantially faster query times than alternative quantization techniques and achieving query efficiency comparable to leading graph-based algorithms. Across four methodological categories, previously unreported top-performing methods are identified, among which VAQ and DUMPY [105] show statistically significant improvements. Specifically, VAQ is the best quantization-based method, SCANN [45] is the best partition-based method, DB-LSH [96] is the best hash-based method, and DUMPY is the best indexing-based method. Hardware-accelerated methods, however, show strong processor-specific dependencies; for instance, SVS-LVQ [4] attains markedly higher throughput on Intel processors, whereas SVS [4] consistently outperforms it on AMD architectures. While both indexing and hardware acceleration deliver substantial throughput gains, these improvements come at the expense of reduced accuracy relative to their vanilla counterparts. Interestingly, a graph-based algorithm, namely, DISKANN that performed poorly in prior benchmarks [101] ranks among the best in our evaluation, likely owing to our use of the authors' original implementation rather than a re-implementation. Furthermore, our statistical analysis reveals that the performance differences among leading graph-based methods are not statistically significant. Finally, this study shows that no single method demonstrates superiority across all performance dimensions. Collectively, the results demonstrate the breadth and rigor of our benchmark, changing the landscape of ANNS by closing gaps in prior evaluations and establishing a comprehensive and reliable resource for the community.

We start with a discussion of the problem statement and related work (Section 2). Then, we present our contributions:

- We integrate all major ANNS paradigms, including underexplored approaches such as hashing and indexing, alongside state-of-the-art models to ensure relevance (Section 3.1.3).
- We evaluate 45 ANNS models across 58 datasets, establishing the most comprehensive benchmark to date and doubling the coverage of prior studies (Sections 3.1.2 and 3.1.3).
- We introduce a novel measure to quantify query efficiency derived from the speedup–recall curve (Section 3.2).
- We employ rigorous statistical analysis to assess the significance of relative rankings across datasets (Section 4).
- We included hardware-accelerated and composite methods, capturing emerging directions in ANNS (Sections 4.1 and 4.2).
- We summarize key findings and outline future research directions (Section 5).

Finally, we recapitulate our contributions and examine the extensive implications of our research (Section 6).

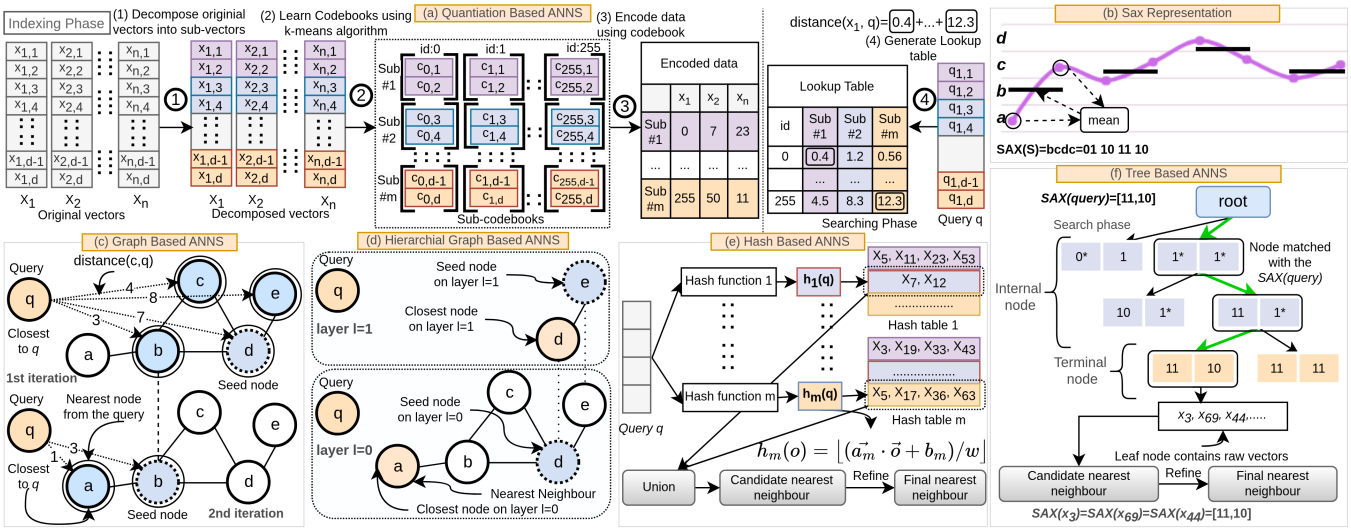


**Figure 1: (a) Illustrates a quantization-based algorithm's indexing and searching phase. (b) SAX representation of vector data. (c) An illustrative example of a graph-based ANNS algorithm where the closest node to the query among the seed and its unvisited neighbors is iteratively chosen as the new seed until convergence. (d) An example of a two-layered hierarchical graph-based algorithm where the nearest node of the 2nd layer acts as the seed node of the 1st layer. (e) Illustrates the search phase of a hashing-based algorithm. (f) Illustrates the search phase of tree-based algorithms.**

## 2 BACKGROUND AND RELATED WORKS

In this section, we first formally define ANNS in Section 2.1. We then briefly review ANNS algorithms across various categories, including hash-based (Section 2.2), quantization-based (Section 2.3), graph-based (Section 2.4), partition-based (Section 2.5), and indexing-based (Section 2.6) methods.

### 2.1 Problem Statement

Similarity search is a fundamental problem aiming to identify items most similar to a given query from a collection of $n$ items. Formally, given $n$ $d$-dimensional vectors $X_b$ and a $d$-dimensional query vector $q$, similarity search aims to determine the vector in $X_b$ nearest to $q$ based on the chosen distance measures. The nearest neighbor ($NN$) similarity search extends to $KNN$ similarity search, which seeks to retrieve the $k$ closest neighbor instead of one. The NN problem can be formally expressed as:

$$NN(q) = \arg\min_{x \in X_b} dist(q, x) \tag{1}$$

ANNS methods fall into four categories by error bounds [29]. The *ng-approximate* category provides no theoretical guarantees on the accuracy of neighbor distances . Similarly, $c-approximate$ KNN algorithms do not provide theoretical guarantees about their error bounds. Nevertheless, for a specified approximation ratio $c$ and query vector $q$ c-approximate ANNS algorithms guarantee that the neighbors returned by the algorithm will not exceed $c$ times the distance of actual nearest neighbors. In contrast, $\epsilon-approximate$ algorithms guarantee that the error bound will not exceed $(1+\epsilon)$. Furthermore, $\delta-\epsilon-approximate$ algorithms ensure that the error bound will not exceed $(1+\epsilon)$ with a probability of $\delta$. Regarding the robustness of theoretical error guarantees, the ranking of these algorithms is as follows: ng-approximate < c-approximate < $\delta-\epsilon$-approximate. Furthermore, time-constrained ANNS algorithms restrict the execution time of queries. Having introduced the preliminaries of ANNS, we now focus on reviewing ANNS methods across five categories, beginning with hashing-based approaches.

### 2.2 Hash-based Algorithms

A key technique in this category is Locality-Sensitive Hashing (LSH) [41, 53], which employs specialized hash functions to project high-dimensional data into compact, lower-dimensional representations while preserving pairwise similarity. Typically, these functions utilize random projection vectors to map input data to scalar values and assign them to specific hash buckets. This design increases the likelihood that similar data points fall into the same bucket, thereby enabling efficient and scalable nearest-neighbor searches. Figure 1(c) illustrates a simplified depiction of hash-based algorithms. The hash function is formulated as follows:

$$h_{\vec{a},b}(o) = \left\lfloor \frac{\vec{a} \cdot \vec{o} + b}{w} \right\rfloor \tag{2}$$

Despite being c-approximate ANNS method, LSH utilizes multiple hash functions to enlarge the candidate set, aggregates results across distinct hash tables, and subsequently identifies nearest neighbors through a linear scan of the combined candidates. C2LSH [39] addresses memory usage concerns of LSH by employing collision counting and virtual rehashing; however, virtual rehashing incurs significant computational costs. To overcome this limitation, QALSH [48] introduces a query-aware bucket partitioning technique that removes the necessity for pre-constructed hash tables.

QALSH indexes base vectors with a B+ tree and dynamically constructs a virtual hash table during queries, minimizing the overhead of hash table creation for each search.

While QALSH reduces the reliance on pre-constructed hash tables, LCCS-LSH [64] identifies nearby items by dynamically combining successive hash values utilizing a Circular Shift Array (CSA) inspired by suffix arrays. This approach is particularly useful for complex datasets but, like many LSH methods, suffers from the curse of dimensionality. PM-LSH [115] addresses this limitation by mapping data to a lower-dimensional space and organising the transformed data using a PM-tree [93]. While this indexing structure mitigates the effects of high dimensionality, QALSH can be memory-intensive for large datasets due to the overhead of storing data splits and metadata. Alternatively, DB-LSH [97] optimizes memory usage by combining bucketing and indexing strategies. In contrast, R2-LSH [74] employs multiple 2-dimensional hash tables and generates candidate lists through a query-centric ball mechanism, effectively reducing the likelihood of false negatives.

S2-LSH and SA-LSH [63] propose a schema for weighted ANNS, incorporating a novel spherical asymmetric transformation. These methods map the database and query vectors from $R^d$ to $R^{2d}$, providing a more efficient way to handle weighted NNS. However, the added asymmetric transformation introduces computational overhead. The LSH-based methods have lower query throughput due to hash boundary issues, whereas graph-based methods achieve higher query performance by greedy search in an approximate nearest neighbor graph. However, the construction time of graph-based methods is much higher than that of hash-based indexes. LSH-APG [114] combines APG with LSH, which introduces a method for selecting suboptimal entry points and a pruning strategy based on LSH, both of which accelerate indexing and query times.

## 2.3 Quantization-based Algorithms

Unlike hash-based approaches, quantization-based ANNS methods are data-dependent and optimize memory usage by encoding high-dimensional vectors into lower-dimensional codes from compact codebooks. Storing codes instead of raw vectors yields substantial memory savings at the expense of approximation errors, resulting in a trade-off between memory efficiency and search precision. Product Quantization (PQ) [58] enhances this strategy by partitioning each vector into multiple subspaces, learning smaller sub-codebooks through clustering, and representing the final codebook as the Cartesian product of these sub-codebooks.

In practice, datasets often exhibit varying importance across subspaces and treating them equally can lead to reduced performance. The Optimized Product Quantization (OPQ) [40] addresses this by rotating dimensions to achieve a more balanced variance distribution across subspaces. To find the optimal rotation, OPQ optimizes space decomposition and codebooks to minimize quantization error. However, achieving a uniform distribution is challenging when a few dimensions dominate the variance of the data. Variance-Aware Quantization (VAQ) [82] addresses this challenge by adaptively allocating bits to each subspace in proportion to its variance, thereby better capturing the data's intrinsic structure. VAQ also introduces a dimension-swapping strategy and adaptive vector sizes to reduce variance disparity among subvectors. Additionally, VAQ proposes a hardware-agnostic pruning method that significantly accelerates queries compared to other quantization algorithms.

Moreover, researchers have devised hardware-accelerated approaches to quantization-based ANNS methods to boost efficiency. PQ uses a cache resident lookup table to compute distances with neighbors. PQFS [5] compresses these lookup tables to store them in the SIMD register that allows utilizing SIMD instruction for faster computation. While the Bolt [12] shares fundamental similarities with PQFS, Bolt incorporates two optimization techniques to accelerate query throughput. Firstly, Bolt utilizes significantly smaller codebooks, specifically employing 16 centroids for each subspace, which reduces the computation times of the k-means algorithm. Secondly, Bold approximates the distance matrix, and this computation is faster than other quantization-based algorithms. These two optimizations enable Bolt to gain significant speedup.

## 2.4 Graph-based Algorithms

Graph-based methods constitute a distinct class of ANNS techniques, differing fundamentally from hash-based and quantization-based approaches. These methods model dataset vectors as nodes within a graph structure, where search is conducted by traversing the graph to identify nearest neighbors (Figure 1(d)). The procedure typically begins at a seed node and iteratively progresses toward the closest adjacent node, continuing until convergence is achieved. Termination occurs once the traversal stabilizes, with no further improvement in the seed node. To minimize the number of hops, hierarchical graph-based methods (Figure 1(c)) construct multi-layer graph structures for accelarating convergence. Each vector is assigned a random integer value $r \in [1, m-1]$, where $m$ denotes the total number of layers. A vector is included in layer $i$ if $r \leq i$. Consequently, higher layers contain fewer nodes, thereby providing a coarse-to-fine search strategy. The nearest neighbor identified in layer $i$ is subsequently used as the entry point for the search within layer $i-1$, enabling efficient navigation toward the target.

The foundation of graph-based ANNS algorithms lies in several fundamental graph structures, which we introduce prior to discussing specific techniques [101]. The $k$-nearest neighbor graph (KNNG) is a directed graph where each node is connected to its $k$ nearest neighbors. However, KNNG only considers the distances of neighbors to construct the index without considering the distribution or coverage of these neighbors, which can lead to the formation of disjoint subgraphs. Conversely, the relative neighborhood graph (RNG) [55] is a sparse undirected graph over a set of points, where an edge between $p$ and $q$ exists only if no third point $r$ is simultaneously closer to both than they are to each other. A spanning tree is a subgraph of a connected, undirected graph that includes all the original graph's vertices. A graph can have multiple spanning trees, with the minimum spanning tree (MST) [46] having the least weight. While MST ensures global connectivity, its construction may occasionally lead to longer traversal paths. The Delaunay Graph (DG) [34] is constructed via Delaunay triangulation, connecting points that form a triangle with no other points inside, thereby ensuring local connectivity while avoiding unfavorable geometric configurations. Having introduced the fundamental graph types, we now briefly describe graph-based ANNS methods categorized by their underlying structures.

*2.4.1 KNNG based Algorithms:* EFANNA [36] integrates hierarchical structure-based techniques with graph-based methods to achieve both efficiency and accuracy. The approach employs a KNNG for rapid exploration, while hierarchical structures provide effective initialization, thereby enhancing search efficiency. Traditional graph-based methods often suffer from convergence to local optima and high construction costs; EFANNA addresses these limitations by offering robust initialization for the expansion process.

*2.4.2 DG and RNG based algorithms:* Some DG-based algorithms utilize RNG to diversify the distribution of edges for each node in the graph, consequently reducing the high degree. NSW [61] constructs its graph incrementally, adding vectors one at a time. Each node connects to a small set of its nearest neighbors already in the graph, forming a local structure that ensures global connectivity. Longer edges, created in the initial phases, enhance search accuracy, while shorter edges, added later, improve search efficiency. Inspired by NSW, HNSW [76] proposed a hierarchical graph in which the hierarchical structure is derived from the probabilistic data structure called a skip list. In the HNSW graph, each layer consists of a standalone NSW graph. As we progress from the top to the bottom layers, the graph becomes denser. FINGER-HNSW [21] is built upon HNSW, which introduces an approximated distance calculation by modelling the angles between neighboring vectors, thereby reducing the computational complexity of distance calculations. More recently, FLATNAV [79] demonstrate that the hierarchical layers in HNSW provide no benefit in high-dimensional settings, showing that a flattened variant achieves comparable performance while relying on hub nodes to guide efficient search. NGT [54] introduces three-degree adjustment methods for optimizing indegrees and outdegrees, improving upon previous graph construction techniques and employing a VP-tree [113] to identify an optimal seed rather than selecting the seed node randomly.

*2.4.3 KNNG and RNG-based algorithms:* KNNG-based graphs focus solely on nearest neighbors, leading to redundancy when nodes closer to each other share similar neighbors. Incorporating RNG within the KNN introduces diversity among neighbors by ensuring that nodes are connected only if there is no closer node between them. DPG is a diversification of an existing KNN graph followed by the introduction of reverse edges. In the construction phase, NSG uses an edge selection strategy to prune the graph in the construction phases inspired by the Monotonic Relative Neighborhood Graph [26] to reduce the index size. In the SSG [37], each node's outgoing edges are evenly distributed across multiple directions, simulating a satellite network that transmits signals to rapidly narrow search parameters. DISKANN [42, 92, 95] enables efficient searching of billion-scale datasets on a single CPU, previously achievable only with GPUs. DISKANN stores indexes on SSDs, reducing reliance on expensive RAM, and divides datasets into smaller chunks for optimal searching. Additionally, DISKANN introduces an optimized data retrieval technique, as the dataset is stored on disk. Glass [104] incorporates ANNS methods such as HNSW and NSG, while also supporting scalar and product quantization variants (e.g., SQ [3], PQ) to reduce memory consumption.

*2.4.4 MST based Algorithms:* HCNNG [99] first constructs a Minimum spanning tree (MST) by taking the data points in the dataset and then creating hierarchical clusters from the MST. This model uses hierarchical clusters to guide the search process, prioritizing the exploration of points likely closer to the query. Leveraging hierarchical graph representation enables the model to quickly narrow down the most promising region of the dataset during the search phase, improving efficiency.

SVS [4], a performance library for Vamana, reduces memory usage through locally adaptive vector quantization while mitigating random access overhead with advanced prefetching. Building on the idea of tightly coupling quantization with graph traversal, SymphonyQG [44] advances this direction by (i) employing RaBitQ for quantized distance estimation; (ii) replacing costly post-processing with implicit re-ranking; and (iii) refining graph construction so node degrees align with SIMD-batched computation, thereby ensuring seamless interaction between quantization and search. Extending the focus from algorithmic integration to system-level optimization, VSAG introduces an optimized framework that overcomes key bottlenecks in graph-based search through a cache-aware layout with software prefetching, automatic parameter tuning that avoids index rebuilding, and adaptive distance computation combining low- and high-precision operations. Collectively, these innovations enable more efficient and accurate traversal.

## 2.5 Partition-based Algorithms

Partition-based approaches divide the high-dimensional vector space into disjoint regions to accelerate similarity search. Given a query vector $q$, the algorithm identifies the region $r_q$ containing $q$, with the nearest neighbors typically residing within $r_q$ or its adjacent regions. To enable efficient traversal, these methods frequently employ recursive tree-based data structures. A classic example is the KD-tree, a binary tree in which each leaf node corresponds to a $k$-dimensional vector, while internal nodes partition the space into two subregions. The partitioning hyperplane is orthogonal to a chosen dimension, with the splitting axis typically determined by cycling through the $k$ dimensions in sequence.

Several systems have extended or refined these principles. Annoy [32] constructs a forest of trees, where each tree recursively partitions the space by selecting two random points and splitting along their bisecting hyperplane. This continues until each node contains only a small number of points. FLANN [78] offers a versatile library for large-scale nearest neighbor search, automatically tuning both the choice of algorithm and its parameters to balance accuracy and efficiency. FLANN supports KD-trees, hierarchical $k$-means trees, and autotuned hybrids.

Other specialized structures leverage metric properties. The VP-tree [113] organizes data in metric spaces by recursively selecting vantage points and partitioning remaining vectors into subsets based on their distances from the chosen reference. This recursive decomposition yields a balanced hierarchy that prunes large portions of the search space, substantially improving query efficiency. Similarly, the Multiple Random Projection Tree (MRPT) [52] accelerates search by combining sparse random projection trees with a voting mechanism that reduces both candidate set size.

Recent methods integrate partitioning with quantization or graph-based refinements. SCANN [45] enhances tree-based partitioning with product quantization, restricting the search to a small subset of clusters and compressing the dataset to reduce memory.

By employing asymmetric distance computations, SCANN minimizes quantization error, achieving superior performance over reconstruction-based techniques. SPTAG (Space Partition Tree and Graph) [110], in contrast, unifies tree partitioning (via KD-trees and balanced $k$-means trees) with graph refinement based on relative neighborhood graphs [22, 23, 100]. This hybrid design improves connectivity, search quality, and scalability, while also supporting dynamic updates and distributed deployments for billion-scale vector search.

## 2.6 Indexing-based algorithms

Figure 1(d) depicts the query phase of an indexing-based ANNS algorithm. In this paradigm, database vectors are hierarchically organized into a tree structure by applying summarization techniques that compress the vector representations. During query processing, the algorithm initiates traversal from the root node, where the compressed representation of the query is compared against the child nodes. At each step, traversal advances toward the branch that most closely aligns with the query representation. This process continues recursively until a leaf node is reached, which contains the candidate set of nearest neighbors for subsequent refinement.

The ISAX framework [91] builds a tree-based index using the SAX symbolic representation [69], optimizing ANN search time despite SAX's computational cost. iSAX 2.0 [18] and iSAX2+ [19] further improve upon the original framework. In this study, we included iSAX2+ due to its improved performance and scalability. The SAX representation converts vectors into a sequence of symbols, facilitating efficient indexing while reducing memory usage and accelerating processing. The process begins with normalizing the vector, followed by applying Piecewise Aggregate Approximation (PAA) [59], which segments the vector into sub-vectors and computes the mean for each. Afterward, SAX discretizes these mean values into symbols, completing the symbolic conversion.

The DS-tree [102] addresses the challenge of uneven variance distribution across dimensions by employing a dynamic data structure that adapts to varying levels of variability. Although Adaptive Piecewise Constant Approximation (APCA) [20] is similar to PAA, APCA uses dynamic-sized segments based on the variance of time-series data. The Extended Adaptive Piecewise Constant Approximation (EAPCA) [103] further refines APCA by using both the mean and standard deviation to characterize each segment, allowing for more precise representation. Building upon this foundation, the DS-tree incorporates EAPCA for dynamic segmentation of data. In a different approach, Dumpy [105] introduces a tree-like framework with nodes that can have multiple children, improving adaptability to varied data distributions. This structure is particularly robust against skewed data, where some values dominate. The Dumpy-fuzzy variant offers faster construction times.

Next, we focus on indexing-based techniques such as Inverted File (IVF) [56] and Inverted Multi-Index (IMI) [9], which, when combined with quantization methods, significantly enhance search efficiency in ANNS. IVF partitions the database into Voronoi cells, grouping vectors with similar proximities. This partitioning confines the search to the nearest regions, effectively improving efficiency. While IMI and IVF rely on partitioning vectors, IMI goes further by dividing the Voronoi cells across multiple dimensions, creating a hierarchical structure. This hierarchical partitioning refines the search process by narrowing the search space even more.

On the complementary front, VA+FILE [31] employs the Karhunen Loève Transform [75] to project a time series of length $n$ into a sequence of real-valued coefficients, which are subsequently quantized into discrete symbols using a scale quantizer. The method utilizes an approximation-based distance measure to prune the majority of candidates via bounding before computing exact distances on the remaining subset. Although inherently scan-based, VA+FILE is classified under the indexing category owing to its pruning strategy analogous to ISAX and DS-TREE, and it constitutes the only scan-based method included in our evaluation

**Table 2: Detailed summary of attributes for the 58 datasets included in this study.**

| Dataset | Dim | # Base | # Query | Dataset | Dim | # Base | # Query |
|---|---|---|---|---|---|---|---|
| ASTRO [94] | 256 | 1,000,000 | 100 | SIFT [73] | 128 | 1,000,000 | 10,000 |
| AUDIO [68] | 192 | 53,387 | 200 | SPACE1B [30] | 100 | 1,000,000 | 100 |
| BIGANN [30] | 128 | 1,000,000 | 100 | TEXT-TO-IMAGE [30] | 200 | 1,000,000 | 100 |
| CIFAR [68] | 512 | 50,000 | 200 | TINY5M [49] | 384 | 5,000,000 | 1,000 |
| CRAWL [68] | 300 | 50,000 | 200 | TURING-ANNS [30] | 100 | 1,000,000 | 100 |
| DEEP1M [68] | 256 | 1,000,000 | 200 | SUN [68] | 512 | 79,106 | 200 |
| ENRON [68] | 1369 | 94,987 | 200 | TREVI [68] | 4096 | 99,900 | 200 |
| GIST [81] | 256 | 1,000,000 | 100 | UKBENCH [68] | 128 | 1,097,907 | 200 |
| GLOVE [68] | 100 | 1,192,514 | 200 | UQV [68] | 256 | 1,000,000 | 10,000 |
| IMAGENET [68] | 150 | 2,340,373 | 200 | WORD2VEC [74] | 300 | 1,000,000 | 1,000 |
| LASTFM [7] | 65 | 292,385 | 100 | ETHZ [109] | 256 | 36,643 | 100 |
| MILLIONSONG [68] | 420 | 992,272 | 200 | VCSEIS [109] | 256 | 160,178 | 100 |
| MNIST [68] | 784 | 69,000 | 200 | TSED [109] | 256 | 519,589 | 100 |
| NETFLIX [74] | 300 | 17,770 | 1000 | LENDB [109] | 256 | 1,000,000 | 100 |
| NOTRE [68] | 128 | 332,668 | 200 | STEAD [109] | 256 | 1,000,000 | 100 |
| NUSWIDE [68] | 500 | 268,643 | 200 | GOEFON [109] | 128 | 275,174 | 100 |
| NYTIMES [7] | 256 | 290,000 | 100 | INSTANCE [109] | 128 | 1,000,000 | 100 |
| RANDOM [68] | 512 | 50,000 | 200 | YELP [49] | 50 | 77,079 | 100 |
| SEISMIC [33] | 256 | 1,000,000 | 100 | MUSIC [49] | 100 | 1,000,000 | 100 |
| SALD [106] | 128 | 1,000,000 | 100 | OBS [109] | 256 | 1,000,000 | 300 |
| PNW [109] | 256 | 1,000,000 | 300 | OBST2024 [109] | 256 | 1,000,000 | 300 |
| NEIC [109] | 256 | 1,000,000 | 300 | MEIER2019JGR [109] | 256 | 1,000,000 | 300 |
| ISC-EHB [109] | 256 | 1,000,000 | 300 | IQUIQUE [109] | 256 | 1,000,000 | 300 |
| AGNEWS [27] | 1024 | 769,382 | 100 | ARXIV [6] | 768 | 1,000,000 | 100 |
| GOOGLE-QA [60] | 768 | 1,000,000 | 100 | LANDMARK [107] | 768 | 760,757 | 100 |
| YAHOO | 384 | 677,305 | 100 | CELEBA [72] | 2048 | 201,599 | 100 |
| CCNEWS | 768 | 495,328 | 100 | CODESEARCHNET [51] | 768 | 1,000,000 | 100 |
| MSCOCO [70] | 768 | 282,360 | 100 | LAION | 512 | 1,000,000 | 100 |
| YANDEX | 200 | 1,000,000 | 100 | LLAMA [87] | 128 | 256,921 | 100 |

# 3 ATLAS OVERVIEW

In this section, we first present the experimental setup in Section 3.1. We then describe the evaluation measures in Section 3.2. Finally, we explain the evaluation procedures in Section 3.3.

## 3.1 Experimental Setup

We now present the experimental environment and implementation, followed by the datasets together with their sources and attributes.

*3.1.1 Experimental Environment.* We conducted our experiments on a Ubuntu 22.04.3 LTS (64-bit) server with 1TB of memory and two AMD EPYC 7713 64-core processor.

*3.1.2 Datasets.* This study evaluates 58 datasets from diverse domains—including datasets reflecting neural network representations, as well as datasets from video, image, text, audio, astronomy, and seismology—to establish a comprehensive benchmark for ANNS algorithms. Table 2 summarizes their dimensionality, training set size, query set size, and sources. In this study, we adopt Euclidean distance as the similarity metric. Our work extends datasets from prior benchmarks [7, 30, 68] by incorporating a broader and more diverse collection of datasets [57].

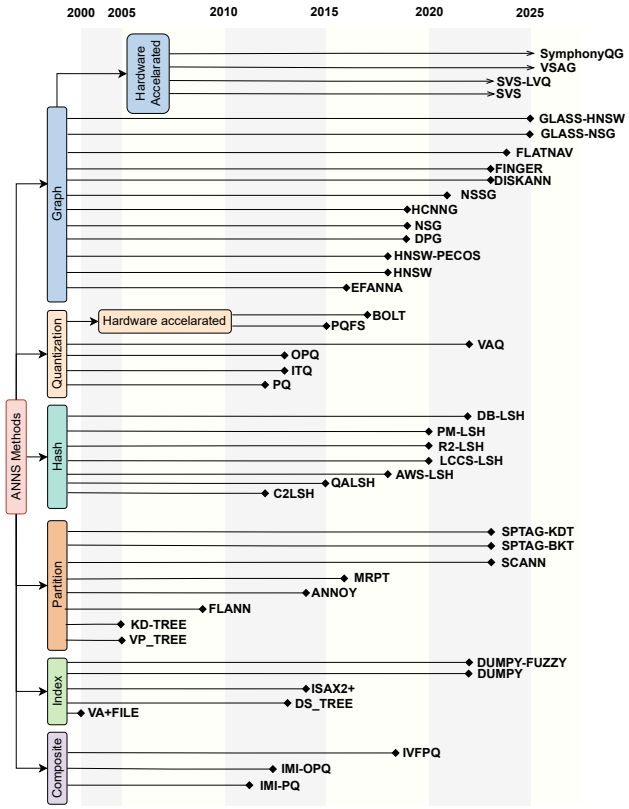


**Figure 2: Taxonomy of evaluated ANNS Methods.**

*3.1.3 Compared algorithms.* Figure 2 illustrates the 45 ANNS methods evaluated in this study, organized into six principal categories. The benchmark further includes hardware-accelerated methods. We compare these approaches with respect to search efficiency, scalability, and robustness across diverse datasets. Method selection was informed by prior studies [29, 30, 68, 101] and recent advances, with emphasis on algorithms that either introduce novel methodological contributions or significantly enhance established baselines. Approximately 33% of the evaluated models were introduced after 2021, underscoring the rapid pace of innovation in this domain. To ensure transparency and reproducibility, all implementations were obtained from publicly available repositories, implemented in C++, and evaluated utilizing single CPU core [1].

## 3.2 Evaluation Measures

This benchmark evaluates ANNS algorithms primarily using recall, which quantifies the proportion of correctly retrieved neighbors among the top-k results. To assess query performance, we employ speedup instead of raw query latency, thereby enabling a standardized comparison by measuring efficiency relative to a linear scan baseline. While recall is widely adopted, recall does not account for the ordering of retrieved instances. Consequently, in scenarios where the ranking of results is critical, relying exclusively on recall may provide an incomplete assessment of algorithmic effectiveness. To address this limitation, several prior studies have employed Mean

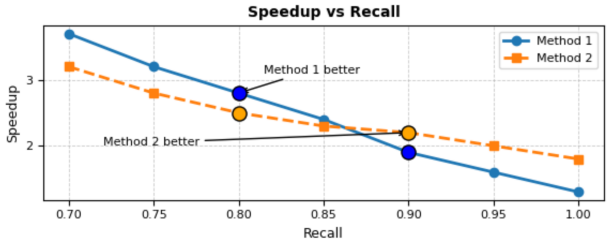


**Figure 3: Toy example of speedup–recall trade-offs between two models on a dataset.**

Average Precision (MAP) in conjunction with recall [29, 30, 82], since MAP explicitly incorporates the ranking quality of the top-k neighbors. Nevertheless, as prior studies have demonstrated that MAP and Recall are highly correlated and thus do not alter the relative ranking of models [68, 82]. Due to space limitations, we report results only using Recall in this study. The evaluation measures are formally defined as follows:

$$Recall@k = \left(\sum_{i=1}^{n_1} \frac{\#\text{ correct neighbors returned by algorithm}}{k}\right)/n_q$$

$$speedup(algorithm) = \frac{\text{query time using linear scan}}{\text{query time using the algorithms}}$$

Prior benchmarks often fail to capture the full query performance characteristics of ANNS methods, typically evaluating at a single operating point—for example, speed at fixed recall or recall at fixed speedup. While convenient, this approach obscures the broader performance landscape and allows the choice of operating point to distort method rankings, producing inconsistent conclusions across datasets. As shown in Figure 3, Method 1 surpasses Method 2 at a recall level of 0.8, whereas Method 2 outperforms Method 1 at a recall level of 0.9, highlighting the necessity for a more comprehensive metric to evaluate query performance. To overcome these limitations, we introduce the Weighted Cumulative Speedup over Recall (WCSR) measure. Unlike point-based evaluations, WCSR aggregates performance across a specified recall range, computing a speedup-weighted average that captures the trade-off between accuracy and efficiency more comprehensively. In doing so, WCSR provides a fairer basis for comparing query efficiency of ANNS methods across diverse operating regimes. This measure quantitatively captures query efficiency by computing the weighted area under the speedup–recall curve across the interval $[\,a\,, b\,]$, formally defined as follows:

$$\mathrm{WCSR}_{a,b} = \frac{1}{2} \cdot \sum_{k=1}^{N} \Delta^k_{\text{Speedup}} \cdot \Delta^k_{\text{Recall}} \cdot \frac{R_k + R_{k-1}}{2}$$

$$R = [R_0, R_1, \ldots, R_N] \quad \text{where } a = R_0 < R_1 < \cdots < R_N = b$$

$$\text{with: } \begin{cases} \Delta^k_{\text{Recall}} = R_k - R_{k-1}, \\ \Delta^k_{\text{Speedup}} = \text{Speedup}(R_{k-1}) + \text{Speedup}(R_k), \end{cases}$$

### 3.3 Evaluation Procedure

Having discussed the evaluation measures, we now turn our focus to the evaluation procedure. This study assesses ANNS models across four primary dimensions: recall, indexing time, memory usage, and speedup. To ensure a systematic and unbiased comparison, all model parameters were carefully tuned. Initial configurations were derived from insights reported in relevant literature and official public repositories, followed by structured parameter exploration to achieve optimal performance. When source publications did not specify parameter settings, we relied on comparative analyses from prior studies [7, 101] to guide selection. Our evaluation adopts a two-phase design. In the first phase, algorithms are analyzed within their respective categories to identify their inherent strengths and limitations. In the second phase, we extend the analysis to cross-category comparisons to synthesize and contrast the most effective methods. This tiered framework ensures a comprehensive understanding of algorithmic trade-offs across diverse ANNS paradigms.

We adopt a rigorous statistical framework to assess whether observed performance differences among algorithms are statistically significant and practically meaningful. Query efficiency is first quantified using the *WCSR* measure, after which we employ non-parametric tests tailored to the nature of the comparison. The Friedman test [35] is applied when comparing more than two methods across multiple datasets, while the Wilcoxon signed-rank test [108] is used to conduct pairwise comparisons on multiple datasets. When a significant Friedman result is observed, the Nemenyi post hoc procedure [80] further discriminates among methods, enabling the derivation of robust algorithmic rankings. This combined use of complementary statistical tests yields a comprehensive and reliable assessment of performance differences. It is important to note that certain models exhibit poor performance on specific datasets. Including these methods when determining the intersection of the maximum recall range would lead to unfair comparisons. Therefore, such underperforming methods were excluded when computing the maximum recall range. Beyond recall, we evaluate index construction time and memory usage. Rather than hand-calculating memory, we log process-level consumption during execution acknowledging implementation-dependent variability.

## 4 EVALUATIONS AND ANALYSIS

In this section, we first evaluate the performance of quantization-based ANNS methods (Section 4.1). We then illustrate the experimental results of ANNS algorithms across various methodologies, including graph-based (Section 4.2), hash-based (Section 4.3), indexing-based (Section 4.4), and partition-based (Section 4.5). Finally, we consolidate these findings by comparing the top performing ANNS methods from all categories in Section 4.6. This analysis is designed to address the following four research questions (**RQ**):

- **(RQ1)** Which methods achieve the highest performance within each algorithmic category?
- **(RQ2)** What are the strengths and limitations inherent to different methodological approaches?
- **(RQ3)** Does an ANNS solution exist that simultaneously balances accuracy, runtime efficiency, and memory consumption?
- **(RQ4)** How do dataset characteristics, such as high dimensionality, influence model performance?

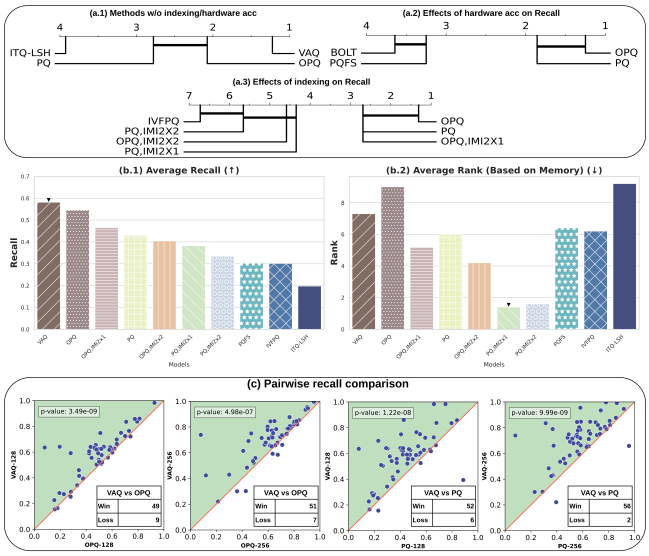


**Figure 4: (a) Critical diagram of quantization-based methods using WCSR in different hierarchies. (b) Average recall and average memory usage rank. (c) Pairwise recall comparison of VAQ with PQ and OPQ.**

### 4.1 Comparison of Quantization based methods

This section presents a comprehensive evaluation of three categories of quantization approaches: standalone quantization techniques, hybrid methods integrating quantization with indexing, and hardware-accelerated quantization variants. All methods are assessed under both 128-bit and 256-bit budget configurations, with each subspace allocated 8 bits, following the standard evaluation paradigm for quantization-based algorithms [82]. Exceptions to this setup include VAQ and the hardware-accelerated methods, for which different bit allocations are required. Unlike the general evaluation procedure described in Section 3.3, this analysis adopts a configuration tailored specifically to the practical constraints of quantization-based approaches. For hardware-accelerated methods such as Bolt and PQFS, optimization restrictions limit their evaluation to a maximum of 4 bits per subspace, a constraint we explicitly adopt in our experiments.

To ensure clarity in performance comparison, rankings are summarized hierarchically using a critical diagram (Figure 4). Due to space constraints, only results from the 256-bit configuration are displayed, as the trends observed at 128 bits are highly consistent. Across both configurations, VAQ demonstrates a clear and statistically significant advantage over competing quantization methods. Notably, the absence of a connecting line between VAQ and OPQ in the critical diagram highlights the distinct superiority of VAQ. On average, VAQ achieves approximately a 4% higher recall than OPQ, establishing VAQ as the leading quantization-based method.

Figure 4 (c) compares VAQ with OPQ and PQ in terms of recall, demonstrating that VAQ achieves superior performance across most datasets in both configurations. Moreover, OPQ incurs substantially higher construction costs than both VAQ and PQ, while VAQ is also more memory-efficient. Considering recall, construction time, and memory usage, VAQ emerges as the best-performing quantization-based method **(RQ1)**. Notably, OPQ's construction cost increases

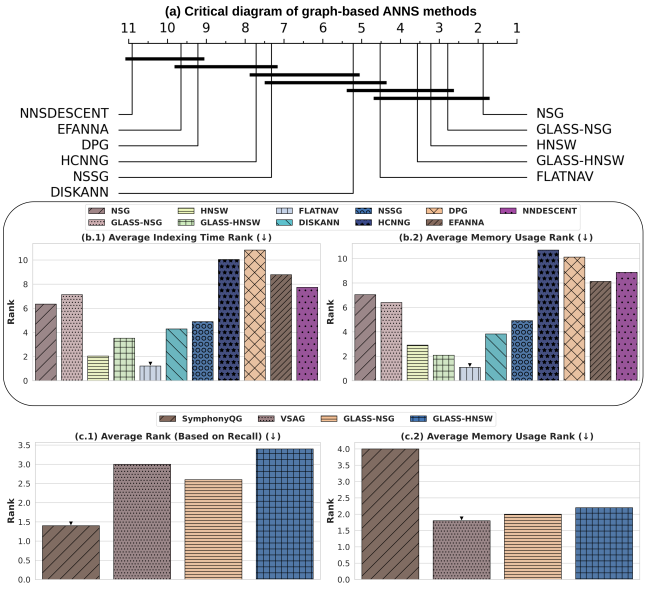


**Figure 5: (a) Average ranking of graph-based algorithms using WCSR; (b) Average indexing time rank, and average memory usage rank of graph-based methods (c) Comparison of hardware accelerated graph based methods with vanilla graph based methods in terms of recall and memory usage.**

significantly with high-dimensional datasets such as Enron and Trevi, due to the iterative computation of rotation vectors, a trend not observed in other quantization approaches **(RQ4)**.

Incorporating indexing into quantization methods reduces recall and increases construction time, while substantially improving query efficiency. As shown in Figure 4, the construction times of IVFPQ, OPQ, and OPQ-IMI2×1 are markedly higher than those of other quantization-based methods. Although OPQ achieves higher recall than PQ, this comes at the expense of considerably longer construction times. Accelerated quantization approaches, in contrast, trade reduced recall for faster query performance. A critical limitation of hardware-accelerated quantization lies in their restriction to 4 bits per subspace, which results in a pronounced recall decline compared to PQ with 8-bit allocations. However, when PQ is similarly restricted to 4 bits per subspace, its recall becomes comparable to that of the hardware-accelerated counterparts.

## 4.2 Comparison of Graph-based methods

In this section, we evaluate graph-based methods as well as graph based method with hardware acceleration. The critical diagram in Figure 5(a) shows that NSG, NSG-GLASS, HNSW, HNSW-GLASS, and FLATNAV consistently constitute the top-performing group of graph-based ANNS methods, with no statistically significant differences among them. Although NSG achieves the highest average rank, the overlap in performance intervals prevents establishing clear superiority over the other leading methods. In contrast, approaches such as NNSDESCENT, EFANNA, and DPG consistently fall into the lowest-performing group across datasets. Recall alone, however, is insufficient to determine algorithmic superiority; construction time and memory usage must also be considered. As

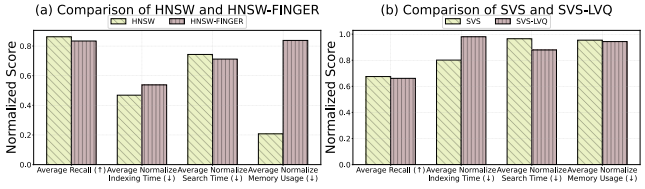


**Figure 6: Comparison of (a) HNSW and HNSW-FINGER and (b) SVS and SVS-LVQ on different measures.**

shown in Figure 5(b), FLATNAV, HNSW, and HNSW-GLASS exhibit the lowest construction times among graph-based methods, whereas NSG and HNSW-GLASS incur comparatively higher costs. Memory usage reveals similar trends as construction time. Balancing recall, construction time, and memory efficiency, NSG, NSG-GLASS, and HNSW emerge as the most effective methods **(RQ1)**. Accordingly, we select these three methods as representatives for the subsequent stage of evaluation.

In this study, we analyze the performance of hardware accelerated graph-based models, specifically SymphonyQG and VSAG, in comparison to their vanilla graph-based counterparts. As illustrated in the Figure 5(c), SymphonyQG achieves the highest recall among all evaluated methods, demonstrating the effectiveness of hardware acceleration in improving search accuracy. However, VSAG does not surpass the performance of GLASS-NSG in terms of recall. From the perspective of memory efficiency, SymphonyQG exhibits a clear disadvantage, performing worse than vanilla graph-based methods, whereas VSAG demonstrates superior memory efficiency. Taken together, these findings highlight the inherent trade-offs in hardware-accelerated graph-based methods: while they can substantially enhance recall or memory efficiency depending on the design, they do not universally outperform traditional graph-based methods across all evaluation measures.

This study extends analysis beyond traditional graph-based approaches by comparing HNSW with HNSW-FINGER and SVS with SVS-LVQ. For this evaluation, we utilize the HNSW implementation from the Pecos library [2], ensuring that HNSW-FINGER, which builds upon HNSW-Pecos, maintains the same level of optimization. Furthermore, while SVS-LVQ is specifically designed for Intel CPUs, HNSW-FINGER relies on AVX-512 instructions, which are incompatible with our AMD machine. Therefore, all experiments presented in Figure 6 were conducted on a Dell Intel CPU Max 9470 HBM machine to ensure compatibility and fairness in evaluation.

HNSW-FINGER, an enhanced variant of HNSW, aims to improve computational efficiency through distance approximation. However, as Figure 6(a) shows, this optimization has drawbacks. Despite its intended advantages, HNSW-FINGER consistently lags behind HNSW across key measures—recall, search time, and construction time—yielding only minor improvements in search speed. We evaluated both algorithms under identical parameter settings to ensure a fair comparison, reinforcing the robustness of our findings. A similar pattern emerges with SVS and SVS-LVQ. As illustrated in Figure 6(b), SVS-LVQ improved search efficiency over SVS but failed to deliver significant gains. While recall and memory usage remains similar for both, SVS-LVQ requires longer construction times without significantly reducing search time. However, when we compared SVS and SVS-LVQ on the AMD EPYC processor, SVS

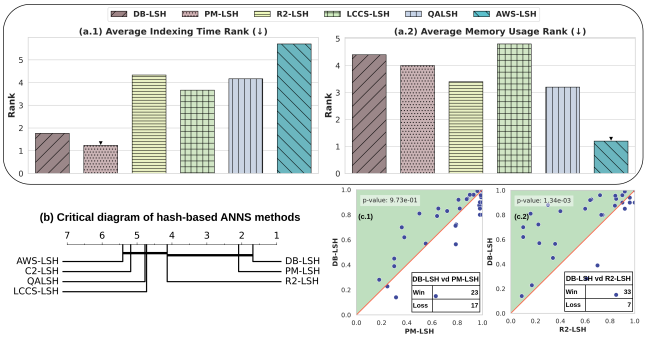


**Figure 7: (a) Average indexing time ranking, and average memory usage ranking of hash-based methods (b) Ranking of hash-based algorithms using WCSR (b) Pairwise recall comparison of DB-LSH with PM-LSH and R2-LSH**

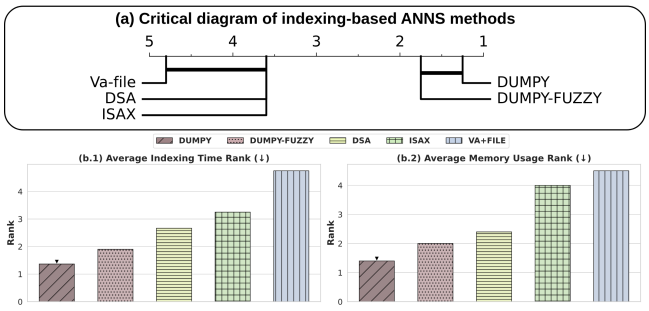


**Figure 8: (a) Ranking of indexing-based algorithms using WCSR (b) Average indexing time ranking and average memory usage ranking of indexing-based methods.**

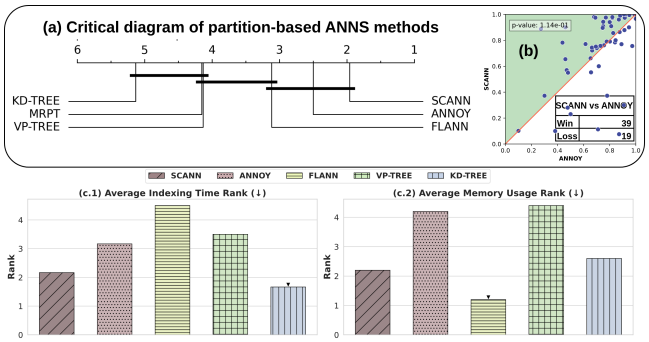


**Figure 9: (a) Ranking of partition-based algorithms using WCSR (b) Pairwise recall comparison of SCANN and ANNOY (c) Average indexing time ranking and average memory usage ranking of partition-based methods.**

outperformed SVS-LVQ by nearly fourfold in query throughput. Despite this platform difference, trends in construction time, memory consumption, and recall performance remained consistent across both hardware environments.

## 4.3 Comparison of LSH based methods

This research incorporates seven hash-based methods for a comprehensive evaluation, although AWS-LSH retrieves the nearest neighbors in a weighted space. To incorporate AWS-LSH in this benchmark, we have assigned equal weight to each dimension. It is crucial to emphasize that the datasets utilized in the referenced papers are not always identical to those employed in this benchmark, even when the data sources are the same. For example, the dimensionality of the CIFAR dataset utilized in this benchmark differs from that in the DB-LSH papers. Moreover, the number of nearest neighbors ($k$) employed in this benchmark is inconsistent with the referenced paper. Consequently, some comparisons may not fully align with the results presented in the original publications.

We computed recall, construction time, and memory usage for each dataset and model using the same methodology employed in the analysis of graph-based methods. Using the recall values obtained across all datasets at a specific speedup. Figure 7 visualizes the average ranking of hash-based ANNS methods based on speedup through a critical diagram. From this diagram, we can infer that DB-LSH outperforms all other methods, followed by PM-LSH, R2-LSH, QALSH, and AWS-LSH. The critical diagram also reveals that a line passes through DB-LSH and PM-LSH, indicating that both perform similarly in statistical terms, though DB-LSH shows a slight advantage. Moreover, Figure (c.2) highlights that DB-LSH achieves a higher recall across more datasets than PM-LSH.

Figure 7 (a.1) reports the average recall, while Figures 7 (a.2) and 7 (a.3) present the average rankings of LSH-based methods with respect to construction time and memory usage, respectively. The results show that DB-LSH delivers the highest speedup, whereas PM-LSH demonstrates superior performance in both construction time and memory efficiency. Although DB-LSH attains a marginally higher average recall than PM-LSH, statistical analysis indicates that this difference is not significant. Taken together, these findings suggest that, when considering search time, construction time, and memory usage, DB-LSH and PM-LSH stand out as the most effective hash-based ANNS methods **(RQ1)**.

## 4.4 Comparison of Indexing based methods

To establish a performance benchmark for indexing-based methods, this benchmark incorporated isax2+, DSA-tree, Dumpy, and Dumpy-Fuzzy, applying the same methodology that we used in benchmarking hash-based methods to generate graphs comparing speedup, recall, and ranking of the algorithms. As shown in Figure 8(a), Dumpy and Dumpy-Fuzzy significantly outperform other indexing-based methods. Additionally, the critical diagram indicates that the performance of Dumpy and Dumpy-Fuzzy is nearly identical. This similarity arises from Dumpy-Fuzzy being a variant of Dumpy, and these results are consistent with previous literature. Therefore, we can conclude that Dumpy and Dumpy-Fuzzy are the best-performing indexing-based ANNS methods **(RQ1)**.

## 4.5 Comparison of Partitioning based methods

This benchmark evaluates the performance of partition-based methods. As illustrated in the critical diagram in Figure 9(a), SCANN outperforms all other methods, with a connecting line to ANNOY indicating comparable performance. Moreover, Figure 9(c) highlights SCANN's superior average recall, which is approximately

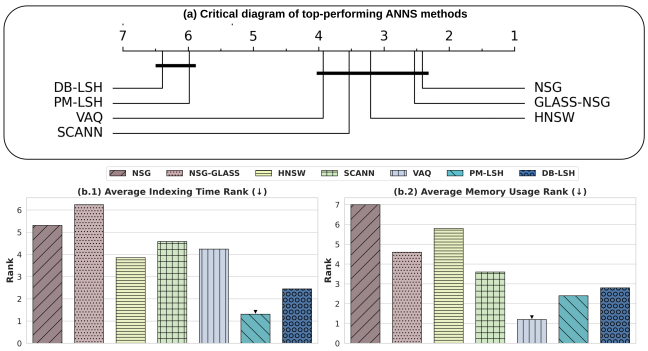


**Figure 10: (a) Ranking of best-performed ANNS methods based on WCSR (b) Average ranking of the best-performed methods based on indexing time and memory usage**

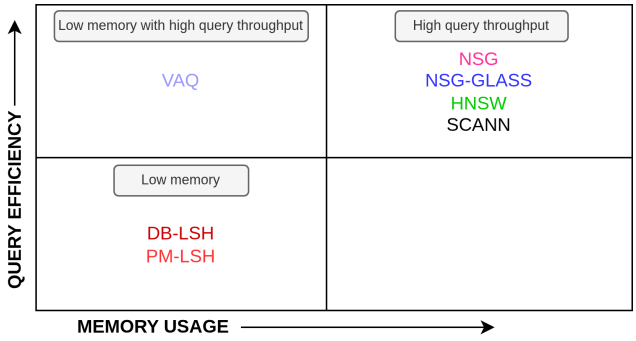


**Figure 11: Recommended method selection across considering query efficiency and memory usage—highlighting the most effective algorithms for each scenario.**

4% higher than that of ANNOY. A pairwise recall comparison further reveals that SCANN outperforms ANNOY on two-thirds of the datasets. Beyond recall, SCANN also exhibits greater efficiency in memory usage and index construction time compared to ANNOY. However, as shown in Figure 9(b), SCANN performs poorly on certain datasets, due to its optimization for MIPS distance [45]. Overall, considering recall, index construction time, and memory efficiency, SCANN emerges as the most effective partition-based method **(RQ1)**, with ANNOY as the next best alternative. In contrast, while KD-TREE achieves the fastest index construction time, its search speed is significantly lower. In this analysis, SCANN's retrieved neighbors were refined by examining $k$, $2k$, $3k$, $4k$, $5k$, and $10k$ candidates, as SCANN relies on quantization and does not compute distances on the raw data directly.

## 4.6 Comparison of top performed methods

Figure 10 presents the average rankings of the top-performing ANNS methods across three key performance dimensions: query efficiency (measured using the WCSR metric and validated through critical diagrams), construction time, and memory usage. The results reveal clear trade-offs among the evaluated methods. NSG achieves the highest speedup but is less efficient in indexing time and memory consumption. Conversely, PM-LSH and DB-LSH exhibit superior construction efficiency yet fall short in terms of query speedup. VAQ distinguishes itself with exceptional memory efficiency while maintaining competitive performance across both construction time and speedup. Notably, VAQ can compete with graph-based methods considering query efficiency. SCANN, in contrast, delivers balanced performance across all dimensions but does not emerge as the leading method in any single category (**RQ2**). These findings emphasize that each approach exhibits distinct strengths and weaknesses, highlighting the necessity of considering multiple evaluation criteria rather than privileging any single dimension. Furthermore, Figure 10(a) shows that the performance differences among NSG, SCANN, NSG-GLASS, HNSW, and VAQ are not statistically significant, as indicated by the connecting line in the critical diagram. In this analysis, the results for SCANN and VAQ were refined by retrieving $2k$, $3k$, $4k$, $5k$, and $10k$ nearest neighbors to more accurately identify the top $k$ nearest neighbors. This refinement was necessary as quantization-based methods approximate, rather than exactly compute, distances on the original vectors.

Our analysis highlights an inherent trade-off among construction time, memory usage, and query speedup, where gains in speed often require higher indexing time and memory consumption. As a result, no single ANNS method universally outperforms others; the optimal choice depends on application needs and available resources (**RQ3**).

# 5 DISCUSSION AND FUTURE RESEARCH

At first, we present the key findings of our analysis in Section 5.1. Followed by a brief discussion of the future research direction of this study in Section 5.2.

## 5.1 Summary of Findings

This research provides an extensive comparative analysis of 45 ANNS methods from five primary categories across 58 diverse datasets, evaluating recall, query efficiency, construction time, and memory usage. The discussion synthesizes both category-specific findings and cross-category insights to highlight methodological trade-offs, implementation nuances, and hardware dependencies that shape the performance landscape of ANNS algorithms.

Our evaluation began with intra-category analyses to identify the leading approaches within each algorithmic family. To strengthen the reliability of our findings, we incorporated statistical significance testing to determine whether observed performance gaps were meaningful rather than incidental. While several methods appeared to outperform others in terms of average recall, many of these differences were not statistically significant. For example, despite apparent disparities among graph-based methods, statistical analysis revealed no significant recall distinctions between NSG, NSG-GLASS, HNSW, HNSW-GLASS, and FLATNAV. Interestingly, the graph-based algorithm DISKANN, which performed poorly in prior benchmarks [101], emerges as one of the top-performing graph-based methods in our evaluation. This improvement likely stems from our use of the authors' original implementation, whereas previous studies reimplemented all methods from scratch. Similarly, recall differences between PM-LSH and DB-LSH were not statistically significant. In contrast, statistical testing detected a significant recall advantage favoring VAQ over OPQ, confirming VAQ as the

most effective quantization-based approach. The inclusion of statistical significance testing therefore, introduces a degree of rigor and interpretability absent in prior evaluations, ensuring that our conclusions rest on statistically validated evidence.

Building upon these statistically validated findings, our cross-category analysis reveals broader trends that extend beyond individual algorithmic families. Across four methodological categories, previously unreported top-performing methods emerge, VAQ and DUMPY exhibiting statistically significant improvements. These results expand upon prior benchmarks by identifying novel state-of-the-art approaches that had not been included in earlier studies, thereby broadening the empirical understanding of ANNS performance across diverse algorithmic paradigms. Building on these category-level insights, our global analysis underscores the fundamental trade-offs that characterize each methodological family. No single approach achieves universal superiority. Graph-based methods demonstrate exceptional query throughput but incur substantial construction time and memory overheads. Quantization-based techniques exhibit strong memory efficiency; however, VAQ successfully breaks this trend by delivering query efficiency comparable to the graph-based solutions while maintaining markedly lower memory consumption. While hash-based methods deliver the fastest construction times, yet lag in query efficiency. Based on our analysis, Figure 11 illustrates method recommendations tailored to specific performance priorities.

Interestingly, CPU architecture emerged as a decisive determinant of performance for hardware-accelerated methods. For example, in our analysis, SVS-LVQ achieved nearly a 20% throughput gain over SVS on Intel processors, yet its efficiency declined sharply on AMD CPUs. This divergence underscores a key limitation of architecture-unaware designs—methods that disregard processor heterogeneity risk unstable and non-portable performance across platforms. These observations emphasize the importance of architecture-conscious design to ensure robustness and reproducibility in real-world deployments. Furthermore, both indexing and hardware acceleration trade accuracy for throughput gains relative to their vanilla counterparts.

Beyond hardware sensitivity, our evaluation revealed significant implementation and stability challenges, particularly among indexing-based methods. ISAX2+ and DS-TREE, for instance, exhibited parameter-dependent instability, frequently leading to segmentation faults when insufficient candidate sets were generated. Likewise, VA+ demonstrated pronounced inconsistency, achieving strong results on certain datasets while performing poorly—with recall dropping below 0.1—on others.

Collectively, these findings highlight that the optimal ANNS method depends on application goals, computational resources, and trade-offs among recall, speed, and memory efficiency. Instead of prescribing a single best algorithm, this study provides a comprehensive empirical foundation to effectively guide method selection under varying system constraints.

## 5.2 Future Research

This benchmark evaluates a diverse selection of datasets and models; however, several promising research avenues remain to be explored. Future research directions include:

- **Alternative Distance Measures**: This analysis focuses on Euclidean distance, though extending it to angular distance and inner product similarity could yield further insights into the impact of similarity measures on performance.
- **Impact of SSD Storage**: The experiments in this benchmark utilized HDDs for storing datasets and indexes. Given the substantial performance differences in read/write speeds between HDDs and SSDs, future studies should assess how SSD storage technology influences ANNS performance.
- **Scalability to Billion-Scale Datasets**: The datasets considered in this study are of moderate scale, each containing up to three million instances. Extending evaluations to billion-sized datasets would greatly enhance the understanding of scalability challenges and highlight methods capable of maintaining efficiency at such scales.
- **Incorporation of ML-based Methods**: Learned indexes based on machine learning were not assessed in this benchmark. Integrating these approaches in future research would clarify their impacts on performance and efficiency in ANNS tasks.
- **Automated Parameter Tuning**: ANNS methods typically require significant manual parameter tuning, impacting usability. Developing robust algorithms capable of self-tuning with minimal training data is crucial. Research focusing on automatic parameter calibration has substantial practical significance and could greatly enhance user accessibility.
- **Implementation Sensitivity in ANNS Evaluation**: Future studies should investigate how performance varies across different implementations of the same algorithm, as engineering optimizations and design choices can significantly influence outcomes. Evaluating multiple implementations will help disentangle algorithmic advances from implementation effects, improving fairness in ANNS benchmarking.

Addressing these areas will refine benchmarks, enhance understanding of ANNS methods, and improve their applicability to a broader range of real-world scenarios.

# 6 CONCLUSION

This study delivers the most comprehensive evaluation of ANNS methods to date, addressing limitations of prior benchmarks that often overlooked recent advances. We analyze 45 state-of-the-art techniques across 58 diverse datasets, providing a rigorous view of the ANNS landscape. To ensure fairness, we introduce an efficiency measure based on the full speedup–recall curve and employ a statistical ranking framework to quantify performance differences and assess their significance. This enables the identification of leading methods within each algorithmic family while clarifying trade-offs among recall, efficiency, memory usage, and construction costs. Our global comparison yields three key insights. First, recent quantization-based methods can outperform graph-based approaches on certain datasets. Second, across four categories, new top-performing methods emerge, with two exhibiting statistically significant gains.. Third, hardware-accelerated methods exhibit architecture-dependent performance, highlighting the importance of hardware-aware design. By releasing all datasets and code, we enhance reproducibility and transparency, offering a practical resource for advancing ANNS research and applications.

# REFERENCES


[1] 2025. GitHub. https://github.com/saminkabir/ATLAS-benchmark. Accessed: 2026-01-15.

[2] 2025. Pecos. https://github.com/amzn/pecos. Accessed: 2025-10-17.

[3] 2025. scalar-quantization-and-product-quantization. https://zilliz.com/learn/scalar-quantization-and-product-quantization

[4] Cecilia Aguerrebere, Ishwar Singh Bhati, Mark Hildebrand, Mariano Tepper, and Theodore Willke. 2023. Similarity Search in the Blink of an Eye with Compressed Indices. *Proc. VLDB Endow.* 16, 11 (jul 2023), 3433–3446. https://doi.org/10.14778/3611479.3611537

[5] Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. 2015. Cache Locality is Not Enough: High-Performance Nearest Neighbor Search with Product Quantization Fast Scan. *Proc. VLDB Endow.* 9, 4 (dec 2015), 288–299. https://doi.org/10.14778/2856318.2856324

[6] arXiv.org submitters. 2024. arXiv Dataset. https://doi.org/10.34740/KAGGLE/DSV/7548853

[7] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems* 87 (2020), 101374. https://doi.org/10.1016/j.is.2019.02.006

[8] Ilias Azizi, Karima Echihabi, and Themis Palpanas. 2025. Graph-Based Vector Search: An Experimental Evaluation of the State-of-the-Art. *Proc. ACM Manag. Data* 3, 1, Article 43 (Feb. 2025), 31 pages. https://doi.org/10.1145/3709693

[9] Artem Babenko and Victor Lempitsky. 2012. The inverted multi-index. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 3069–3076. https://doi.org/10.1109/CVPR.2012.6248038

[10] Anthony Bagnall, Jason Lines, Aaron Bostrom, James Large, and Eamonn Keogh. 2017. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 31, 3 (01 May 2017), 606–660. https://doi.org/10.1007/s10618-016-0483-9

[11] Nurjahan Begum and Eamonn Keogh. 2014. Rare time series motif discovery from unbounded streams. *Proc. VLDB Endow.* 8, 2 (Oct. 2014), 149–160. https://doi.org/10.14778/2735471.2735476

[12] Davis W. Blalock and John V. Guttag. 2017. Bolt: Accelerated Data Mining with Fast Vector Compression. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*. ACM, 727–735. https://doi.org/10.1145/3097983.3098195

[13] Oren Boiman, Eli Shechtman, and Michal Irani. 2008. In defense of Nearest-Neighbor based image classification. *IEEE Conference in Computer Vision and Pattern Recognition*, 1 – 8.

[14] Paul Boniol, John Paparrizos, Yuhao Kang, Themis Palpanas, Ruey S. Tsay, Aaron J. Elmore, and Michael J. Franklin. 2022. Theseus: navigating the labyrinth of time-series anomaly detection. *Proc. VLDB Endow.* 15, 12 (Aug. 2022), 3702–3705. https://doi.org/10.14778/3554821.3554879

[15] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael Franklin. 2021. SAND in action: subsequence anomaly detection for streams. *Proceedings of the VLDB Endowment* 14 (07 2021), 2867–2870. https://doi.org/10.14778/3476311.3476365

[16] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J. Franklin. 2021. SAND in action: subsequence anomaly detection for streams. *Proc. VLDB Endow.* 14, 12 (July 2021), 2867–2870. https://doi.org/10.14778/3476311.3476365

[17] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. 2000. LOF: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (Dallas, Texas, USA) *(SIGMOD '00)*. Association for Computing Machinery, New York, NY, USA, 93–104. https://doi.org/10.1145/342009.335388

[18] Alessandro Camerra, Themis Palpanas, Jin Shieh, and Eamonn Keogh. 2010. iSAX 2.0: Indexing and Mining One Billion Time Series. In *2010 IEEE International Conference on Data Mining*. 58–67. https://doi.org/10.1109/ICDM.2010.124

[19] Alessandro Camerra, Jin Shieh, Themis Palpanas, Thanawin Rakthanmanon, and Eamonn Keogh. 2014. Beyond one billion time series: Indexing and mining very large time series collections with iSAX2+. *Knowledge and Information Systems* 39 (04 2014). https://doi.org/10.1007/s10115-012-0606-6

[20] Kaushik Chakrabarti, Eamonn Keogh, Sharad Mehrotra, and Michael Pazzani. 2002. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Trans. Database Syst.* 27, 2 (jun 2002), 188–228. https://doi.org/10.1145/568518.568520

[21] Patrick Chen, Wei-Cheng Chang, Jyun-Yu Jiang, Hsiang-Fu Yu, Inderjit Dhillon, and Cho-Jui Hsieh. 2023. FINGER: Fast Inference for Graph-based Approximate Nearest Neighbor Search. In *Proceedings of the ACM Web Conference 2023* (<conf-loc>, <city>Austin</city>, <state>TX</state>, <country>USA</country>, </conf-loc>) *(WWW '23)*. Association for Computing Machinery, New York, NY, USA, 3225–3235. https://doi.org/10.1145/3543507.3583318

[22] Qi Chen, Haidong Wang, Mingqin Li, Gang Ren, Scarlett Li, Jeffery Zhu, Jason Li, Chuanjie Liu, Lintao Zhang, and Jingdong Wang. 2018. *SPTAG: A library for fast approximate nearest neighbor search*. https://github.com/Microsoft/SPTAG

[23] Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. 2021. SPANN: Highly-efficient Billion-scale Approximate Nearest Neighbor Search. In *35th Conference on Neural Information Processing Systems (NeurIPS 2021)*.

[24] T. Cover and P. Hart. 1967. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory* 13, 1 (1967), 21–27. https://doi.org/10.1109/TIT.1967.1053964

[25] Michele Dallachiesa, Themis Palpanas, and Ihab F. Ilyas. 2014. Top-k nearest neighbor search in uncertain data series. *Proc. VLDB Endow.* 8, 1 (Sept. 2014), 13–24. https://doi.org/10.14778/2735461.2735463

[26] Donald W. Dearholt, Nathalie Gonzales, G. Kurup, Fox 3CRL, and NM Las Cruces. 1988. Monotonic Search Networks For Computer Vision Databases. *Twenty-Second Asilomar Conference on Signals, Systems and Computers* 2 (1988), 548–553. https://api.semanticscholar.org/CorpusID:53136542

[27] Gianna Del corso, Antonio Gulli, and Francesco Romani. 2005. Ranking a stream of news. 97–106. https://doi.org/10.1145/1060745.1060764

[28] Rui Ding, Qiang Wang, Yingnong Dang, Qiang Fu, Haidong Zhang, and Dongmei Zhang. 2015. YADING: fast clustering of large-scale time series data. *Proc. VLDB Endow.* 8, 5 (Jan. 2015), 473–484. https://doi.org/10.14778/2735479.2735481

[29] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2018. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. *Proc. VLDB Endow.* 12, 2 (oct 2018), 112–127. https://doi.org/10.14778/3282495.3282498

[30] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2019. Return of the Lernaean Hydra: Experimental Evaluation of Data Series Approximate Similarity Search. *Proc. VLDB Endow.* 13, 3 (nov 2019), 403–420. https://doi.org/10.14778/3368289.3368303

[31] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. 2000. Vector approximation based indexing for non-uniform high dimensional data sets. In *Proceedings of the Ninth International Conference on Information and Knowledge Management* (McLean, Virginia, USA) *(CIKM '00)*. Association for Computing Machinery, New York, NY, USA, 202–209. https://doi.org/10.1145/354756.354820

[32] I. R. I. for Seismology with Artificial Intelligence. 2015. *ANNOY (Approximate Nearest Neighbors Oh Yeah)*. https://github.com/spotify/annoy

[33] I. R. I. for Seismology with Artificial Intelligence. 2018. *Seismic Data Access*. http://ds.iris.edu/data/access/

[34] Steven Fortune. 2004. Voronoi Diagrams and Delaunay Triangulations. In *Handbook of Discrete and Computational Geometry, 2nd Ed.* https://api.semanticscholar.org/CorpusID:19663265

[35] Milton Friedman. 1937. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *J. Amer. Statist. Assoc.* 32 (1937), 675–701. https://api.semanticscholar.org/CorpusID:120581754

[36] Cong Fu and Deng Cai. 2016. EFANNA : An Extremely Fast Approximate Nearest Neighbor Search Algorithm Based on kNN Graph. *CoRR* abs/1609.07228 (2016). arXiv:1609.07228 http://arxiv.org/abs/1609.07228

[37] Cong Fu, Changxu Wang, and Deng Cai. 2022. High Dimensional Similarity Search With Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2022), 4139–4150. https://doi.org/10.1109/TPAMI.2021.3067706

[38] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search with the Navigating Spreading-out Graph. *Proc. VLDB Endow.* 12, 5 (jan 2019), 461–474. https://doi.org/10.14778/3303753.3303754

[39] Junhao Gan, Jianlin Feng, Qiong Fang, and Wilfred Ng. 2012. Locality-sensitive hashing scheme based on dynamic collision counting. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data* (Scottsdale, Arizona, USA) *(SIGMOD '12)*. Association for Computing Machinery, New York, NY, USA, 541–552. https://doi.org/10.1145/2213836.2213898

[40] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized Product Quantization for Approximate Nearest Neighbor Search. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2946–2953. https://doi.org/10.1109/CVPR.2013.379

[41] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity Search in High Dimensions via Hashing. In *Proceedings of the 25th International Conference on Very Large Data Bases (VLDB '99)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 518–529.

[42] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *Proceedings of the ACM Web Conference 2023* (Austin, TX, USA) *(WWW '23)*. Association for Computing Machinery, New York, NY, USA, 3406–3416. https://doi.org/10.1145/3543507.3583552

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

[44] Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. 2025. SymphonyQG: Towards Symphonious Integration of Quantization and Graph for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 1, Article 80 (Feb. 2025), 26 pages. https://doi.org/10.1145/3709730

[45] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3887–3896. http://proceedings.mlr.press/v119/guo20h.html

[46] Michael Held and Richard M. Karp. 1970. The Traveling-Salesman Problem and Minimum Spanning Trees. *Oper. Res.* 18, 6 (Dec. 1970), 1138–1162. https://doi.org/10.1287/opre.18.6.1138

[47] Bing Hu, Yanping Chen, and Eamonn Keogh. [n. d.]. *Time Series Classification under More Realistic Assumptions*. 578–586. https://doi.org/10.1137/1.9781611972832.64 arXiv:https://epubs.siam.org/doi/pdf/10.1137/1.9781611972832.64

[48] Qiang Huang, Jianlin Feng, Yikai Zhang, Qiong Fang, and Wilfred Ng. 2015. Query-aware locality-sensitive hashing for approximate nearest neighbor search. *Proc. VLDB Endow.* 9, 1 (sep 2015), 1–12. https://doi.org/10.14778/2850469.2850470

[49] Qiang Huang, Yifan Lei, and Anthony KH Tung. 2021. Point-to-Hyperplane Nearest Neighbor Search Beyond the Unit Hypersphere. In *Proceedings of the 2021 International Conference on Management of Data (SIGMOD)*. 777–789.

[50] Yizheng Huang and Jimmy Huang. 2024. A Survey on Retrieval-Augmented Text Generation for Large Language Models. arXiv:2404.10981 [cs.IR] https://arxiv.org/abs/2404.10981

[51] Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2019. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436* (2019).

[52] Ville Hyvönen, Teemu Pitkänen, Sotiris Tasoulis, Elias Jääsaari, Risto Tuomainen, Liang Wang, Jukka Corander, and Teemu Roos. 2016. Fast nearest neighbor search through sparse random projections and voting. In *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 881–888.

[53] Piotr Indyk and Rajeev Motwani. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing* (Dallas, Texas, USA) *(STOC '98)*. Association for Computing Machinery, New York, NY, USA, 604–613. https://doi.org/10.1145/276698.276876

[54] Masajiro Iwasaki and Daisuke Miyazaki. 2018. Optimization of Indexing Based on k-Nearest Neighbor Graph for Proximity Search in High-dimensional Data. *CoRR* abs/1810.07355 (2018). arXiv:1810.07355 http://arxiv.org/abs/1810.07355

[55] J.W. Jaromczyk and G.T. Toussaint. 1992. Relative neighborhood graphs and their relatives. *Proc. IEEE* 80, 9 (1992), 1502–1517. https://doi.org/10.1109/5.163414

[56] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547. https://doi.org/10.1109/TBDATA.2019.2921572

[57] Elias Jääsaari, Ville Hyvönen, Matteo Ceccarello, Teemu Roos, and Martin Aumüller. 2025. VIBE: Vector Index Benchmark for Embeddings. https://doi.org/10.48550/arXiv.2505.17810

[58] Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

[59] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases. *Knowledge and Information Systems* 3, 3 (01 Aug 2001), 263–286. https://doi.org/10.1007/PL00011669

[60] Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. 2021. GooAQ: Open Question Answering with Diverse Answer Types. *arXiv preprint* (2021).

[61] Jon M. Kleinberg. 2000. Navigation in a small world. *Nature* 406 (2000), 845–845. https://api.semanticscholar.org/CorpusID:4425543

[62] Atsutake Kosuge and Takashi Oshima. 2019. An Object-Pose Estimation Acceleration Technique for Picking Robot Applications by Using Graph-Reusing k-NN Search. In *First International Conference on Graph Computing, GC 2019, Laguna Hills, CA, USA, September 25-27, 2019*. IEEE, 68–74. https://doi.org/10.1109/GC46384.2019.00018

[63] Yifan Lei, Qiang Huang, Mohan Kankanhalli, and Anthony Tung. 2019. Sublinear Time Nearest Neighbor Search over Generalized Weighted Space. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 3773–3781. https://proceedings.mlr.press/v97/lei19a.html

[64] Yifan Lei, Qiang Huang, Mohan Kankanhalli, and Anthony K. H. Tung. 2020. Locality-Sensitive Hashing Scheme based on Longest Circular Co-Substring. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 2589–2599. https://doi.org/10.1145/3318464.3389778

[65] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS '20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.

[66] Conglong Li, Minjia Zhang, David G. Andersen, and Yuxiong He. 2020. Improving Approximate Nearest Neighbor Search through Learned Adaptive Early Termination. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data* (Portland, OR, USA) *(SIGMOD '20)*. Association for Computing Machinery, New York, NY, USA, 2539–2554. https://doi.org/10.1145/3318464.3380600

[67] Hao Li, W. Liu, and Heng Ji. 2014. Two-Stage Hashing for Fast Document Retrieval. In *Annual Meeting of the Association for Computational Linguistics*. https://api.semanticscholar.org/CorpusID:13057049

[68] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data — Experiments, Analyses, and Improvement. *IEEE Transactions on Knowledge and Data Engineering* 32, 8 (2020), 1475–1488. https://doi.org/10.1109/TKDE.2019.2909204

[69] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery* (San Diego, California) *(DMKD '03)*. Association for Computing Machinery, New York, NY, USA, 2–11. https://doi.org/10.1145/882082.882086

[70] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. 2015. Microsoft COCO: Common Objects in Context. arXiv:1405.0312 [cs.CV] https://arxiv.org/abs/1405.0312

[71] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. 2008. Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining* (2008), 413–422. https://api.semanticscholar.org/CorpusID:6505449

[72] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Deep Learning Face Attributes in the Wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[73] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* 60 (2004), 91–110. https://api.semanticscholar.org/CorpusID:174065

[74] Kejing Lu and Mineichi Kudo. 2020. R2LSH: A Nearest Neighbor Search Scheme Based on Two-dimensional Projected Spaces. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*. 1045–1056. https://doi.org/10.1109/ICDE48307.2020.00095

[75] Claudio Maccone. 2007. Advantages of Karhunen–Loève transform over fast Fourier transform for planetary radar and space debris detection. *Acta Astronautica* 60, 8 (2007), 775–779. https://doi.org/10.1016/j.actaastro.2006.08.015

[76] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (apr 2020), 824–836. https://doi.org/10.1109/TPAMI.2018.2889473

[77] Yitong Meng, Xiao Yan, Weiwen Liu, Huanhuan Wu, and James Cheng. 2020. Wasserstein Collaborative Filtering for Item Cold-start Recommendation. In *Proceedings of the 28th ACM Conference on User Modeling, Adaptation and Personalization* (<conf-loc>, <city>Genoa</city>, <country>Italy</country>, </conf-loc>) *(UMAP '20)*. Association for Computing Machinery, New York, NY, USA, 318–322. https://doi.org/10.1145/3340631.3394870

[78] Marius Muja and David G. Lowe. 2009. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*. INSTICC Press, 331–340.

[79] Blaise Munyampirwa, Vihan Lakshman, and Benjamin Coleman. 2024. Down with the Hierarchy: The'H'in HNSW Stands for" Hubs". *arXiv preprint arXiv:2412.01940* (2024).

[80] Peter Bjorn Nemenyi. 1963. *Distribution-free multiple comparisons*. Princeton University.

[81] Aude Oliva and Antonio Torralba. 2001. Modeling the Shape of the Scene: A Holistic Representation of the Spatial Envelope. *International Journal of Computer Vision* 42 (2001), 145–175. https://api.semanticscholar.org/CorpusID:11664336

[82] John Paparrizos, Ikraduya Edian, Chunwei Liu, Aaron J. Elmore, and Michael J. Franklin. 2022. Fast Adaptive Similarity Search through Variance-Aware Quantization. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. 2969–2983. https://doi.org/10.1109/ICDE53745.2022.00268

[83] John Paparrizos and Michael J. Franklin. 2019. GRAIL: efficient time-series representation learning. *Proc. VLDB Endow.* 12, 11 (jul 2019), 1762–1777. https://doi.org/10.14778/3342263.3342648

[84] John Paparrizos and Luis Gravano. 2016. k-Shape: Efficient and Accurate Clustering of Time Series. *SIGMOD Rec.* 45, 1 (June 2016), 69–76. https:

//doi.org/10.1145/2949741.2949758
[85] John Paparrizos and Luis Gravano. 2017. Fast and Accurate Time-Series Clustering. *ACM Trans. Database Syst.* 42, 2, Article 8 (June 2017), 49 pages. https://doi.org/10.1145/3044711
[86] John Paparrizos, Chunwei Liu, Bruno Barbarioli, John Hwang, Ikraduya Edian, Aaron J. Elmore, Michael J. Franklin, and Sanjay Krishnan. 2021. VergeDB: A Database for IoT Analytics on Edge Devices. In *Conference on Innovative Data Systems Research*. https://api.semanticscholar.org/CorpusID:231906525
[87] Leonid Pekelis, Michael Feil, Forrest Moret, Mark Huang, and Tiffany Peng. 2024. Llama 3 Gradient: A series of long context models. https://gradient.ai/blog/scaling-rotational-embeddings-for-long-context-language-models
[88] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web* (Hong Kong, Hong Kong) *(WWW '01)*. Association for Computing Machinery, New York, NY, USA, 285–295. https://doi.org/10.1145/371920.372071
[89] Kunal Sawarkar, Abhilasha Mangal, and Shivam Raj Solanki. 2024. Blended RAG: Improving RAG (Retriever-Augmented Generation) Accuracy with Semantic Search and Hybrid Query-Based Retrievers. In *2024 IEEE 7th International Conference on Multimedia Information Processing and Retrieval (MIPR)*. 155–161. https://doi.org/10.1109/MIPR62202.2024.00031
[90] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646. https://doi.org/10.1109/JIOT.2016.2579198
[91] Jin Shieh and Eamonn Keogh. 2008. iSAX: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Las Vegas, Nevada, USA) *(KDD '08)*. Association for Computing Machinery, New York, NY, USA, 623–631. https://doi.org/10.1145/1401890.1401966
[92] Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. 2021. FreshDiskANN: A Fast and Accurate Graph-Based ANN Index for Streaming Similarity Search. *CoRR* abs/2105.09613 (2021). arXiv:2105.09613 https://arxiv.org/abs/2105.09613
[93] Tomás Skopal, Jaroslav Pokorný, and Václav Snásel. 2004. PM-tree: Pivoting Metric Tree for Similarity Search in Multimedia Databases. In *Advances in Databases and Information Systems, 8th East European Conference, ADBIS 2004, Budapest, Hungary, September 22-25, 2004, Local Proceeding*. http://www.sztaki.hu/conferences/ADBIS/9-Skopal.pdf
[94] Soldi, S., Beckmann, V., Baumgartner, W. H., Ponti, G., Shrader, C. R., Lubiński, P., Krimm, H. A., Mattana, F., and Tueller, J. 2014. Long-term variability of AGN at hard X-rays. *AA* 563 (2014), A57. https://doi.org/10.1051/0004-6361/201322653
[95] Suhas Jayaram Subramanya, Devvrit, Rohan Kadekodi, Ravishankar Krishnaswamy, and Harsha Simhadri. 2019. DiskANN: Fast Accurate Billion-point Nearest Neighbor Search on a Single Node. In *NeurIPS 2019*. https://www.microsoft.com/en-us/research/publication/diskann-fast-accurate-billion-point-nearest-neighbor-search-on-a-single-node/
[96] Yao Tian, Xi Zhao, and Xiaofang Zhou. 2022. DB-LSH: Locality-Sensitive Hashing with Query-based Dynamic Bucketing. In *ICDE*. IEEE, 2250–2262.
[97] Yao Tian, Xi Zhao, and Xiaofang Zhou. 2024. DB-LSH 2.0: Locality-Sensitive Hashing With Query-Based Dynamic Bucketing. *IEEE Transactions on Knowledge and Data Engineering* 36, 3 (2024), 1000–1015. https://doi.org/10.1109/TKDE.2023.3295831
[98] Antonio Torralba, Rob Fergus, and Yair Weiss. 2008. Small codes and large image databases for recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 1–8. https://doi.org/10.1109/CVPR.2008.4587633
[99] Javier Vargas Muñoz, Marcos A. Gonçalves, Zanoni Dias, and Ricardo da S. Torres. 2019. Hierarchical Clustering-Based Graphs for Large Scale Approximate Nearest Neighbor Search. *Pattern Recogn.* 96, C (dec 2019), 10 pages. https://doi.org/10.1016/j.patcog.2019.106970
[100] Jingdong Wang, Naiyan Wang, You Jia, Jian Li, Gang Zeng, Hongbin Zha, and Xian-Sheng Hua. 2014. Trinary-Projection Trees for Approximate Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 2 (2014), 388–403.
[101] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 14, 11 (jul 2021), 1964–1978. https://doi.org/10.14778/3476249.3476255
[102] Yang Wang, Peng Wang, Jian Pei, Wei Wang, and Sheng Huang. 2013. A data-adaptive and dynamic segmentation index for whole matching on time series. *Proc. VLDB Endow.* 6, 10 (aug 2013), 793–804. https://doi.org/10.14778/2536206.2536208
[103] Yang Wang, Peng Wang, Jian Pei, Wei Wang, and Sheng Huang. 2013. A data-adaptive and dynamic segmentation index for whole matching on time series. *Proc. VLDB Endow.* 6, 10 (aug 2013), 793–804. https://doi.org/10.14778/2536206.2536208
[104] Zihao Wang. 2025. Graph Library for Approximate Similarity Search. https://github.com/zilliztech/pyglass
[105] Zeyu Wang, Qitong Wang, Peng Wang, Themis Palpanas, and Wei Wang. 2023. Dumpy: A Compact and Adaptive Index for Large Data Series Collections. *Proc. ACM Manag. Data* 1, 1, Article 111 (may 2023), 27 pages. https://doi.org/10.1145/3588965
[106] Dongtao Wei, Kaixiang Zhuang, Lei Ai, Qunlin Chen, Wenjing Yang, Wei Liu, Kangcheng Wang, Jiangzhou Sun, and Jiang Qiu. 2018. Structural and functional brain scans from the cross-sectional Southwest University adult lifespan dataset. *Scientific Data* 5, 1 (17 Jul 2018), 180134. https://doi.org/10.1038/sdata.2018.134
[107] T. Weyand, A. Araujo, B. Cao, and J. Sim. 2020. Google Landmarks Dataset v2 - A Large-Scale Benchmark for Instance-Level Recognition and Retrieval. In *Proc. CVPR*.
[108] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (1945), 80–83. http://www.jstor.org/stable/3001968
[109] Jack Woollam, Jannes Münchmeyer, Frederik Tilmann, Andreas Rietbrock, Dietrich Lange, Thomas Bornstein, Tobias Diehl, Carlo Giunchi, Florian Haslinger, Dario Jozinović, Alberto Michelini, Joachim Saul, and Hugo Soto. 2022. SeisBench—A Toolbox for Machine Learning in Seismology. *Seismological Research Letters* 93, 3 (03 2022), 1695–1709. https://doi.org/10.1785/0220210324 arXiv:https://pubs.geoscienceworld.org/ssa/srl/article-pdf/93/3/1695/5596371/srl-2021324.1.pdf
[110] Yuming Xu, Hengyu Liang, Jin Li, Shuotao Xu, Qi Chen, Qianxi Zhang, Cheng Li, Ziyue Yang, Fan Yang, Yuqing Yang, et al. 2023. SPFresh: Incremental In-Place Update for Billion-Scale Vector Search. In *Proceedings of the 29th Symposium on Operating Systems Principles*. 545–561.
[111] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2016. Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. 1317–1322. https://doi.org/10.1109/ICDM.2016.0179
[112] Chin-Chia Michael Yeh, Yan Zhu, Liudmila Ulanova, Nurjahan Begum, Yifei Ding, Hoang Anh Dau, Zachary Zimmerman, Diego Furtado Silva, Abdullah Mueen, and Eamonn Keogh. 2018. Time series joins, motifs, discords and shapelets: a unifying view that exploits the matrix profile. *Data Min. Knowl. Discov.* 32, 1 (Jan. 2018), 83–123. https://doi.org/10.1007/s10618-017-0519-9
[113] Peter N. Yianilos. 1993. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms* (Austin, Texas, USA) *(SODA '93)*. Society for Industrial and Applied Mathematics, USA, 311–321.
[114] Xi Zhao, Yao Tian, Kai Huang, Bolong Zheng, and Xiaofang Zhou. 2023. Towards Efficient Index Construction and Approximate Nearest Neighbor Search in High-Dimensional Spaces. *Proc. VLDB Endow.* 16, 8 (apr 2023), 1979–1991. https://doi.org/10.14778/3594512.3594527
[115] Bolong Zheng, Xi Zhao, Lianggui Weng, Nguyen Quoc Viet Hung, Hang Liu, and Christian S. Jensen. 2020. PM-LSH: A fast and accurate LSH framework for high-dimensional approximate NN search. *Proc. VLDB Endow.* 13, 5 (jan 2020), 643–655. https://doi.org/10.14778/3377369.3377374

Received 20 February 2007; revised 12 March 2009; accepted 5 June 2009