Noname manuscript No.
(will be inserted by the editor)

Survey of Filtered Approximate Nearest Neighbor Search over the Vector-Scalar Hybrid Data

Yanjun Lin · Kai Zhang · Zhenying He · Yinan Jing · X. Sean Wang

Received: date / Accepted: date

Abstract Filtered approximate nearest neighbor search (FANNS), an extension of approximate nearest neighbor search (ANNS) that incorporates scalar filters, has been widely applied to constrained retrieval of vector data. Despite its growing importance, no dedicated survey on FANNS over the vector-scalar hybrid data currently exists, and the field has several problems, including inconsistent definitions of the search problem, insufficient framework for algorithm classification, and incomplete analysis of query difficulty. This survey paper formally defines the concepts of hybrid dataset and hybrid query, as well as the corresponding evaluation metrics. Based on these, a pruning-focused framework is proposed to classify and summarize existing algorithms, providing a broader and finer-grained classification framework compared to the existing ones. In addition, a review is conducted on representative hybrid datasets, followed by an analysis on the difficulty of hybrid queries from the perspective of distribution relationships between data and queries. This paper aims to establish a structured foundation for FANNS over the vector-scalar hybrid data, facilitate more meaningful comparisons between FANNS algorithms, and offer practical recommendations for practitioners. The code used for downloading hybrid datasets and analyzing query difficulty is available at `https://github.com/lyj-fdu/FANNS`.

Keywords Filtered Approximate Nearest Neighbor Search · Filtered Similarity Search · Hybrid Search · Hybrid Query · Vector-Scalar Hybrid Data

Yanjun Lin
School of Computer Science, Fudan University, Shanghai, China
E-mail: yjlin24@m.fudan.edu.cn

Kai Zhang
School of Computer Science, Fudan University, Shanghai, China
E-mail: zhangk@fudan.edu.cn

Zhenying He
School of Computer Science, Fudan University, Shanghai, China
E-mail: zhenying@fudan.edu.cn

Yinan Jing
School of Computer Science, Fudan University, Shanghai, China
E-mail: jingyn@fudan.edu.cn

X. Sean Wang
School of Computer Science, Fudan University, Shanghai, China
E-mail: xywangCS@fudan.edu.cn

1 Introduction

Nearest neighbor search (NNS) over the vector data has been widely used in various real-world applications, such as embedding-based retrieval (EBR) for search engines and recommendation systems [12, 27, 38, 59, 61, 65], retrieval-augmented generation (RAG) for large language models (LLMs) [31, 42, 43, 56, 103, 104], and many other cross-disciplinary usages [1, 9, 16, 34, 52, 53]. Typically, unstructured data (e.g., images, texts, or audio) are first encoded into high-dimensional feature vectors using embedding models (e.g., CNN [37, 55, 85], Transformer [13, 25, 90], or VGGish [1]), and then nearest neighbor search is performed by retrieving the nearest vectors to a query vector based on a vector similarity function. To alleviate the prohibitively high computational complexity of exact NNS, approximate nearest neighbor search (ANNS) has been proposed to relax

[1] `https://github.com/tensorflow/models/tree/master/research/audioset/vggish`

the requirement for exactness while significantly improving the search efficiency. ANNS is typically implemented using a well-designed index, which can be hash-based [17, 32, 39, 51, 54], tree-based [11, 21, 23, 57, 74], quantization-based [19, 49, 69, 70, 75], or graph-based [29, 30, 67, 68, 79, 86, 101].

Filtered nearest neighbor search (FNNS) over the vector-scalar hybrid data extends NNS to retrieving only the nearest vectors among those that satisfy a given scalar filter. It has attracted increasing attention over the past decade. In the context of EBR, users on an e-commerce platform may need to find products most similar to an item in a given image, while applying a filter on scalars like color or price [96]. For RAG, in order to ensure the freshness of responses from LLMs, time-aware RAG can be achieved by assigning different weights to document timestamps during retrieval [31]. In the case of industrial-scale knowledge graphs (KGs) such as Saga [44], users can find related entities via KG embeddings, while also specifying the entity type and the values they contain [73]. In the literature, there has been a surge of studies [15, 28, 33, 35, 73, 78, 91, 93, 96–100, 102, 105, 107] focused on solving the problem of filtered approximate nearest neighbor search (FANNS) over the vector-scalar hybrid data.

1.1 Motivation

While a number of survey papers have been published on ANNS over the vector data [6, 14, 60, 77, 92, 94], there is currently no survey on FANNS over the vector-scalar hybrid data. Below, we outline three key reasons (**R1**-**R3**) that highlight the need for such a survey.

R1: Inconsistent definitions of the search problem. In FANNS, datasets and queries containing both vectors and scalars are referred as hybrid datasets and hybrid queries, respectively. However, the definitions of these terms vary across studies. For hybrid datasets, some define scalars as simple numbers [28, 35, 99, 107], some as collections of labels [15, 33], and others as values within a schema [73, 78, 91, 93, 96–98, 100, 102, 105] similar to that in a relational database. For hybrid queries, some define scalar filters as equality comparisons [15, 33, 93, 97, 98], some as range comparisons [28, 99, 107], and others as general operations [35, 73, 78, 91, 96, 100, 102, 105]. Additionally, there is inconsistency in how evaluation metrics are defined. For example, some studies define the *selectivity* as the proportion of data points that do not satisfy the scalar filter [91, 96], while others define it as the proportion of data points that do [73, 78, 102]. The latter definition is also referred to as the *specificity* in some studies [15, 33].

R2: Insufficient framework for algorithm classification. Current framework classifies FANNS algorithms based on *when* the scalar filter is applied, typically into three categories: pre-filtering, post-filtering, and in-filtering [28, 33, 35, 73, 78, 99], which correspond to removing data points that do not satisfy the filter before, after, or during the ANNS. However, this framework has two major shortcomings. First, it is not enough to cover all algorithms. For example, some algorithms [28, 73] first apply part of the scalar filter to identify relevant data partitions, then perform ANNS within them, and finally apply the complete filter. In this case, the filter is applied in two stages, which fails to fit any of the aforementioned three categories. Second, it is too coarse to distinguish between algorithms. For example, in graph-based indices, the filter can be applied during either result update [102] or neighborhood expansion [78], but only the latter avoids unnecessary vector similarity calculations. Although both are classified as in-filtering, their effects differ significantly.

R3: Incomplete analysis of query difficulty. In the context of ANNS, considerable efforts have been made to understand query difficulty through factors such as *relative contrast* [5, 36], *intrinsic dimensionality* [4, 5, 41], *query expansion* [3, 5], ϵ*-hardness* [106], and *Steiner-hardness* [95]. In contrast, for FANNS, factors that impact the difficulty of hybrid queries remain underexplored. At present, *selectivity* is the only commonly used factor for evaluating the hybrid query difficulty. Identifying more factors is essential not only for explaining algorithm performance fluctuations from various perspectives, but also for constructing evaluation benchmarks across a wider range of difficulty levels by their combination.

1.2 Our Contributions

Driven by the above discussion, we present a survey on FANNS over the vector-scalar hybrid data, covering its definitions, algorithms, datasets and query difficulty. Below, we summarize our main contributions.

(1) More systematic definition of the search problem. For **R1**, we formally define the problem of FANNS by specifying the hybrid dataset, the hybrid query, and relevant evaluation metrics (section 2). Specifically, a hybrid dataset is defined as one in which each data point is a pair of a scalar-tuple following a scalar schema and a vector in a vector space. A hybrid query is defined as comprising a scalar filter, a vector similarity function, a query vector, and a target result size. Key evaluation

metrics including *recall* and *selectivity* are defined with precise semantics to eliminate ambiguity.

(2) Finer-grained classification of FANNS algorithms. For **R2**, we propose a pruning-focused framework to classify FANNS algorithms (section 3). Our framework comprises four distinct strategies, each emphasizing either vector pruning or scalar pruning: vector-solely pruning (VSP), vector-centric joint pruning (VJP), scalar-centric joint pruning (SJP), and scalar-solely pruning (SSP). Building on this framework, we summarize existing FANNS algorithms using the unified terminology in our formal definitions, effectively classifying them and revealing their interrelationships. Our framework provides a broader and finer-grained classification compared to the existing one.

(3) Deeper analysis of query difficulty through a new factor. For **R3**, we examine existing hybrid datasets (section 4) and analyze the factors that impact the difficulty of hybrid queries (section 5). Specifically, we collect hybrid datasets used in existing FANNS studies, discuss their underlying construction strategies, and detail their main characteristics. Motivated by realistic scenarios, we identify the *distribution* factor, which refers to the relationship between high dimensional distributions of two sets of vectors. We verify the impact of the *distribution* factor to the hybrid query difficulty by conducting a case study, offering qualitative explanations based on UMAP visualizations [72] and quantitative insights using the Wasserstein distance [66]. We finally propose a schema towards more comprehensive evaluation of FANNS algorithms combining both *selectivity* and *distribution* factors.

2 Preliminaries

In this section, we first formally define the hybrid dataset, the hybrid query, and the corresponding evaluation metrics. Then, we outline representative ANNS algorithms, which serve as the foundation for FANNS algorithms. Table 1 summarizes the notations used in this paper.

2.1 Definition of Hybrid Dataset

We begin with the definitions of scalar schema and vector space, followed by a formal definition of hybrid dataset.

Definition 1 (Scalar Schema) Let $\mathbb{S} = (\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_m)$ be a *scalar schema* containing m *scalars*, where each scalar $\mathbb{S}_i$ has a simple data type, such as integer, float, or string.

Table 1 Summary of Notations

Notation	Description
$\mathbb{S}$	A scalar schema, consisting of m scalars, i.e., $\mathbb{S} = (\mathbb{S}_1, \mathbb{S}_2, \ldots, \mathbb{S}_m)$, where each scalar $\mathbb{S}_j$ has a simple data type
$\mathbb{V}$	A vector space, specifically the d-dimensional vector space over the field of real numbers, i.e., $\mathbb{V} = \mathbb{R}^d$
$\mathcal{D}$	A hybrid dataset, consisting of n data points, i.e., $\mathcal{D} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, where each data point $\mathbf{p}_i = (\mathbf{s}_i, \mathbf{v}_i)$ has a scalar-tuple $\mathbf{s}_i \in \mathbb{S}$ and a vector $\mathbf{v}_i \in \mathbb{R}^d$
f_s	A scalar filter, $f_s : \mathbb{S} \to \{0, 1\}$, which evaluates a scalar-tuple in $\mathbb{S}$ to false or true
$\mathcal{D}_{f_s}$	A filtered subset, $\mathcal{D}_{f_s} = \{\mathbf{p} \in \mathcal{D} \mid f_s(\mathbf{p}.\mathbf{s}) = 1\}$, which contains data points that satisfy f_s in $\mathcal{D}$
f_v	A vector similarity function, $f_v : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, which measures the similarity between two vectors in $\mathbb{R}^d$, where smaller values indicate higher similarity
q	A hybrid query, $q = (f_s, f_v, \mathbf{v}_q, k)$, containing a scalar filter f_s, a vector similarity function f_v, a query vector $\mathbf{v}_q$, and target result size k
$\lvert\cdot\rvert$	The cardinality of a set
$recall@k$	The recall to measure the accuracy of a hybrid query with a target result size k, defined as $recall = \frac{\lvert\mathcal{R} \cap \tilde{\mathcal{R}}\rvert}{\lvert\mathcal{R}\rvert}$, where $\mathcal{R}$ and $\tilde{\mathcal{R}}$ are the result for FNNS and FANNS, respectively
sel_{f_s}	The selectivity of a scalar filter f_s, defined as $sel_{f_s} = 1 - \frac{\lvert\mathcal{D}_{f_s}\rvert}{\lvert\mathcal{D}\rvert}$, which is the fraction of data points that do not satisfy the scalar filter f_s.

Definition 2 (Vector Space) Let $\mathbb{V}$ be a *vector space*, which is a set of vectors that are closed under vector addition and constant multiplication. In this paper, we specifically consider the d-dimensional vector space over the field of real numbers, i.e., $\mathbb{V} = \mathbb{R}^d$, where each vector is a d-dimensional tuple of real numbers.

Definition 3 (Hybrid Dataset) Let $\mathcal{D} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\} = \{(\mathbf{s}_1, \mathbf{v}_1), (\mathbf{s}_2, \mathbf{v}_2), \ldots, (\mathbf{s}_n, \mathbf{v}_n)\}$ be a *hybrid dataset* containing n data points, where each *data point* $\mathbf{p}_i = (\mathbf{s}_i, \mathbf{v}_i)$ is a pair of a *scalar-tuple* $\mathbf{s}_i \in \mathbb{S}$ (also denoted as $\mathbf{p}_i.\mathbf{s}$) and a *vector* $\mathbf{v}_i \in \mathbb{R}^d$ (also denoted as $\mathbf{p}_i.\mathbf{v}$). Each scalar-tuple $\mathbf{s}_i = (s_{i,1}, s_{i,2}, \ldots, s_{i,m})$ is a value in $\mathbb{S}$, where each $s_{i,j}$ is a *scalar value* in $\mathbb{S}_i$ and can either take a value of the corresponding data type or be NULL. Each vector $\mathbf{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,d})$ is a value in $\mathbb{R}^d$, where each $v_{i,j}$ is a real number in $\mathbb{R}$.

2.2 Definition of Hybrid Query

We proceed to define the corresponding scalar filter and vector similarity function, along with a formal definition of hybrid query.

Definition 4 (Scalar Filter) Let $f_s : \mathbb{S} \to \{0, 1\}$ be a *scalar filter* that evaluates the scalar-tuple of a data point $\mathbf{p}$ to either false ($f_s(\mathbf{p}.\mathbf{s}) = 0$) or true ($f_s(\mathbf{p}.\mathbf{s}) = 1$). $\mathcal{D}_{f_s} = \{\mathbf{p} \in \mathcal{D} \mid f_s(\mathbf{p}.\mathbf{s}) = 1\}$ is the *filtered subset* whose data points are taken from $\mathcal{D}$ and *satisfy* f_s.

A scalar filter can vary in complexity, ranging from simple constraints like binary comparisons to more complex constraints like regular expressions. A scalar filter that allows arbitrary constraints is referred to as a *general scalar filter*. Unless otherwise specified, a scalar filter is assumed to be a general scalar filter.

Some algorithms [28, 33, 93, 97–99, 107], discussed later, simplify the scalar filter for specialized applications. A *simplified scalar filter* consists of simple constraints that check whether a scalar equals a specific value (*equality constraints*) or falls within a value range (*range constraints*). A *simplified scalar filter* with only *equality constraints* is called a *simplified equality scalar filter*, while one with only *range constraints* is called a *simplified range scalar filter*. For ease of analysis, we express a *simplified scalar filter* in the disjunctive normal form (DNF):

$$f_s(\mathbf{p}.\mathbf{s}) = \bigvee_{i=1}^{t} \left(\bigwedge_{j=1}^{m} f_{i,j}(\mathbf{p}.s_j) \right) \in \{0, 1\}, \tag{1}$$

where the j-th *sub-filter* $f_{i,j} : \mathbb{S}_j \to \{0, 1\}$ imposes either no constraint or a simple constraint on the j-th scalar $\mathbf{p}.s_j \in \mathbb{S}_j$. A sub-filter always evaluates to true when no constraint is imposed, and is referred to as an *active sub-filter* when a simple constraint is specified.

Definition 5 (Vector Similarity Function) Let $f_v : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a *vector similarity function* that measures the similarity between the vectors of two data points $\mathbf{p}_x$ and $\mathbf{p}_y$ to a real number $f_v(\mathbf{p}_x.\mathbf{v}, \mathbf{p}_y.\mathbf{v}) \in \mathbb{R}$, where smaller values indicate higher similarity.

Given two vectors $\mathbf{p}_x.\mathbf{v} = (v_{x,1}, v_{x,2}, \ldots, v_{x,d})$ and $\mathbf{p}_y.\mathbf{v} = (v_{y,1}, v_{y,2}, \ldots, v_{y,d})$ in $\mathbb{R}^d$, the form of a vector similarity function f_v can vary depending on the specific application.

A common approach is to directly define the vector similarity function using a distance function, which obeys the properties of non-negativity, identity, symmetry, and the triangle inequality. For example, the Euclidean distance:

$$d(\mathbf{p}_x.\mathbf{v}, \mathbf{p}_y.\mathbf{v}) = \sqrt{\sum_{i=1}^{d} (v_{x,i} - v_{y,i})^2} \in [0, +\infty), \tag{2}$$

can serve directly as a vector similarity function, i.e., $f_v(\cdot, \cdot) = d(\cdot, \cdot)$. A smaller Euclidean distance value indicates that two vectors are closer in the vector space, implying higher similarity.

Alternatively, a non-distance function can be used to indirectly to define the vector similarity function. For example, the inner product:

$$\langle \mathbf{p}_x.\mathbf{v}, \mathbf{p}_y.\mathbf{v} \rangle = \sum_{i=1}^{d} v_{x,i} v_{y,i} \in \mathbb{R}, \tag{3}$$

can be transformed into a vector similarity function by taking its negation, i.e., $f_v(\cdot, \cdot) = -\langle \cdot, \cdot \rangle$. A larger inner product value (smaller in negative) indicates that two vectors are more aligned in direction, implying higher similarity.

Definition 6 (Hybrid Query) Let $q = \{f_s, f_v, \mathbf{v}_q, k\}$ be a *hybrid query*, which contains a scalar filter f_s, a vector similarity function f_v, a *query vector* $\mathbf{v}_q \in \mathbb{R}^d$, and a *target result size* k.

Given a hybrid dataset $\mathcal{D}$ and a hybrid query q, the problems of FNNS is to find a result set $\mathcal{R} \subseteq \mathcal{D}_{f_s}$ containing $\min(k, |\mathcal{D}_{f_s}|)$ data points whose vectors are the most similar to $\mathbf{v}_q$ under f_v when only considering vectors in $\mathcal{D}_{f_s}$. This can be formally expressed as:

$$\mathcal{R} = \arg \min_{\substack{\mathcal{S} \subseteq \mathcal{D}_{f_s}, \\ |\mathcal{S}| = \min(k, |\mathcal{D}_{f_s}|)}} \sum\nolimits_{\mathbf{p} \in \mathcal{S}} f_v(\mathbf{p}.\mathbf{v}, \mathbf{v}_q). \tag{4}$$

FANNS relaxes FNNS by allowing a small number of errors in the result set, denoted $\tilde{\mathcal{R}}$, thereby trading accuracy for efficiency. The accuracy of FANNS is typically measured by recall, as defined in subsection 2.3. It is worth noting that when the scalar filter f_s always evaluates to true, FNNS and FANNS reduce to traditional NNS and ANNS, respectively.

2.3 Definitions of Evaluation Metrics

This subsection clarifies the semantics of key evaluation metrics, including *recall* and *selectivity*.

Definition 7 (Recall) Let *recall@k* be the *recall* that measures the accuracy of a hybrid query with a target result size k, which is defined as follows:

$$recall@k = \frac{|\mathcal{R} \cap \tilde{\mathcal{R}}|}{|\mathcal{R}|} \in [0, 1]. \tag{5}$$

In this equation, $\mathcal{R}$ denotes the ground truth result set obtained from FNNS, while $\tilde{\mathcal{R}}$ represents the result returned by a given query method, which may correspond to either FNNS or FANNS. The numerator, $|\mathcal{R} \cap \tilde{\mathcal{R}}|$, indicates the number of ground truth results that are successfully retrieved. The denominator, $|\mathcal{R}|$, reflects the total number of ground truth results. Note

that $|\mathcal{R}| \leq k$, as the *filtered subset* $\mathcal{D}_{f_s}$ may contain fewer than k data points.

A larger *recall* indicates higher quality of the result. Specifically, the *recall* of FNNS is 1, and the *recall* of FANNS is between 0 and 1 due to its approximate nature. If multiple vectors have the same similarity to $\mathbf{v}_q$ under f_v, the result set may not be unique, in this case, *distance-based recall* variants are available [6].

Definition 8 (Selectivity) Let sel_{f_s} be the *selectivity* that measures the fraction of data points that do **not** satisfy the scalar filter f_s, which is defined as follows:

$$sel_{f_s} = 1 - \frac{|\mathcal{D}_{f_s}|}{|\mathcal{D}|} \in [0, 1]. \tag{6}$$

In the existing literature, the definition of *selectivity* remains inconsistent. Some studies define it as the proportion of data points that do **not** satisfy the scalar filter [91, 96], while others define it as the proportion of data points that satisfy the filter [73, 78, 102]. In this work, we adopt the former definition, where higher selectivity indicates that a larger portion of the dataset is filtered out. This aligns with the intuitive understanding that a process is "more selective" when it excludes more candidates. The latter definition is more appropriately referred to as *specificity* [15, 33].

A scalar filter f_s is said to be *selective* if sel_{f_s} is close to 1, *unselective* if it is close to 0, and *moderate* if it lies in between.

2.4 Background of ANNS Algorithms

Early research on ANNS primarily focused on hash-based [17, 32, 39, 51, 54] and tree-based [11, 21, 23, 57, 74] indices, as they are natural extensions of traditional indexing structures in relational databases to high-dimensional spaces. However, despite their favorable theoretical complexity, these methods fail to scale effectively when the vector dimensionality exceeds 10 [26], largely due to the "curse of dimensionality" [45].

In recent years, the focus of ANNS has shifted toward quantization-based [19, 49, 69, 70, 75] and graph-based [29, 30, 67, 68, 79, 86, 101] indices, which have demonstrated significantly better empirical performance under different efficiency-accuracy trade-offs [26]. As a result, almost all existing FANNS algorithms are built upon adaptations of these two index structures. Therefore, we briefly introduce the IVF index and the graph index as representative ANNS methods to facilitate the subsequent discussion of the FANNS algorithms.

Algorithm 1: Search on the IVF Index

Input: *nlist* centroids $\mathcal{C} = \{\mathbf{v}_{c_1}, \ldots, \mathbf{v}_{c_{nlist}}\}$ with inverted lists $\mathcal{L} = \{l_{c_1}, \ldots, l_{c_{nlist}}\}$, target number of visited lists *nprobe*, vector similarity function f_v, query vector $\mathbf{v}_q$, target result size k
Output: approximate query result $\tilde{\mathcal{R}}$

1 $\tilde{\mathcal{R}} \leftarrow \{\}$; // Result set
2 $\mathcal{C}' \leftarrow \{$*nprobe* centroids closest to $\mathbf{v}_q$ in $\mathcal{C}\}$;
3 **foreach** centroid $\mathbf{v}_c \in \mathcal{C}'$ **do**
4 $l_c \leftarrow$ inverted list of $\mathbf{v}_c$ from $\mathcal{L}$;
5 **foreach** vector $\mathbf{v} \in l_c$ **do**
6 $\mathbf{v}_f \leftarrow \arg\max_{\mathbf{v}' \in \tilde{\mathcal{R}}} f_v(\mathbf{v}', \mathbf{v}_q)$;
7 **if** $|\tilde{\mathcal{R}}| < k$ **or** $f_v(\mathbf{v}, \mathbf{v}_q) < f_v(\mathbf{v}_f, \mathbf{v}_q)$ **then**
8 $\tilde{\mathcal{R}} \leftarrow \tilde{\mathcal{R}} \cup \{\mathbf{v}\}$;
9 **if** $|\tilde{\mathcal{R}}| > k$ **then**
10 $\mathbf{v}_f \leftarrow \arg\max_{\mathbf{v}' \in \tilde{\mathcal{R}}} f_v(\mathbf{v}', \mathbf{v}_q)$;
11 $\tilde{\mathcal{R}} \leftarrow \tilde{\mathcal{R}} \setminus \{\mathbf{v}_f\}$;
12 **return** $\tilde{\mathcal{R}}$;

Algorithm 2: Search on the Graph Index

Input: graph $G(\mathcal{V}, \mathcal{E})$, entry vector $\mathbf{v}_e$, vector similarity function f_v, query vector $\mathbf{v}_q$, target result size k
Output: approximate query result $\tilde{\mathcal{R}}$

1 $\tilde{\mathcal{R}} \leftarrow \{\mathbf{v}_e\}$; // Result set
2 $\mathcal{C} \leftarrow \{\mathbf{v}_e\}$; // Candidate set
3 **while** $|\mathcal{C}| > 0$ **do**
4 $\mathbf{v}_c \leftarrow \arg\min_{\mathbf{v}' \in \mathcal{C}} f_v(\mathbf{v}', \mathbf{v}_q)$;
5 $\mathcal{C} \leftarrow \mathcal{C} \setminus \{\mathbf{v}_c\}$;
6 **foreach** $\mathbf{v} \in$ *neighborhood*$(\mathbf{v}_c)$ **do**
7 $\mathbf{v}_f \leftarrow \arg\max_{\mathbf{v}' \in \tilde{\mathcal{R}}} f_v(\mathbf{v}', \mathbf{v}_q)$;
8 **if** $|\tilde{\mathcal{R}}| < k$ **or** $f_v(\mathbf{v}, \mathbf{v}_q) < f_v(\mathbf{v}_f, \mathbf{v}_q)$ **then**
9 $\mathcal{C} \leftarrow \mathcal{C} \cup \{\mathbf{v}\}$;
10 $\tilde{\mathcal{R}} \leftarrow \tilde{\mathcal{R}} \cup \{\mathbf{v}\}$;
11 **if** $|\tilde{\mathcal{R}}| > k$ **then**
12 $\mathbf{v}_f \leftarrow \arg\max_{\mathbf{v}' \in \tilde{\mathcal{R}}} f_v(\mathbf{v}', \mathbf{v}_q)$;
13 $\tilde{\mathcal{R}} \leftarrow \tilde{\mathcal{R}} \setminus \{\mathbf{v}_f\}$;
14 **return** $\tilde{\mathcal{R}}$;

IVF index. The inverted file (IVF) index is a form of quantization-based indices. It first trains a quantizer, typically using k-means clustering [75], to partition the vector space $\mathbb{R}^d$ into *nlist* clusters represented by their centroids, denoted $\mathcal{C} = \{\mathbf{v}_{c_1}, \mathbf{v}_{c_2}, \ldots, \mathbf{v}_{c_{nlist}}\}$, where each centroid $\mathbf{v}_{c_i}$ is a value in $\mathbb{R}^d$. Each vector in the dataset is then assigned to its nearest centroid, and an inverted list is created for each cluster to store references to the vectors within it. Together, the centroids $\mathcal{C}$ and their corresponding inverted lists $\mathcal{L}$ form the IVF index. For more details, please refer to [26].

During the search (algorithm 1), the traversal is performed within the *nprobe* clusters whose centroids are closest to the query vector (line 2), and all the vectors

within these selected clusters are scanned (lines 3- 5) to find the k nearest neighbors of the query vector $\mathbf{v}_q$ (lines 6- 11). Optionally, compression techniques such as product quantization [49] or additive quantization [19, 69, 70] can be applied to vectors to save memory and accelerate the search process.

Graph Index. In a graph index, each vector is represented as a vertex, with edges linking vertices whose vectors are similar. Together, the vertex set $\mathcal{V}$ and the edge set $\mathcal{E}$ form the graph index $G(\mathcal{V}, \mathcal{E})$. The index construction typically follows one of the three strategies: incremental construction, refinement, or divide-and-conquer. For more details, please refer to [92].

During search (algorithm 2), the traversal follows a greedy routing strategy, beginning with an entry vector $\mathbf{v}_e$ and gradually expanding the neighbors towards the query vector $\mathbf{v}_q$. This process involves repeated steps of *neighborhood expansion* (lines 4- 6) and *result update* (lines 7- 13). Optionally, several optimizations can be applied to enhance the search process, such as maintaining a visited set [29, 30, 67, 68, 86, 101] to avoid redundant calculations, expanding the result set to include more than k vectors and retaining only the top-k upon returning [29, 30, 67, 68, 86, 101] to improve the accuracy, and utilizing hierarchical graph structures [68, 101] to reduce search time.

3 Review of FANNS Algorithms

In this section, we propose a pruning-focused framework (Figure 1) to classify 17 existing FANNS algorithms (**A1**–**A17**), demonstrating how each algorithm aligns with one of the four distinct pruning strategies defined in our framework and revealing the interrelationships among these algorithms (Figure 2). The core design of each algorithm is summarized using the terminology given in section 2.

3.1 The Pruning-Focused Framework

Existing framework classifies FANNS algorithms based on *when* the scalar filter is applied, namely, pre-filtering, post-filtering, and in-filtering [28, 33, 35, 73, 78, 99]. However, as discussed in **C2** of subsection 1.1, this classification is not enough to cover all algorithms, and too coarse to distinguish between algorithms.

In contrast, our framework focuses on the pruning behaviors of indices for hybrid queries. Since a hybrid query involves both vectors and scalars, it supports two types of pruning: *vector pruning* and *scalar pruning*. *Vector pruning* refers to avoiding the computation of vector similarities for data points whose vectors are far from the query vector, typically achieved by searching on a vector index. *Scalar pruning* refers to skipping vector similarity calculations for data points that do not satisfy the scalar filter, which can be efficiently implemented, as an example, using a bitmap generated by a scalar index to identify such points.

By emphasizing the dominant pruning behavior, our pruning-focused framework identifies four distinct strategies: *vector-solely pruning* (VSP), *vector-centric joint pruning* (VJP), *scalar-centric joint pruning* (SJP), and *scalar-solely pruning* (SSP). This finer-grained classification framework captures the core design principles of FANNS algorithms and overcomes the limitations of the existing framework by offering a more robust and extensible classification for both current and future FANNS algorithms, as detailed in the following subsections.

3.2 VSP-based FANNS algorithms

Vector-solely pruning (VSP) performs only vector pruning without any scalar pruning. The underlying rationale of VSP-based FANNS algorithms [58, 91, 96, 100, 102] (**A1**, **A2**) is that FANNS can be viewed as a direct extension of ANNS. These algorithms typically adapt existing vector indices with minimal modifications. VSP-based FANNS algorithms are suitable for scenarios with unselective scalar filters. However, due to the lack of scalar pruning, they may degrade to computing similarities for nearly all vectors when the scalar filter is highly selective.

A1: Post-Filtering Algorithm Family. Post-Filtering Algorithm Family represents a class of methods [58, 91, 96, 100] that directly use the top-k interface of a vector index designed for ANNS to retrieve the K' nearest neighbors (K'-*NN*), and then apply the scalar filter to find the final K nearest neighbors (*filtered* K-*NN*). For the choice of vector index, any type can be used, including IVF indices [58, 91, 96, 100] that require minimal memory footprint, or graph indices [91, 100] that offer higher efficiency and accuracy. For the selection of K', some methods choose a K' much larger than K, hoping to retain at least K elements after scalar filtering [91, 96]; while others start with a K' slightly larger than K and iteratively increase it until at least K data points are retained [100]. Overall, Post-Filtering Algorithm Family is highly flexible and easy to implement since it can use any vector index off-the-shelf, but it faces the challenge of estimating the optimal K' due to the unpredictable selectivity of the scalar filter during a specific search.

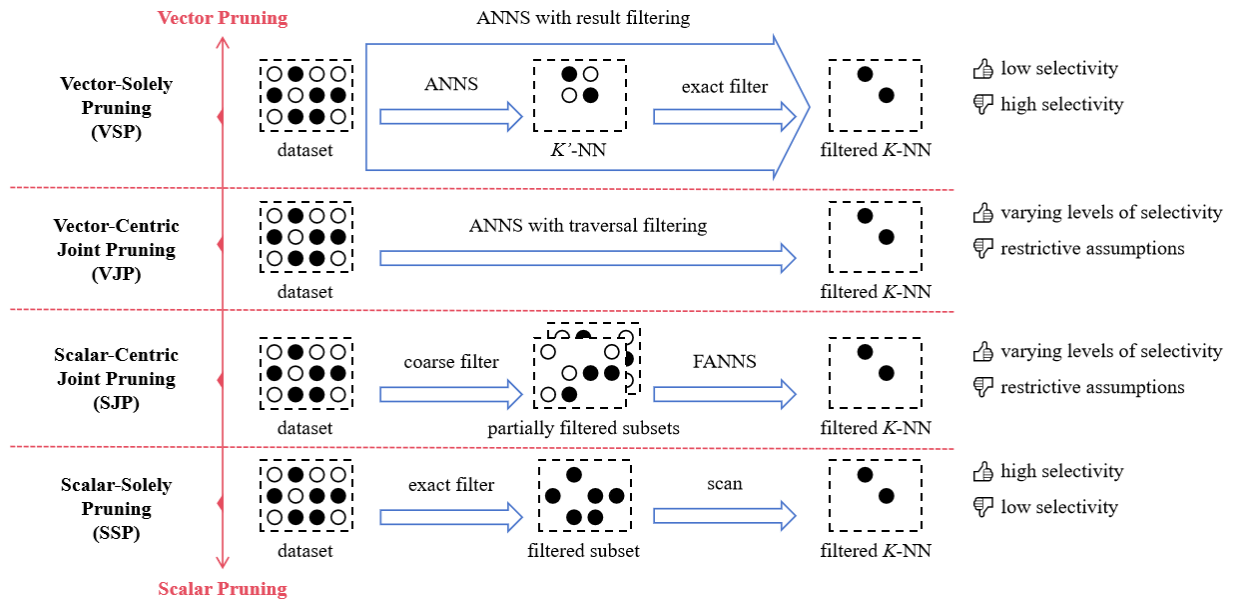

Fig. 1 The proposed pruning-focused framework for classifying FANNS algorithms

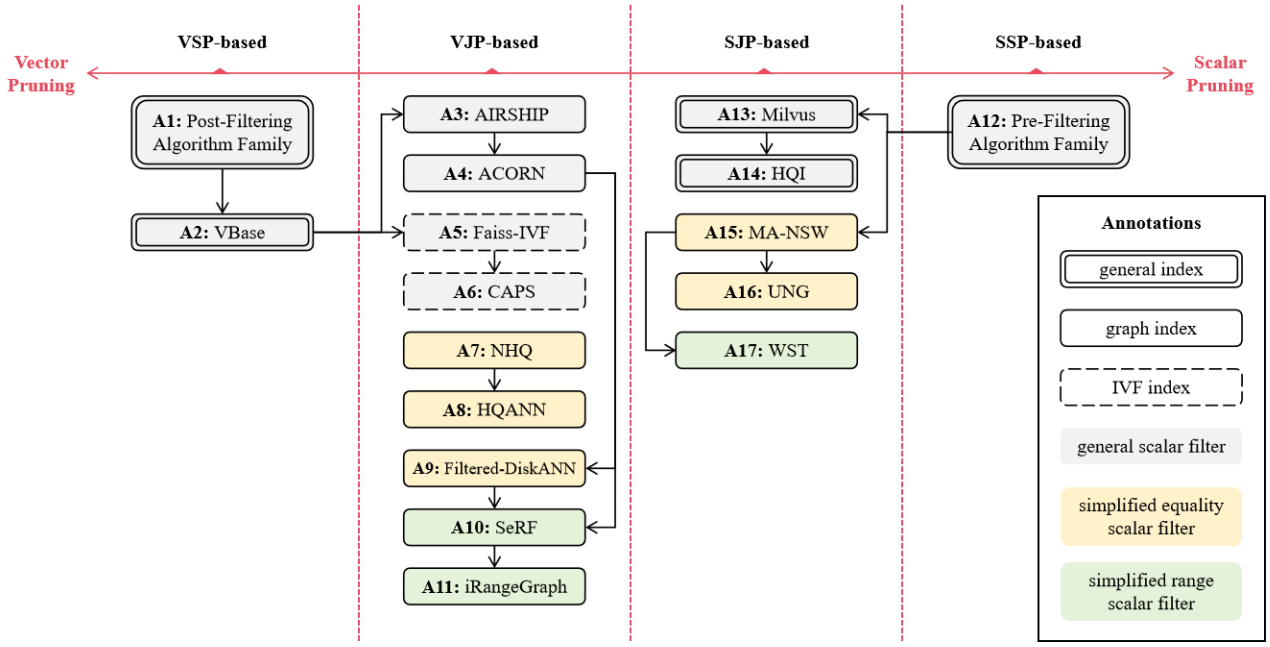

Fig. 2 Classification of FANNS algorithms under the pruning-focused framework and interrelationships among them.

A2: VBase. VBase [102] optimizes the Post-Filtering Algorithm Family (**A1**) by dynamically selecting the optimal K'. Specifically, it modifies the result update process during the search over the ANNS index (subsection 2.4) by adding only data points that satisfy the scalar filter to the result set. Besides, it introduces an improved termination check that requires the result set contain at least K data points and meet the "relaxed monotonicity" condition [102], which ensures the result vectors are sufficiently similar to the query vector. With these modifications, the number of traversed data points during search serves as K' in the Post-Filtering Algorithm Family, thereby addressing the difficulty in estimating K', and this dynamically selected K' has been proven to be optimal [102].

3.3 VJP-based FANNS algorithms

Vector-centric joint pruning (VJP) incorporates scalar pruning into the vector-pruning-centric search process. VJP-based FANNS algorithms [26, 33, 35, 58, 78, 93,

97, 99, 105, 107] (**A3**–**A11**) further modify the search process over the ANNS index, so that the search direction is primarily guided by the similarity to the query vector, with adjustments made by the scalar filter, which may significantly reduce the number of vector similarity computations. However, the effectiveness of the scalar filter in assisting the guidance of search direction is not always guaranteed. Therefore, some studies go beyond modifying the search process by incorporating scalar information into the construction of indices [33, 93, 97, 99, 107] (**A7**–**A11**). Overall, VJP-based FANNS algorithms tend to be more efficient than VSP-based FANNS algorithms (subsection 3.2) across varying levels of selectivity, but their results may be less reliable [78, 93, 97, 105] (**A3**, **A4**, **A7**, **A8**), and their applicability may be limited by their restrictive assumptions [33, 93, 97, 99, 107] (**A7**–**A11**).

A3: AIRSHIP and A4: ACORN. Attribute-Constrained Similarity Search on Proximity Graph (AIRSHIP) [105] and ANN Constraint-Optimized Retrieval Network (ACORN) [78] both incorporate scalar pruning into the search process over the graph index (subsection 2.4). In AIRSHIP, data points that satisfy and do not satisfy the scalar filter are probabilistically visited during neighbor expansion, allowing exploiting satisfied data points to enhance efficiency, while exploring unsatisfied yet potentially useful data points for comprehensiveness. When updating the result set, it only adds data points that satisfy the scalar filter. In ACORN, only data points that satisfy the scalar filter are visited during neighbor expansion, aiming for thorough scalar pruning by traversing the *predicate subgraph* [78]. While many graph indices [92] use a Relative Neighborhood Graph (RNG) [89] approximation on Delaunay Graph (DG) [7] for sparsity, it is not suitable for ACORN since a sparse graph often leads to a disconnected *predicate subgraph* [78]. Thus, ACORN retains the DG for a dense graph and designs a compression strategy to save memory. Overall, both algorithms are more efficient than VBase (**A2**) with graph indices, but uncertainty around the connectivity of the traversed subgraph makes search results potentially unreliable.

A5: Faiss-IVF and A6: CAPS. Faiss-IVF [26, 58] and Constrained Approximate Partitioned Search (CAPS) [35] both incorporate scalar pruning into the search process over the IVF index (subsection 2.4). In Faiss-IVF, similarity calculations for vectors that do not satisfy the scalar filter are skipped during scanning. In CAPS, an *attribute frequency tree* (AFT) [35] is built for each cluster during index construction, with each AFT recursively partitioning the cluster based on its most frequently occurring scalar values. During search, the AFT narrows the scan scope within each cluster, then a scan similar to Faiss-IVF is performed in this refined scope. Overall, both algorithms are more efficient than VBase (**A2**) with IVF indices, and their memory footprint is smaller than that of graph indices, albeit with potentially lower search speed and accuracy.

A7: NHQ and A8: HQANN. Native Hybrid Query (NHQ) [93] builds a composite graph index to enable joint pruning. It assumes discrete scalar values, and a *simplified equality scalar filter* (subsection 2.2). For each data point, it transforms its scalar-tuple into a *scalar vector* by encoding scalar values as numeric values, and then combines its vector with this *scalar vector* to form a *fusion vector*. Accordingly, it defines a *fusion distance* metric to measure the similarity between *fusion vectors*, such that data points with similar scalars and vectors have similar *fusion vectors*. Consequently, FANNS over data points is transformed into ANNS over *fusion vectors*, allowing both index construction and search accomplished using *fusion vectors* and *fusion distance*. Hybrid Query Approximate Nearest Neighbor Search (HQANN) [97] follows the core idea of NHQ, but introduces a different definition of the *fusion distance*. Overall, both algorithms achieve high search efficiency by converting FANNS into ANNS under its assumptions, but uncertainty around the optimal form of *fusion distance* makes search results potentially unreliable.

A9: Filtered-DiskANN. Filtered-DiskANN [33] includes two similar methods, StitchedVamana and FilteredVamana, both incorporating scalar information into the construction and search process of the Vamana [86] graph index. It assumes discrete scalar values, and a *simplified equality scalar filter* (subsection 2.2) whose conjunctive part in Equation 1 only have one *active sub-filter*. StitchedVamana constructs a separate graph for each scalar value over the subset of data points with that value in their scalar-tuple, then overlays these *scalar-specific subgraphs* by unioning their edges and selectively retaining a limited number of edges to save memory. For a search using a scalar filter, it performs searches using each sub-filter, and then merges their results. For a search using a sub-filter, it only visits data points that satisfy the scalar filter during neighbor expansion, effectively performing ANNS on the *scalar-specific subgraph*. FilteredVamana has a similar search process to StitchedVamana but constructs an approximate index by incrementally adding data points to an initially empty graph. For each newly added data point, it performs searches using each scalar value in the scalar-tuple of that data point as scalar filter, and the union of these results forms the candidate neighbors for

connecting the new point. Overall, Filtered-DiskANN efficiently searches within subgraphs that meet the scalar filter like ACORN (**A4**), but achieves higher accuracy by explicitly constructing these subgraphs under its assumptions.

A10: SeRF. Segment Graph for Range-Filtering (SeRF) [107] follows the core idea of Filtered-DiskANN (**A9**) but operates under different assumptions. It assumes that each data point has only one scalar whose value is drawn from a discrete and orderable set, and a *simplified range scalar filter* (subsection 2.2). For index construction, similar to Filtered-DiskANN which builds a graph by approximately overlaying *scalar-specific subgraphs* for all possible scalar values, SeRF constructs a graph by approximately overlaying *range-specific subgraphs* for all possible scalar value ranges (in the form of $[a, b], a \leq b$). Assuming there are n distinct scalar values (the same as the number of data points), Filtered-DiskANN overlays n subgraphs and achieves a worst-case space complexity of $O(Mn)$ by limiting each point to M neighbors [33], while SeRF overlays n^2 subgraphs and has a worst-case space complexity of $O(Mn^2)$ due to its "lossless compression" [107]. For FANNS, similar to Filtered-DiskANN which effectively performs searches on *scalar-specific subgraphs*, SeRF effectively performs searches on *range-specific subgraphs*. Overall, SeRF also provides higher efficiency and accuracy than ACORN (**A4**) under its assumptions, but it suffers from a high memory footprint.

A11: iRangeGraph. Improvising Range-dedicated Graphs (iRangeGraph) [99] follows the same assumptions and search strategy as SeRF (**A10**), but adopts a different approach for index construction. Rather than overlaying numerous *range-specific subgraphs*, iRangeGraph constructs a moderate number of *range-specific subgraphs* and organizes them in a *segment tree* structure [99]. In this *segment tree*, each node represents a range and stores the range-specific subgraph constructed over the subset of data points with scalar values in that range, which results in a worst-case space complexity of $O(Mn \log n)$. During search, as each data point appears in *range-specific subgraphs* at multiple levels of the tree, its neighbors for expansion are the union of its neighbors across all the tree levels. Overall, iRangeGraph achieves a smaller memory footprint than SeRF (**A10**) while ensuring search efficiency and accuracy under its assumptions.

3.4 SSP-based FANNS algorithms

Scalar-Solely Pruning (SSP) performs only scalar pruning without any vector pruning. SSP-based FANNS algorithms [78, 91, 96, 102] (**A12**) are easy to implement and memory efficient without the need for vector indices, and are suitable for scenarios with selective scalar filters. However, due to the lack of vector pruning, they may degrade to computing similarities for nearly all vectors when the scalar filter is unselective.

A12: Pre-Filtering Algorithm Family. Pre-Filtering Algorithm Family represents a class of methods [78, 91, 96, 102] that first apply the scalar filter to retrieve a subset of data points (*filtered subset*), and then find the *filtered K-NN* by calculating similarities for all vectors in this subset through brute-force scan. Since the number of possible *filtered subsets* can be extremely large or even uncountable (e.g., when the scalar filter is based on regular expressions), it is impractical to construct vector indices for all possible *filtered subsets* to accelerate the search process, making vector pruning infeasible. However, the efficiency of brute-force scan can be improved. For example, AnalyticDB-V (ADBV) [96] trades accuracy for efficiency by pre-compressing data vectors using Voronoi graph product quantization and performing asymmetric distance computation, in which query vectors remain uncompressed while data vectors are compressed.

3.5 SJP-based FANNS algorithms

Scalar-Centric Joint Pruning (SJP) incorporates vector pruning into the scalar-pruning-centric search process. To achieve this, SJP-based FANNS algorithms [15, 28, 73, 91, 98] (**A13**–**A17**) define rules to select a limited number of *filtered subsets*, and then build vector indices for each selected subset. During search, they first use part of the scalar filter to coarsely retrieve some of these preselected subsets (*partially filtered subsets*), and then carry out hybrid searches on them to obtain the final *filtered K-NN*. Notably, the *partially filtered subsets* may contain data points that do not satisfy the scalar filter, so scalar pruning is not as complete as in SSP-based FANNS algorithms (subsection 3.4), but since vector indices are built on these subsets, vector pruning can be leveraged to accelerate the search process. Overall, SJP-based FANNS algorithms require careful selection of subsets to build vector indices, and their applicability is all limited by their restrictive assumptions.

A13: Milvus-Partition and A14: HQI. Milvus-Partition [91] and Hybrid Query Index (HQI) [73] both parti-

tion the dataset into disjoint subsets based on workload characteristics and build vector indices for each subset. During search, they both retrieve *partially filtered subsets* and apply one of the VSP-, VJP-, or SSP-based FANNS algorithms for each subset, and finally merge the results to obtain the final *filtered K-NN*. However, the criteria and structure of dataset partitioning differ between them. In Milvus-Partition, partitioning is scalar-based and single-layered. It first identifies the most frequently used scalar from prior workload, and then evenly divides the dataset into subsets based on the values of this scalar. In HQI, partitioning is filter-based and multi-layered. It first identifies several frequently used scalar filters from prior workload, and then constructs an *extended qd-tree* [73] by iteratively partitioning the dataset according to these filters. Overall, both algorithms achieve high search efficiency when workload characteristics are stable, but their applicability is limited by the need for the prior workload information to guide the partitioning process.

A15: MA-NSW. Multiattribute ANNS based on Navigable Small World (MA-NSW) [98] constructs multiple NSW [67] graph indices based on the values of the scalar-tuples. It assumes discrete scalar values, and a *simplified equality scalar filter* (subsection 2.2). For index construction, MA-NSW first defines a containment relationship between two scalar-tuples $\mathbf{s}_1$ and $\mathbf{s}_2$, where $\mathbf{s}_1$ is included by $\mathbf{s}_2$ ($\mathbf{s}_1 \subseteq \mathbf{s}_2$) if $\mathbf{s}_2$ has at least one scalar with a NULL value and all other scalar values are identical to those of $\mathbf{s}_1$. Assuming the scalar schema has m scalars, each with m_i distinct values including the NULL value, there will be $\Pi_{i=1}^{m} m_i$ possible distinct scalar-tuples. For each observed scalar-tuple, MA-NSW identifies a subset of data points whose scalar-tuples are either identical to or included by the given scalar-tuple, and then constructs an NSW on this subset. Since a single NSW has a space complexity of $O(Mn)$, the worst-case space complexity of MA-NSW is $O(Mn^{m+1})$. For a search using a scalar filter, where each conjunctive sub-filter corresponds to a subset with a prebuilt NSW, MA-NSW retrieves all relevant subsets, performs ANNS on each NSW, and finally merges the results. Overall, despite the high efficiency of MA-NSW under its assumptions, its memory footprint can be prohibitively large.

A16: UNG. Unified Navigating Graph (UNG) [15] constructs a single graph index based on the values of the scalar-tuples. It has the same assumptions and containment relationship definition as in MA-NSW (**A15**). Additionally, it defines a minimal containment relationship between two scalars $\mathbf{s}_1$ and $\mathbf{s}_2$, where $\mathbf{s}_1$ is minimally included by $\mathbf{s}_2$ if $\mathbf{s}_1 \subseteq \mathbf{s}_2$ and no other scalar $\mathbf{s}_3$ exists such that $\mathbf{s}_1 \subseteq \mathbf{s}_3 \subseteq \mathbf{s}_2$. During index construction, for each scalar-tuple, UNG identifies the subset of data points whose scalar-tuples are identical to the given scalar-tuple, and then constructs a graph index on this subset. After that, if the scalar-tuple of one subset is minimally included by that of another, UNG selectively adds directed edges from some data points in the latter subset to some in the former subset, organizing these subsets in a manner similar to a *prefix tree* structure [15]. Since all the subsets are disjoint from each other and the number of added edges is limited, the worst-case space complexity of UNG is $O(Mn)$. For a search using a scalar filter, where each conjunctive sub-filter corresponds to a group of linked subsets with a *concatenated graph index* by combining graph indices of these subsets with directed edges connecting these subsets, UNG retrieves all relevant groups, performs ANNS on each *concatenated graph index*, and finally merges the results. Overall, UNG achieves a small memory footprint and high efficiency when its assumptions are satisfied and containment relationships are abundant.

A17: WST. Window Search Tree (WST) [28] shares the same assumptions and a similar index structure with iRangeGraph (**A11**), but introduces four different search methods: VamanaWST, OptimizedPostfiltering, ThreeSplit, and SuperPostfiltering. WST organizes *range-specific subgraphs* in an extended form of *segment tree* [28], where each node represents a range and its children recursively divide this range into β sub-ranges, which reduces to the standard *segment tree* when $\beta = 2$. VamanaWST recursively searches the WST from top to bottom to identify a set of nodes whose ranges are disjoint and the union of their ranges is the query range (i.e., the scalar filter), then performs ANNS on each node, and finally merges the results. OptimizedPostfiltering selects a single node with the smallest range that contains the query range and applies one of the VSP-, VJP-, or SSP-based algorithms on this node to obtain the result. ThreeSplit first selects a node with the largest range contained within the query range and performs ANNS on this node, then applies OptimizedPostfiltering to the remaining portions of the query range, and finally merges the results. SuperPostfiltering is similar to OptimizedPostfiltering but does not rely on *range-specific subgraphs* in the WST; instead, it constructs an arbitrary set of *range-specific subgraphs* to achieve an expected "small blowup" [28]. Overall, all four methods achieve small memory footprint and high search efficiency under given assumptions.

4 Review of Hybrid Datasets

In this section, we discuss the construction strategies of existing hybrid datasets, and present several examples to detail their contents, as summarized in Table 2.

4.1 Hybrid Dataset Construction

Hybrid datasets can be seen as vector datasets with added scalars. However, unlike the evaluation of ANNS, which benefits from a standardized set of vector datasets [8, 49, 50, 80] provided by standard benchmarks like ANN-Benchmarks [6], the evaluation of FANNS lacks a comparable standard for hybrid datasets. As a result, existing FANNS studies often customize their own hybrid datasets.

Some studies synthesize scalars such as generating random values from a uniform distribution [15, 28, 33, 35, 73, 78, 91, 93, 96–98, 100, 105, 107], while others use "organic" scalars that already exists in the original data sources [28, 33, 78, 96, 99, 102, 105, 107]. In the former case, the vectors can be sourced from any vector dataset, including those in the ANN-Benchmarks. In the latter case, the original data source is typically collected by crawling web pages, where unstructured data (e.g., images, texts, or audio) are used to extract the feature vectors through a pre-trained model (e.g., CNN [37, 55, 85], Transformer [13, 25, 90], or VGGish [2]), and structured data (e.g., publication dates, keywords, and likes) serve as the corresponding scalars.

Based on the above analysis, we categorize existing hybrid datasets into two types: (1) *synthesized hybrid datasets*, which contain only synthesized scalars; and (2) *organic hybrid datasets*, which contain organic scalars, either exclusively or in combination with synthesized ones.

4.2 Representative Hybrid Datasets

Among existing hybrid datasets, we select nine representative examples (**D1**-**D9**) for detailed description, including three *synthesized hybrid datasets* (**D1**-**D3**) whose vectors are sourced from ANN-Benchmarks, and six *organic hybrid datasets* (**D4**-**D9**) with traceable raw data sources. A concise summary of these datasets is provided in Table 2, including information on data size, vector source and dimensionality, number of organic scalars, and their usage in recent studies.

D1: SIFT-1M, D2: GIST-1M, and D3: Deep-10M. They are widely-used vector datasets from ANN-Benchmarks. SIFT-1M [3] [49] contains 1 million 128-dimensional vectors, extracted from the INRIA Holidays images [48] using local SIFT descriptors [63]. GIST-1M [3] [49] contains 1 million 960-dimensional vectors, extracted from the INRIA Holidays images [48] using global GIST descriptors [76]. Deep-10M [4] [8] contains 10 million 96-dimensional vectors, extracted from the images on the Web using the GoogLeNet model [87]. Original SIFT, GIST and Deep do not have organic scalars, so synthesized scalars are generated to make them hybrid datasets. Notably, SIFT and Deep also have 1 billion versions, namely SIFT-1B [3] and DEEP-1B [4].

D4: MNIST-8M. MNIST-8M [5] [62] contains 8,100,000 data points, each comprising a 784-dimensional vector representation of a handwritten digit image generated using SVMs [22], along with an integer label between 0 and 9. These images are derived from the infinite MNIST dataset (InfiMNIST) [6], which extends the original MNIST dataset (MNIST) [24] by dynamically generating new samples through careful elastic deformations, theoretically enabling the creation of an unlimited number of images. Notably, from InfiMNIST, 10,000 samples ranging from 0 to 9,999 form the MNIST testing set, 60,000 samples ranging from 10,000 to 69,999 form the MNIST training set, and 8,100,000 samples ranging from 10,000 to 8,109,999 form the MNIST-8M.

D5: MTG. MTG [7] contains 40,274 data points, each comprising a 1,152-dimensional vector representation of a "Magic: The Gathering" [8] card image generated using the OpenCLIP model [20], along with 21 scalars, such as "artist", "rarity", and "toughness". Notably, the content of "image" scalar is the raw card image, and several scalars ending with "uri" provide traceable links to the websites from which the dataset was crawled.

D6: GloVe-Twitter and D7: Glove-Crawl. GloVe-Twitter [9] and GloVe-Crawl [9] both contain word vectors generated using the GloVe algorithm [80] from the Twitter corpus and the Common Crawl corpus, respectively.

[2] https://github.com/tensorflow/models/tree/master/research/audioset/vggish

[3] http://corpus-texmex.irisa.fr/

[4] https://research.yandex.com/blog/benchmarks-for-billion-scale-similarity-search

[5] https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multiclass.html#mnist8m

[6] https://leon.bottou.org/projects/infimnist

[7] https://huggingface.co/datasets/TrevorJS/mtg-scryfall-cropped-art-embeddings-open-clip-ViT-SO400M-14-SigLIP-384

[8] https://scryfall.com/

[9] https://nlp.stanford.edu/projects/glove/

Table 2 Summary of Selected Hybrid Datasets

Dataset	#Points	Vector		Scalar-Tuple	Used in
		Source	#Dimensions	#Organic	
D1: SIFT-1M	1,000,000	Image	128	0	[15, 28, 33, 35, 73, 78, 91, 93, 96–98, 100, 105]
D2: GIST-1M	1,000,000	Image	960	0	[15, 28, 73, 93, 97, 98, 100]
D3: Deep-10M	10,000,000	Image	96	0	[28, 91, 96, 97, 107]
D4: MNIST-8M	8,100,000	Image	784	1	[98, 105]
D5: MTG	40,274	Image	1,152	21	[15]
D6: GloVe-Twitter	1,183,514	Word	25; 50; 100; 200	1	[15, 28, 35, 93, 97, 98]
D7: GloVe-Crawl	1,989,995	Word	300	1	[15, 35, 93]
D8: LAION-1M	1,000,448	Image	512	15	[99, 107]
		Text	512		
D9: YouTube	6,134,598	Video	1,024	3	[15, 78]
		Audio	128		

GloVe-Twitter contains 1,183,514 unique words, with corresponding word vectors generated in four different dimensions: 25, 50, 100, and 200. This means that GloVe-Twitter provides four sets of word vectors, each containing 1,183,514 vectors for the respective dimensions. In contrast, GloVe-Crawl contains 1,989,995 unique words, with word vectors generated in a single fixed dimension of 300.

D8: LAION-1M. LAION-1M [10] contains the first 1,000,448 data points from LAION-400M [83], with each data point comprising 2 vectors and 15 scalars. Each vector pair includes a 512-dimensional image embedding and a corresponding 512-dimensional text embedding, both generated from the Common Crawl corpus using the same CLIP model [81]. The scalar part includes the image URL and associated metadata, such as the image's width and height. Notably, LAION-5B [84] is also available, containing a significantly larger dataset of 5,526,641,167 data points, while maintaining a structure similar to the previous versions.

D9: YouTube. YouTube [11] [84], an updated version of YouTube-8M [2], contains 6,134,598 data points, each comprising 2 vectors and 3 scalars. Each vector pair includes a 1,024-dimensional video embedding and a corresponding 128-dimensional audio embedding, generated from the YouTube corpus using Inception model [46] and VGGish model [12], respectively. The scalars include the sample ID, video URL and video labels.

[10] `https://deploy.laion.ai/`

[11] `https://research.google.com/youtube8m/download.html`

[12] `https://github.com/tensorflow/models/blob/master/research/audioset/vggish/README.md`

5 The Distribution Factor for Query Difficulty

Understanding query difficulty is essential for analyzing algorithm performance and designing evaluation benchmarks. Query difficulty influences both the efficiency and accuracy of an algorithm. Specifically, more difficult queries tend to increase search time and decrease *recall*. Multiple factors may contribute to query difficulty. Identifying these factors not only helps explain performance fluctuations across different queries from various perspectives, but also enables the construction of evaluation benchmarks across a wider range of difficulty levels through their combination.

This section goes beyond the *selectivity* factor to explore the role of the *distribution* factor in the difficulty of hybrid queries. We begin by motivating the incorporation of the *distribution* factor by examining real-world hybrid datasets. Next, we conduct carefully designed experiments to assess the impact of the *distribution* factor on query difficulty, followed by both qualitative visualizations and quantitative measurements to explain the results. Finally, we propose a schema towards more comprehensive FANNS algorithm evaluation by incorporating both *selectivity* and *distribution* factors.

5.1 Incorporation of Distribution Factor: Motivation

The *selectivity* measures the proportion of data points excluded by the scalar filter (subsection 2.3). It is currently the only factor used to evaluate the difficulty of hybrid queries. As discussed in section 3, hybrid queries with high *selectivity* are more difficult for VSP-based FANNS algorithms, whereas those with low *selectivity* are more difficult for SSP-based algorithms. In the meantime, VJP-based and SJP-based algorithms are

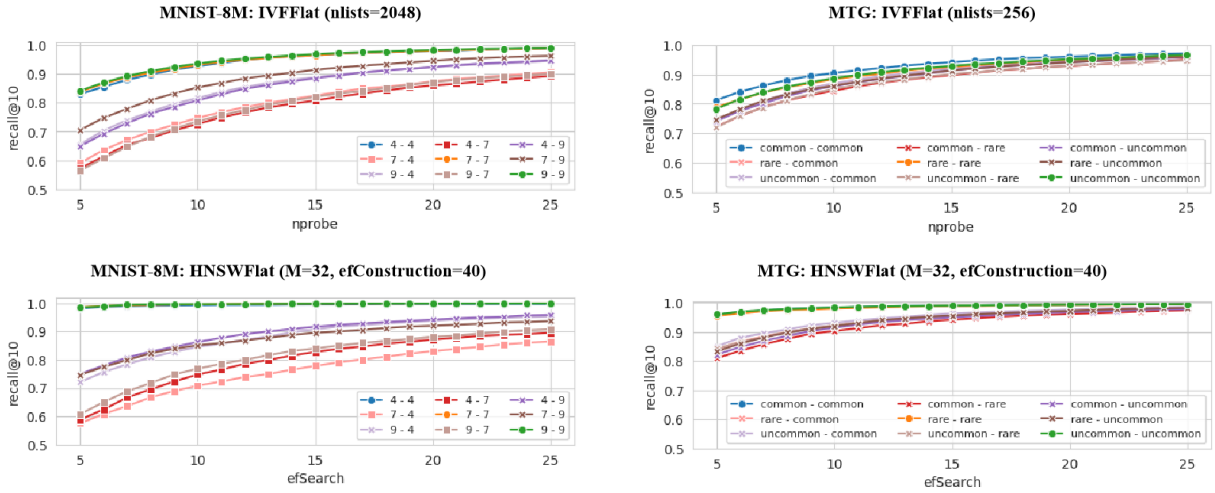

Fig. 3 Performance evaluations on hybrid queries with oracle partition indices. Each "x-y" is a set of hybrid queries, where the scalar value of base vectors is x and the scalar value of query vectors is y. Each set of base vectors is indexed using IVF and HNSW. The distribution relationship of each hybrid query set is ID, POD, or OOD, represented by circle, cross, or square.

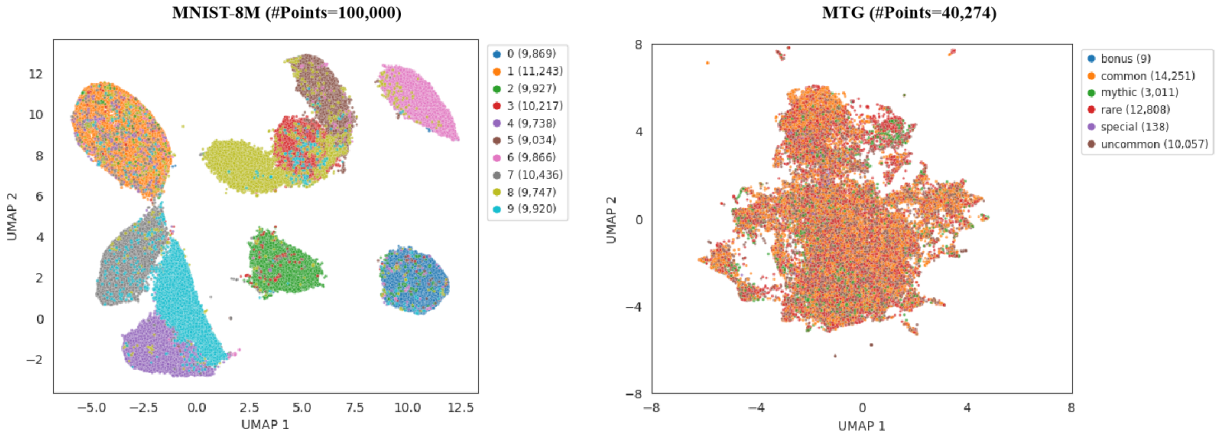

Fig. 4 UMAP visualizations of MNIST-8M (Left) and MTG (Right). Each *filtered subset* has a 2-dimensional distribution. In MNIST-8M, the distributions of each *filtered subset* are well-separated, exhibiting clustering behavior, while in MTG, the distributions of all *filtered subsets* are highly overlapping, sharing a similar overall distribution.

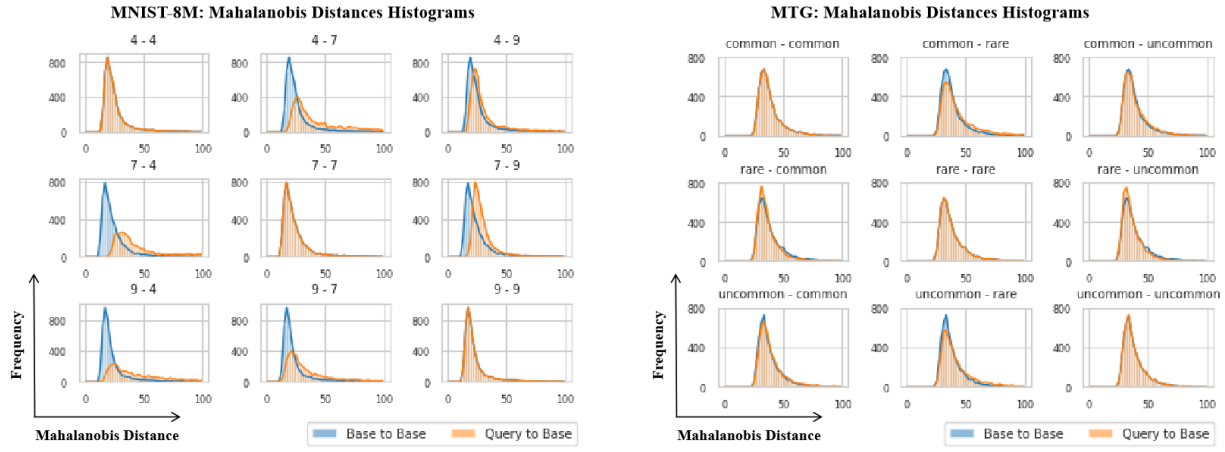

Fig. 5 Mahalanobis Distance Histograms of MNIST-8M (Left) and MTG (Right). Each subgraph titled "x-y" shows the histograms of both "x-y" (in orange) and "x-x" (in blue), illustrating the distribution shift between query vectors and base vectors when sampled from different *filtered subsets*, compared to when they are sampled from the same *filtered subset*.

robust across varying levels of *selectivity* if their corresponding assumptions are satisfied.

In real-world scenarios, hybrid datasets often exhibit clustering behavior, where data points with similar scalars tend to form distinct clusters in the vector space. For instance, in e-commerce platforms, products from the same brand or sharing a common style often exhibit clustered embeddings, reflecting their inherent similarity [71]. In the domain of news articles, temporal proximity can lead to clustering, as articles covering the same event within a short timeframe tend to have highly similar content and embeddings [82].

This motivates us to incorporate the *distribution* factor into the hybrid query difficulty. In the high dimensional vector space, each group of vectors has a high dimensional distribution. The *distribution* factor refers to the relationship between distributions of two sets of vectors. If query vectors and base vectors are sampled from two different *filtered subsets* within a clustered dataset, the relationship between their distributions is likely to be Out-of-Distribution (OOD) [18, 47], leading to difficult hybrid queries, because query vectors are far from their nearest neighbors in the base vectors, and the nearest neighbors are also distant from each another [18]. Conversely, if query vectors and base vectors are sampled from the same *filtered subset* within a clustered dataset, or sampled from a dataset without clustering behavior, their distribution relationship is likely to be In-Distribution (ID) [18, 47], resulting in easier hybrid queries.

5.2 Impact of Distribution Factor: Experiments and Explanations

To illustrate how the *distribution* factor impacts the difficulty of hybrid queries, we conduct experiments on two hybrid datasets introduced in section 4: MNIST-8M, which exhibits clustering behavior, and MTG, which does not. For demonstration purposes, we consider only one vector column and one scalar column in each dataset, with the scalar filter checking whether the scalar equals a specific value. Concretely, for MNIST-8M, we use the first 1,000,000 data points and select `digit` as the scalar attribute. For MTG, we use all 40,274 data points and choose `rarity` as the scalar attribute. Under this setup, each *filtered subset* consists of vectors associated with a particular scalar value, and different *filtered subsets* exhibit distinct distributions. To evaluate whether two *filtered subsets* are OOD, ID, or in an intermediate state (referred to as Partially-Overlapping-Distribution, POD), we design hybrid queries across *filtered subsets*, identifying their relationships based on the performance gap relative to a baseline.

Experiment Design. Suppose the chosen scalar $\mathbb{S}_c$ has $|\mathbb{S}_c|$ unique values. Let "$s_{base} - s_{query}$" be a set of hybrid queries, where base vectors are sampled from the *filtered subset* where the scalar value is $s_{base} \in \mathbb{S}_c$, and query vectors are sampled from the *filtered subset* where the scalar value is $s_{query} \in \mathbb{S}_c$. This results in a total of $|\mathbb{S}_c|^2$ hybrid query sets. Among these, hybrid queries where $s_{base} = s_{query}$ are referred to as *baseline hybrid queries*, with $|\mathbb{S}_c|$ such queries in total. To achieve theoretically optimal search performance for each hybrid query, an ANN index must exist for the corresponding base vectors. This index, referred to as the *oracle partition index* [78], transforms FANNS over the entire dataset into ANNS within base vectors.

For demonstration purposes, we select 3 scalar values from each dataset: $\mathbb{S}_c = \{4, 7, 9\}$ from MNIST-8M, and $\mathbb{S}_c = \{\texttt{common}, \texttt{rare}, \texttt{uncommon}\}$ from MTG. Each hybrid query set contains 1,000 randomly sampled vectors from each *filtered subset*, with $k = 10$. Each set of base vectors has 2 types of ANN indices—IVFFlat and HNSWFlat—constructed using the Faiss library [13]. For IVFFlat, the construction parameter `nlists` (the number of inverted lists) is set to 2,048 and 256, respectively, as recommended by the library. The search parameter `nprobe` (the number of inverted lists visited during a query) is varied from 5 to 25, respectively, and the average `recall@10` is measured for each hybrid query set. For HNSWFlat, the construction parameters `M` (the number of neighbors in the graph) and `efConstruction` (the depth of exploration during construction) are both set to 32 and 40, following the library's default settings. The search parameter `efSearch` (the depth of exploration during the search) is also varied from 5 to 25, and the average `recall@10` is measured for each hybrid query set.

Experiment Results. The results are presented in Figure 3. In this figure, each set of hybrid queries is classified based on the distribution relationship between query vectors and base vectors, represented by circle, cross, and square for ID, POD, and OOD, respectively.

For both datasets, the three sets of *baseline hybrid queries* achieve the best performance, as the query vectors and base vectors belong to the same distribution, naturally classified as ID. For MNIST-8M, the remaining 6 sets of hybrid queries exhibit clear stratification and are classified as either POD or OOD, while for MTG, all 6 remaining sets have nearly identical query performance and are classified as POD. It is also observed that the relative query performance is consistent for both IVFFlat and HNSWFlat indices. This in-

[13] `https://github.com/facebookresearch/faiss/tree/v1.9.0`

dicates that ID queries are consistently easier, while POD and OOD queries are relatively more difficult, regardless of the index type. Furthermore, compared to IVFFlat, HNSWFlat demonstrates significant speedup for ID queries but offers very little improvement for POD and OOD queries. This highlights the potential for further optimization of graph-based indices in handling queries across different distribution relationships. Notably, recent studies [18, 47] have made encouraging progress in this direction.

Qualitative Explanations. To provide an initial overview of hybrid datasets and a qualitative understanding of distribution relationships between *filtered subsets*, we visualize each dataset in a 2-dimensional space using Uniform Manifold Approximation and Projection (UMAP) [72]. UMAP is a graph-based algorithm for dimensionality reduction. It first constructs a weighted k-neighbor graph and then computes a low-dimensional layout of this graph, capturing both the global structure, similar to techniques like PCA [40] and Laplacian Eigenmaps [10], and the local structure, similar to techniques like t-SNE [64] and LargeVis [88]. This graph-based dimensionality reduction process, which is closely related to graph-based indices for vector similarity search, makes UMAP a suitable tool for visualization.

In Figure 4, we perform UMAP visualizations [14] on the MNIST-8M and MTG datasets. Each *filtered subset* has a 2-dimensional distribution. For MNIST-8M, we uniformly sample 100,000 data points. The visualization reveals 10 well-separated distributions of *filtered subsets*, exhibiting clustering behavior, and the number of data points is balanced across *filtered subsets*. For MTG, we use all its 40,274 data points. The visualization reveals 6 highly overlapping distributions of *filtered subsets*, sharing a similar overall distribution, and the majority of data points are concentrated with 3 scalar values (`common`, `rare`, and `uncommon`).

The visualizations partially explain the experimental results shown in Figure 3. For MNIST-8M, hybrid queries such as those between `4` and `9` are classified as ID, as their handwritten digits are visually similar, leading to closer image embeddings and a relatively large overlap between their distributions of *filtered subsets*. In contrast, queries such as those between `4` and `7` are classified as OOD due to the significant visual differences in their handwritten digits, resulting in highly separated distributions of *filtered subsets*. For MTG, all *filtered subsets* share a similar overall distribution, resulting in nearly identical query performance across all 6 remaining sets of hybrid queries, which are classified as ID. However, the 2-dimensional nature of UMAP visualizations cannot fully capture the distribution relationships in high-dimensional space. For example, for MNIST-8M, the distributions of *filtered subsets* for `7` and `9` exhibit some degree of overlap, but the performance for `9-7` is significantly worse than that for `7-9`, leading the former to be classified as OOD and the latter as POD.

Quantitative Explanations. To complement qualitative visualizations and provide a quantitative understanding of distribution relationships between *filtered subsets*, we use the Mahalanobis distance [66], which has been employed in recent studies [18, 47] to quantify the OOD property in vector similarity search. The Mahalanobis distance measures the distance from a vector $\mathbf{v}$ to a *filtered subset* $\mathcal{D}_{f_s}$, and is defined as $d_M(\mathbf{v}, \mathcal{D}_{f_s}) = \sqrt{(\mathbf{v} - \bar{\mathbf{v}}_{\mathcal{D}_{\mathbf{f_s}}})^T S^{-1}_{\mathcal{D}_{f_s}} (\mathbf{v} - \bar{\mathbf{v}}_{\mathcal{D}_{\mathbf{f_s}}})}$, where $\bar{\mathbf{v}}_{\mathcal{D}_{f_s}}$ is the mean vector of $\mathcal{D}_{f_s}$ and $S^{-1}_{\mathcal{D}_{f_s}}$ is the inverse of the covariance matrix of $\mathcal{D}_{f_s}$.

In Figure 5, we calculate Mahalanobis distance histograms for all sets of hybrid queries in MNIST-8M and MTG. For a set of hybrid queries "$s_{base} - s_{query}$" mentioned above, we uniformly sample $\hat{\mathcal{D}}_{f_{s_{base}}}$ from $\mathcal{D}_{f_{s_{base}}}$ and $\hat{\mathcal{D}}_{f_{s_{query}}}$ from $\mathcal{D}_{f_{s_{query}}}$, each containing 5,000 data points. Notably, when $\hat{\mathcal{D}}_{f_{s_{base}}} = \hat{\mathcal{D}}_{f_{s_{query}}}$, the two sampled subsets are required to be non-intersecting. We then use an open-source library [15] to compute the Mahalanobis distances in the Euclidean space. Each subgraph titled "$base - query$" shows both the orange histogram of Mahalanobis distances from each vector in $\hat{\mathcal{D}}_{f_{s_{base}}}$ to $\hat{\mathcal{D}}_{f_{s_{query}}}$, and the blue histogram of Mahalanobis distances from each vector in $\hat{\mathcal{D}}_{f_{s_{base}}}$ to $\hat{\mathcal{D}}_{f_{s_{base}}}$, illustrating the distribution shift between query vectors and base vectors when sampled from different *filtered subsets*, compared to when they are sampled from the same *filtered subset*.

The calculations also partially explain the experimental results shown in Figure 3. For both datasets, in each diagonal subgraphs (from the top left to the bottom right), two histograms show complete overlap between orange and blue, which explains why *baseline hybrid queries* achieve the highest performance and are classified as ID (Figure 3). The off-diagonal subgraphs are asymmetric, as the Mahalanobis distance depends on the covariance matrix of the base vectors. This explains why `7-9` is POD but `9-7` is OOD (Figure 3). In each off-diagonal subgraph, two histograms exhibit

[14] `https://github.com/lmcinnes/umap/tree/release-0.5.7`

[15] `https://github.com/mosegui/mahalanobis`. We modify the original library to compute the pseudo-inverse of the covariance matrix instead of the regular inverse for robustness.

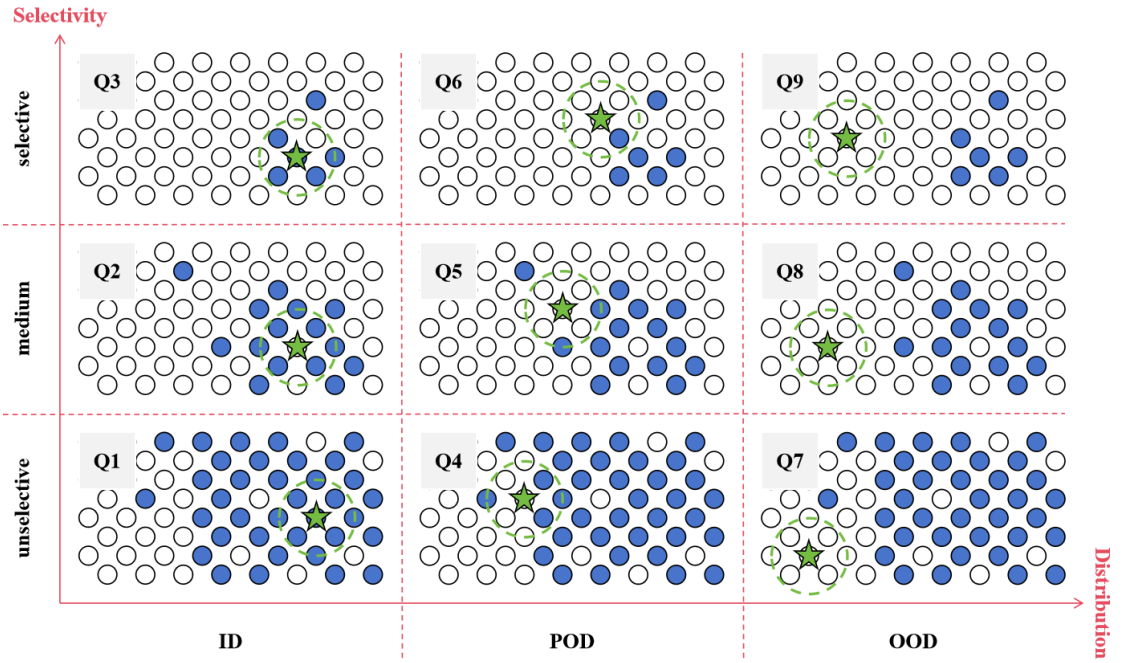

Fig. 6 Hybrid query sets with varying levels of difficulty under the control of both the *distribution* factor and the *selectivity* factor. Each region represents a hybrid query set, where discs represent data points, base vectors are blue discs, and query vectors locate in a green dashed circular area centered on a green pentagram. A lower proportion of blue discs indicates higher *selectivity*, and less overlap between the green and blue area indicates greater proximity to OOD.

some degree of overlap. Notably, `4-9` and `7-9` in MNIST-8M and all the remaining pairs in MTG show near-complete overlap between orange and blue, which explains their relative high performance and are classified as POD (Figure 3). However, the remaining subgraphs in MNIST-8M cannot be classified as POD or OOD based solely on the size of the overlapping region. For example, while the overlap size suggests that `4-7` > `9-7` > `9-4`, these pairs are classified as OOD, POD, and OOD, respectively (Figure 3).

The above analysis suggests that the Mahalanobis distance is still imperfect. Its success in distinguishing OOD queries in existing studies [18, 47] can be largely attributed to the fact that the base vectors and query vectors are sampled from different modalities (e.g., text and image embeddings generated by CLIP [81]), which are inherently far apart and have almost no overlap, making them easy to distinguish. However, in the context of hybrid queries, the distinguishing ability of Mahalanobis distance is insufficient, as it fails to completely classify query properties (POD or OOD) based on the degree of histogram overlap. This highlights the need to develop a more precise metric, which can better support the design of the set of hybrid queries with desired distribution relationship between base vectors and query vectors. Notably, a recent study [18] explored the Wasserstein distance as an alternative, but it suffers from symmetry, meaning that the distances between `x-y` and `y-x` are the same, making it less effective than Mahalanobis distance in explaining query difficulty.

5.3 Towards More Comprehensive Evaluation of FANNS Algorithms

While the *selectivity* factor measures the proportion of data points excluded by the scalar filter, the *distribution* factor provides insights into the relationship between distributions of base vectors and query vectors. As discussed above, both factors contribute to the hybrid query difficulty. Combining these two factors, we propose a schema towards more comprehensive evaluation of FANNS algorithms, in which each factor serves as a dimension to partition the space of hybrid query sets with varying levels of difficulty.

Figure 6 illustrates the space partitioned by these two factors, *distribution* and *selectivity*. There are 9 sets of hybrid queries (**Q1**-**Q9**) across 9 regions, with each set representing a combination of a specific *distribution* and *selectivity*. In each region, discs represent data points, the base vectors are discs with blue color, and the query vectors are enclosed within a green dashed circular area centered on a green pentagram. A lower proportion of blue discs in a region indicates higher *selectivity*, and less overlap between the green and blue area indicates greater proximity to OOD.

Although the *selectivity* can be easily controlled by adjusting the range of scalar values, there is currently no straightforward method for controlling the *distribution*. As discussed in subsection 5.2, an effective metric for quantifying the distribution relationship between base vectors and query vectors is still lacking. Consequently, the practical realization of our proposed schema depends on the development of a more suitable metric for measuring the *distribution* factor. We view it as an important research direction coming out of this survey.

6 Open Questions and Research Directions

Aside from the above-mentioned research direction, this section discusses open questions and possible research directions in the field of FANNS, progressing from the internal structures of its indices, to the workload-aware optimization of its algorithms, and ultimately to its system-level solutions.

6.1 Innovation in Index Structures

Designing efficient FANNS indices through the integration of traditional data structures is a notable trend.

For example, NHQ (**A7**) leverages the *prefix tree* [15], iRangeGraph (**A11**) and WST (**A17**) utilize the *segment trees* [28, 99], and HQI (**A14**) employs the *qd-tree* [73]. These integrations demonstrate that traditional data structures can play a crucial role in building more efficient FANNS indices.

However, existing FANNS indices integrated with traditional data structures are tightly coupled with specific assumptions, such as a *simplified equality scalar filter* or a *simplified range scalar filter* (subsection 2.2). Therefore, a key open question remains: how to integrate or design data structures that can support efficient FANNS indexing under more general scalar filters.

6.2 Optimization via Realistic Workloads

Optimizing dedicated FANNS algorithms based on realistic workloads is another important research direction.

For instance, CAPS (**A6**) partitions clusters of the IVF index according to the power-law distribution of scalar values [35], Milvus (**A13**) performs pre-indexing dataset partitioning based on frequently queried scalar values [91], and HQI (**A14**) benefits from workloads that exhibit scalar filter stability [73].

The primary challenge here is how to effectively identify and quantify workload characteristics in specific application scenarios, and to perform targeted optimizations based on these characteristics.

6.3 Combination of Multiple Algorithms

Combining multiple FANNS algorithms and dynamically select the most suitable one for a given hybrid query is a promising direction at the system level. This strategy enables the system to exploit the strengths of different FANNS algorithms under varying conditions.

Several studies have explored this strategy by estimating *selectivity* for algorithm selection, typically using Pre-Filtering Algorithm Family (**A12**) for high *selectivity* and pre-built FANNS indices for moderate or low *selectivity*. For example, database systems such as ADBV [96], Milvus [91], and VBase [102] utilize cost models to estimate *selectivity* and dynamically switch between FANNS algorithms. In the case of ACORN (**A4**), the algorithm shifts to Pre-Filtering Algorithm Family (**A12**) when *selectivity* is high [78].

Remaining challenges include identifying and estimating more workload-aware metrics beyond *selectivity* to guide algorithm selection, and optimizing storage efficiency when multiple indices coexist.

7 Conclusion

In this paper, we formally defined the FANNS problem, covering the hybrid dataset, the hybrid query, and key evaluation metrics. We then proposed a pruning-focused framework to classify and summarize existing FANNS algorithms. Next, we reviewed existing hybrid datasets, outlining their construction strategies and detailing their contents. We also endeavored to understand the difficulty of hybrid queries by incorporating *distribution* in addition to *selectivity*, providing insights through qualitative visualizations and quantitative measurements, and proposed a schema towards more comprehensive evaluation of FANNS algorithms. Finally, we highlighted open questions and possible research directions. Going forward, we will try to develop a benchmark for enabling more comprehensive evaluation of FANNS algorithms.

Acknowledgements This paper was supported by National Natural Science Foundation of China (U24A20232), and Key Technology Research and Development Program of Shandong Province (2024CXGC010113).

References

1. Abramov E, Palchikov N (2024) Embedding-based search in jetbrains ides. In: Bezzubov A, Dig D, Bryksin T, Golubev Y (eds) Proceedings of the

1st ACM/IEEE Workshop on Integrated Development Environments, IDE 2024, Lisbon, Portugal, 20 April 2024, ACM, pp 62–65, DOI 10.1145/3643796.3648456, URL https://doi.org/10.1145/3643796.3648456

2. Abu-El-Haija S, Kothari N, Lee J, Natsev P, Toderici G, Varadarajan B, Vijayanarasimhan S (2016) Youtube-8m: A large-scale video classification benchmark. CoRR abs/1609.08675, URL http://arxiv.org/abs/1609.08675, 1609.08675
3. Ahle TD, Aumüller M, Pagh R (2017) Parameter-free locality sensitive hashing for spherical range reporting. In: Klein PN (ed) Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19, SIAM, pp 239–256, DOI 10.1137/1.9781611974782.16, URL https://doi.org/10.1137/1.9781611974782.16
4. Aumüller M, Ceccarello M (2019) The role of local intrinsic dimensionality in benchmarking nearest neighbor search. In: Amato G, Gennaro C, Oria V, Radovanovic M (eds) Similarity Search and Applications - 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2-4, 2019, Proceedings, Springer, Lecture Notes in Computer Science, vol 11807, pp 113–127, DOI 10.1007/978-3-030-32047-8_11, URL https://doi.org/10.1007/978-3-030-32047-8_11
5. Aumüller M, Ceccarello M (2021) The role of local dimensionality measures in benchmarking nearest neighbor search. Inf Syst 101:101,807, DOI 10.1016/J.IS.2021.101807, URL https://doi.org/10.1016/j.is.2021.101807
6. Aumüller M, Bernhardsson E, Faithfull AJ (2020) Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Inf Syst 87, DOI 10.1016/J.IS.2019.02.006, URL https://doi.org/10.1016/j.is.2019.02.006
7. Aurenhammer F, Klein R, Lee D (2013) Voronoi Diagrams and Delaunay Triangulations. World Scientific, DOI 10.1142/8685, URL https://doi.org/10.1142/8685
8. Babenko A, Lempitsky VS (2016) Efficient indexing of billion-scale datasets of deep descriptors. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, pp 2055–2063, DOI 10.1109/CVPR.2016.226, URL https://doi.org/10.1109/CVPR.2016.226
9. Barachanou A, Tsalakanidou F, Papadopoulos S (2024) REBECCA at erisk 2024: Search for symptoms of depression using sentence embeddings and prompt-based filtering. In: Faggioli G, Ferro N, Galuscáková P, de Herrera AGS (eds) Working Notes of the Conference and Labs of the Evaluation Forum (CLEF 2024), Grenoble, France, 9-12 September, 2024, CEUR-WS.org, CEUR Workshop Proceedings, vol 3740, pp 794–802, URL https://ceur-ws.org/Vol-3740/paper-74.pdf
10. Belkin M, Niyogi P (2003) Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput 15(6):1373–1396, DOI 10.1162/089976603321780317, URL https://doi.org/10.1162/089976603321780317
11. Bentley JL (1975) Multidimensional binary search trees used for associative searching. Commun ACM 18(9):509–517, DOI 10.1145/361002.361007, URL https://doi.org/10.1145/361002.361007
12. Borisyuk F, Malreddy S, Mei J, Liu Y, Liu X, Maheshwari P, Bell A, Rangadurai K (2021) Visrel: Media search at scale. In: Zhu F, Ooi BC, Miao C (eds) KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021, ACM, pp 2584–2592, DOI 10.1145/3447548.3467081, URL https://doi.org/10.1145/3447548.3467081
13. Brown TB, Mann B, Ryder N, Subbiah M, Kaplan J, Dhariwal P, Neelakantan A, Shyam P, Sastry G, Askell A, Agarwal S, Herbert-Voss A, Krueger G, Henighan T, Child R, Ramesh A, Ziegler DM, Wu J, Winter C, Hesse C, Chen M, Sigler E, Litwin M, Gray S, Chess B, Clark J, Berner C, McCandlish S, Radford A, Sutskever I, Amodei D (2020) Language models are few-shot learners. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H (eds) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html
14. Cai D (2021) A revisit of hashing algorithms for approximate nearest neighbor search. IEEE Trans Knowl Data Eng 33(6):2337–2348, DOI 10.1109/TKDE.2019.2953897, URL https://doi.org/10.1109/TKDE.2019.2953897
15. Cai Y, Shi J, Chen Y, Zheng W (2024) Navigating labels and vectors: A unified approach to fil-

tered approximate nearest neighbor search. Proc ACM Manag Data 2(6), DOI 10.1145/3698822, URL https://doi.org/10.1145/3698822

16. Chang Z, Yu L, Xu Y, Hu W (2024) Neural embeddings for knn search in biological sequence. In: Wooldridge MJ, Dy JG, Natarajan S (eds) Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, AAAI Press, pp 38–45, DOI 10.1609/AAAI.V38I1.27753, URL https://doi.org/10.1609/aaai.v38i1.27753
17. Charikar M (2002) Similarity estimation techniques from rounding algorithms. In: Reif JH (ed) Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada, ACM, pp 380–388, DOI 10.1145/509907.509965, URL https://doi.org/10.1145/509907.509965
18. Chen M, Zhang K, He Z, Jing Y, Wang XS (2024) Roargraph: A projected bipartite graph for efficient cross-modal approximate nearest neighbor search. Proc VLDB Endow 17(11):2735–2749, DOI 10.14778/3681954.3681959, URL https://www.vldb.org/pvldb/vol17/p2735-chen.pdf
19. Chen Y, Guan T, Wang C (2010) Approximate nearest neighbor search by residual vector quantization. Sensors 10(12):11,259–11,273, DOI 10.3390/S101211259, URL https://doi.org/10.3390/s101211259
20. Cherti M, Beaumont R, Wightman R, Wortsman M, Ilharco G, Gordon C, Schuhmann C, Schmidt L, Jitsev J (2023) Reproducible scaling laws for contrastive language-image learning. In: IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023, IEEE, pp 2818–2829, DOI 10.1109/CVPR52729.2023.00276, URL https://doi.org/10.1109/CVPR52729.2023.00276
21. Ciaccia P, Patella M, Zezula P (1997) M-tree: An efficient access method for similarity search in metric spaces. In: Jarke M, Carey MJ, Dittrich KR, Lochovsky FH, Loucopoulos P, Jeusfeld MA (eds) VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece, Morgan Kaufmann, pp 426–435, URL http://www.vldb.org/conf/1997/P426.PDF
22. Cortes C, Vapnik V (1995) Support-vector networks. Mach Learn 20(3):273–297, DOI 10.1007/BF00994018, URL https://doi.org/10.1007/BF00994018
23. Dasgupta S, Freund Y (2008) Random projection trees and low dimensional manifolds. In: Dwork C (ed) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008, ACM, pp 537–546, DOI 10.1145/1374376.1374452, URL https://doi.org/10.1145/1374376.1374452
24. Deng L (2012) The MNIST database of handwritten digit images for machine learning research [best of the web]. IEEE Signal Process Mag 29(6):141–142, DOI 10.1109/MSP.2012.2211477, URL https://doi.org/10.1109/MSP.2012.2211477
25. Devlin J, Chang M, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C, Solorio T (eds) Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers), Association for Computational Linguistics, pp 4171–4186, DOI 10.18653/V1/N19-1423, URL https://doi.org/10.18653/v1/n19-1423
26. Douze M, Guzhva A, Deng C, Johnson J, Szilvasy G, Mazaré P, Lomeli M, Hosseini L, Jégou H (2024) The faiss library. CoRR abs/2401.08281, DOI 10.48550/ARXIV.2401.08281, URL https://doi.org/10.48550/arXiv.2401.08281, 2401.08281
27. Du M, Ramisa A, C AKK, Chanda S, Wang M, Rajesh N, Li S, Hu Y, Zhou T, Lakshminarayana N, Tran S, Gray D (2022) Amazon shop the look: A visual search system for fashion and home. In: Zhang A, Rangwala H (eds) KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022, ACM, pp 2822–2830, DOI 10.1145/3534678.3539071, URL https://doi.org/10.1145/3534678.3539071
28. Engels J, Landrum B, Yu S, Dhulipala L, Shun J (2024) Approximate nearest neighbor search with window filters. In: Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024, OpenReview.net, URL https://openreview.net/forum?id=8t8zBaGFar

29. Fu C, Xiang C, Wang C, Cai D (2019) Fast approximate nearest neighbor search with the navigating spreading-out graph. Proc VLDB Endow 12(5):461–474, DOI 10.14778/3303753.3303754, URL http://www.vldb.org/pvldb/vol12/p461-fu.pdf
30. Fu C, Wang C, Cai D (2022) High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility. IEEE Trans Pattern Anal Mach Intell 44(8):4139–4150, DOI 10.1109/TPAMI.2021.3067706, URL https://doi.org/10.1109/TPAMI.2021.3067706
31. Gao Y, Xiong Y, Gao X, Jia K, Pan J, Bi Y, Dai Y, Sun J, Guo Q, Wang M, Wang H (2023) Retrieval-augmented generation for large language models: A survey. CoRR abs/2312.10997, DOI 10.48550/ARXIV.2312.10997, URL https://doi.org/10.48550/arXiv.2312.10997, 2312.10997
32. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Atkinson MP, Orlowska ME, Valduriez P, Zdonik SB, Brodie ML (eds) VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK, Morgan Kaufmann, pp 518–529, URL http://www.vldb.org/conf/1999/P49.pdf
33. Gollapudi S, Karia N, Sivashankar V, Krishnaswamy R, Begwani N, Raz S, Lin Y, Zhang Y, Mahapatro N, Srinivasan P, Singh A, Simhadri HV (2023) Filtered-diskann: Graph algorithms for approximate nearest neighbor search with filters. In: Ding Y, Tang J, Sequeda JF, Aroyo L, Castillo C, Houben G (eds) Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023, ACM, pp 3406–3416, DOI 10.1145/3543507.3583552, URL https://doi.org/10.1145/3543507.3583552
34. Gupta D, Loane RF, Gayen S, Demner-Fushman D (2023) Medical image retrieval via nearest neighbor search on pre-trained image features. Knowl Based Syst 278:110,907, DOI 10.1016/J.KNOSYS.2023.110907, URL https://doi.org/10.1016/j.knosys.2023.110907
35. Gupta G, Yi J, Coleman B, Luo C, Lakshman V, Shrivastava A (2023) CAPS: A practical partition index for filtered similarity search. CoRR abs/2308.15014, DOI 10.48550/ARXIV.2308.15014, URL https://doi.org/10.48550/arXiv.2308.15014, 2308.15014
36. He J, Kumar S, Chang S (2012) On the difficulty of nearest neighbor search. In: Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012, icml.cc / Omnipress, URL http://icml.cc/2012/papers/580.pdf
37. He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016, IEEE Computer Society, pp 770–778, DOI 10.1109/CVPR.2016.90, URL https://doi.org/10.1109/CVPR.2016.90
38. He Y, Tian Y, Wang M, Chen F, Yu L, Tang M, Chen C, Zhang N, Kuang B, Prakash A (2023) Que2engage: Embedding-based retrieval for relevant and engaging products at facebook marketplace. In: Ding Y, Tang J, Sequeda JF, Aroyo L, Castillo C, Houben G (eds) Companion Proceedings of the ACM Web Conference 2023, WWW 2023, Austin, TX, USA, 30 April 2023 - 4 May 2023, ACM, pp 386–390, DOI 10.1145/3543873.3584633, URL https://doi.org/10.1145/3543873.3584633
39. Heo J, Lee Y, He J, Chang S, Yoon S (2012) Spherical hashing. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012, IEEE Computer Society, pp 2957–2964, DOI 10.1109/CVPR.2012.6248024, URL https://doi.org/10.1109/CVPR.2012.6248024
40. Hotelling H (1933) Analysis of a complex of statistical variables into principal components. Journal of Educational Psychology 24(6):417–441, DOI 10.1037/h0071325, URL https://doi.org/10.1037/h0071325
41. Houle ME (2013) Dimensionality, discriminability, density and distance distributions. In: Ding W, Washio T, Xiong H, Karypis G, Thuraisingham B, Cook DJ, Wu X (eds) 13th IEEE International Conference on Data Mining Workshops, ICDM Workshops, TX, USA, December 7-10, 2013, IEEE Computer Society, pp 468–473, DOI 10.1109/ICDMW.2013.139, URL https://doi.org/10.1109/ICDMW.2013.139
42. Hu Y, Lu Y (2024) RAG and RAU: A survey on retrieval-augmented language model in natural language processing. CoRR abs/2404.19543, DOI 10.48550/ARXIV.2404.19543, URL https://doi.org/10.48550/arXiv.2404.19543, 2404.19543
43. Huang Y, Huang J (2024) A survey on retrieval-augmented text generation for large language models. CoRR abs/2404.10981, DOI 10.48550/ARXIV.2404.10981, URL https://doi.org/10.48550/arXiv.2404.10981, 2404.10981

44. Ilyas IF, Rekatsinas T, Konda V, Pound J, Qi X, Soliman MA (2022) Saga: A platform for continuous construction and serving of knowledge at scale. In: Ives ZG, Bonifati A, Abbadi AE (eds) SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12 - 17, 2022, ACM, pp 2259–2272, DOI 10.1145/3514221.3526049, URL https://doi.org/10.1145/3514221.3526049
45. Indyk P, Motwani R (1998) Approximate nearest neighbors: Towards removing the curse of dimensionality. In: Vitter JS (ed) Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing, Dallas, Texas, USA, May 23-26, 1998, ACM, pp 604–613, DOI 10.1145/276698.276876, URL https://doi.org/10.1145/276698.276876
46. Ioffe S, Szegedy C (2015) Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Bach FR, Blei DM (eds) Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015, JMLR.org, JMLR Workshop and Conference Proceedings, vol 37, pp 448–456, URL http://proceedings.mlr.press/v37/ioffe15.html
47. Jaiswal S, Krishnaswamy R, Garg A, Simhadri HV, Agrawal S (2022) Ood-diskann: Efficient and scalable graph ANNS for out-of-distribution queries. CoRR abs/2211.12850, DOI 10.48550/ARXIV.2211.12850, URL https://doi.org/10.48550/arXiv.2211.12850, 2211.12850
48. Jégou H, Douze M, Schmid C (2008) Hamming embedding and weak geometric consistency for large scale image search. In: Forsyth DA, Torr PHS, Zisserman A (eds) Computer Vision - ECCV 2008, 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part I, Springer, Lecture Notes in Computer Science, vol 5302, pp 304–317, DOI 10.1007/978-3-540-88682-2_24, URL https://doi.org/10.1007/978-3-540-88682-2_24
49. Jégou H, Douze M, Schmid C (2011) Product quantization for nearest neighbor search. IEEE Trans Pattern Anal Mach Intell 33(1):117–128, DOI 10.1109/TPAMI.2010.57, URL https://doi.org/10.1109/TPAMI.2010.57
50. Jégou H, Tavenard R, Douze M, Amsaleg L (2011) Searching in one billion vectors: Re-rank with source coding. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic, IEEE, pp 861–864, DOI 10.1109/ICASSP.2011.5946540, URL https://doi.org/10.1109/ICASSP.2011.5946540
51. Jin Z, Li C, Lin Y, Cai D (2014) Density sensitive hashing. IEEE Trans Cybern 44(8):1362–1371, DOI 10.1109/TCYB.2013.2283497, URL https://doi.org/10.1109/TCYB.2013.2283497
52. Kirchoff KE, Wellnitz J, Hochuli JE, Maxfield T, Popov KI, Gomez SM, Tropsha A (2024) Utilizing low-dimensional molecular embeddings for rapid chemical similarity search. In: Goharian N, Tonellotto N, He Y, Lipani A, McDonald G, Macdonald C, Ounis I (eds) Advances in Information Retrieval - 46th European Conference on Information Retrieval, ECIR 2024, Glasgow, UK, March 24-28, 2024, Proceedings, Part II, Springer, Lecture Notes in Computer Science, vol 14609, pp 34–49, DOI 10.1007/978-3-031-56060-6_3, URL https://doi.org/10.1007/978-3-031-56060-6_3
53. Kosonocky CW, Feller AL, Wilke CO, Ellington AD (2023) Using alternative SMILES representations to identify novel functional analogues in chemical similarity vector searches. Patterns 4(12):100,865, DOI 10.1016/J.PATTER.2023.100865, URL https://doi.org/10.1016/j.patter.2023.100865
54. Kulis B, Darrell T (2009) Learning to hash with binary reconstructive embeddings. In: Bengio Y, Schuurmans D, Lafferty JD, Williams CKI, Culotta A (eds) Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada, Curran Associates, Inc., pp 1042–1050, URL https://proceedings.neurips.cc/paper/2009/hash/6602294be910b1e3c4571bd98c4d5484-Abstract.html
55. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. Proc IEEE 86(11):2278–2324, DOI 10.1109/5.726791, URL https://doi.org/10.1109/5.726791
56. Lewis PSH, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, Küttler H, Lewis M, Yih W, Rocktäschel T, Riedel S, Kiela D (2020) Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Larochelle H, Ranzato M, Hadsell R, Balcan M, Lin H (eds) Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, URL https:

//proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html
57. Li H, Ai Q, Zhan J, Mao J, Liu Y, Liu Z, Cao Z (2023) Constructing tree-based index for efficient and effective dense retrieval. In: Chen H, Duh WE, Huang H, Kato MP, Mothe J, Poblete B (eds) Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2023, Taipei, Taiwan, July 23-27, 2023, ACM, pp 131–140, DOI 10.1145/3539618.3591651, URL https://doi.org/10.1145/3539618.3591651
58. Li J, Liu H, Gui C, Chen J, Ni Z, Wang N, Chen Y (2018) The design and implementation of a real time visual search system on JD e-commerce platform. In: Proceedings of the 19th International Middleware Conference, Middleware Industrial Track 2018, Rennes, France, December 10-14, 2018, ACM, pp 9–16, DOI 10.1145/3284028.3284030, URL https://doi.org/10.1145/3284028.3284030
59. Li S, Lv F, Jin T, Lin G, Yang K, Zeng X, Wu X, Ma Q (2021) Embedding-based product retrieval in taobao search. In: Zhu F, Ooi BC, Miao C (eds) KDD '21: The 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, Singapore, August 14-18, 2021, ACM, pp 3181–3189, DOI 10.1145/3447548.3467101, URL https://doi.org/10.1145/3447548.3467101
60. Li W, Zhang Y, Sun Y, Wang W, Li M, Zhang W, Lin X (2020) Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement. IEEE Trans Knowl Data Eng 32(8):1475–1488, DOI 10.1109/TKDE.2019.2909204, URL https://doi.org/10.1109/TKDE.2019.2909204
61. Lin J, Yadav S, Liu F, Rossi N, Suram PR, Chembolu S, Chandran P, Mohapatra H, Lee T, Magnani A, Liao C (2024) Enhancing relevance of embedding-based retrieval at walmart. In: Serra E, Spezzano F (eds) Proceedings of the 33rd ACM International Conference on Information and Knowledge Management, CIKM 2024, Boise, ID, USA, October 21-25, 2024, ACM, pp 4694–4701, DOI 10.1145/3627673.3680047, URL https://doi.org/10.1145/3627673.3680047
62. Loosli G, Canu S, Bottou L (2007) Training invariant support vector machines using selective sampling. In: Bottou L, Chapelle O, DeCoste D, Weston J (eds) Large Scale Kernel Machines, MIT Press, Cambridge, MA., pp 301–320, URL http://leon.bottou.org/papers/loosli-canu-bottou-2006
63. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110, DOI 10.1023/B:VISI.0000029664.99615.94, URL https://doi.org/10.1023/B:VISI.0000029664.99615.94
64. van der Maaten L, Hinton G (2008) Visualizing data using t-sne. Journal of Machine Learning Research 9(86):2579–2605, URL http://jmlr.org/papers/v9/vandermaaten08a.html
65. Magnani A, Liu F, Chaidaroon S, Yadav S, Suram PR, Puthenputhussery A, Chen S, Xie M, Kashi A, Lee T, Liao C (2022) Semantic retrieval at walmart. In: Zhang A, Rangwala H (eds) KDD '22: The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 14 - 18, 2022, ACM, pp 3495–3503, DOI 10.1145/3534678.3539164, URL https://doi.org/10.1145/3534678.3539164
66. Mahalanobis PC (2018) Reprint of: Mahalanobis, p.c. (1936) "on the generalised distance in statistics.". Sankhya A 80(1):1–7, DOI 10.1007/s13171-019-00164-5, URL https://doi.org/10.1007/s13171-019-00164-5
67. Malkov Y, Ponomarenko A, Logvinov A, Krylov V (2014) Approximate nearest neighbor algorithm based on navigable small world graphs. Inf Syst 45:61–68, DOI 10.1016/J.IS.2013.10.006, URL https://doi.org/10.1016/j.is.2013.10.006
68. Malkov YA, Yashunin DA (2020) Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans Pattern Anal Mach Intell 42(4):824–836, DOI 10.1109/TPAMI.2018.2889473, URL https://doi.org/10.1109/TPAMI.2018.2889473
69. Martinez J, Clement J, Hoos HH, Little JJ (2016) Revisiting additive quantization. In: Leibe B, Matas J, Sebe N, Welling M (eds) Computer Vision - ECCV 2016 - 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II, Springer, Lecture Notes in Computer Science, vol 9906, pp 137–153, DOI 10.1007/978-3-319-46475-6_9, URL https://doi.org/10.1007/978-3-319-46475-6_9
70. Martinez J, Zakhmi S, Hoos HH, Little JJ (2018) LSQ++: lower running time and higher recall in multi-codebook quantization. In: Ferrari V, Hebert M, Sminchisescu C, Weiss Y (eds) Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XVI, Springer, Lecture Notes in Computer Science, vol 11220, pp 508–523, DOI 10.1007/978-3-030-01270-0_30, URL https://doi.

org/10.1007/978-3-030-01270-0_30
71. McAuley JJ, Targett C, Shi Q, van den Hengel A (2015) Image-based recommendations on styles and substitutes. In: Baeza-Yates R, Lalmas M, Moffat A, Ribeiro-Neto BA (eds) Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval, Santiago, Chile, August 9-13, 2015, ACM, pp 43–52, DOI 10.1145/2766462.2767755, URL https://doi.org/10.1145/2766462.2767755
72. McInnes L, Healy J (2018) UMAP: uniform manifold approximation and projection for dimension reduction. CoRR abs/1802.03426, URL http://arxiv.org/abs/1802.03426, 1802.03426
73. Mohoney J, Pacaci A, Chowdhury SR, Mousavi A, Ilyas IF, Minhas UF, Pound J, Rekatsinas T (2023) High-throughput vector similarity search in knowledge graphs. Proc ACM Manag Data 1(2):197:1–197:25, DOI 10.1145/3589777, URL https://doi.org/10.1145/3589777
74. Muja M, Lowe DG (2009) Fast approximate nearest neighbors with automatic algorithm configuration. In: Ranchordas A, Araújo H (eds) VISAPP 2009 - Proceedings of the Fourth International Conference on Computer Vision Theory and Applications, Lisboa, Portugal, February 5-8, 2009 - Volume 1, INSTICC Press, pp 331–340
75. Norouzi M, Fleet DJ (2013) Cartesian k-means. In: 2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013, IEEE Computer Society, pp 3017–3024, DOI 10.1109/CVPR.2013.388, URL https://doi.org/10.1109/CVPR.2013.388
76. Oliva A, Torralba A (2001) Modeling the shape of the scene: A holistic representation of the spatial envelope. Int J Comput Vis 42(3):145–175, DOI 10.1023/A:1011139631724, URL https://doi.org/10.1023/A:1011139631724
77. Pan JJ, Wang J, Li G (2024) Survey of vector database management systems. VLDB J 33(5):1591–1615, DOI 10.1007/S00778-024-00864-X, URL https://doi.org/10.1007/s00778-024-00864-x
78. Patel L, Kraft P, Guestrin C, Zaharia M (2024) ACORN: performant and predicate-agnostic search over vector embeddings and structured data. Proc ACM Manag Data 2(3):120, DOI 10.1145/3654923, URL https://doi.org/10.1145/3654923
79. Peng Y, Choi B, Chan TN, Yang J, Xu J (2023) Efficient approximate nearest neighbor search in multi-dimensional databases. Proc ACM Manag Data 1(1):54:1–54:27, DOI 10.1145/3588908, URL https://doi.org/10.1145/3588908
80. Pennington J, Socher R, Manning CD (2014) Glove: Global vectors for word representation. In: Moschitti A, Pang B, Daelemans W (eds) Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL, ACL, pp 1532–1543, DOI 10.3115/V1/D14-1162, URL https://doi.org/10.3115/v1/d14-1162
81. Radford A, Kim JW, Hallacy C, Ramesh A, Goh G, Agarwal S, Sastry G, Askell A, Mishkin P, Clark J, Krueger G, Sutskever I (2021) Learning transferable visual models from natural language supervision. In: Meila M, Zhang T (eds) Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event, PMLR, Proceedings of Machine Learning Research, vol 139, pp 8748–8763, URL http://proceedings.mlr.press/v139/radford21a.html
82. Raza S, Ding C (2022) News recommender system: a review of recent progress, challenges, and opportunities. Artif Intell Rev 55(1):749–800, DOI 10.1007/S10462-021-10043-X, URL https://doi.org/10.1007/s10462-021-10043-x
83. Schuhmann C, Vencu R, Beaumont R, Kaczmarczyk R, Mullis C, Katta A, Coombes T, Jitsev J, Komatsuzaki A (2021) LAION-400M: open dataset of clip-filtered 400 million image-text pairs. CoRR abs/2111.02114, URL https://arxiv.org/abs/2111.02114, 2111.02114
84. Schuhmann C, Beaumont R, Vencu R, Gordon C, Wightman R, Cherti M, Coombes T, Katta A, Mullis C, Wortsman M, Schramowski P, Kundurthy S, Crowson K, Schmidt L, Kaczmarczyk R, Jitsev J (2022) LAION-5B: an open large-scale dataset for training next generation image-text models. In: Koyejo S, Mohamed S, Agarwal A, Belgrave D, Cho K, Oh A (eds) Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022, URL http://papers.nips.cc/paper_files/paper/2022/hash/a1859debfb3b59d094f3504d5ebb6c25-Abstract-Datasets_and_Benchmarks.html
85. Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Bengio Y, LeCun Y (eds) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May

7-9, 2015, Conference Track Proceedings, URL http://arxiv.org/abs/1409.1556

86. Subramanya SJ, Devvrit, Simhadri HV, Krishnaswamy R, Kadekodi R (2019) Diskann: Fast accurate billion-point nearest neighbor search on a single node. In: Wallach HM, Larochelle H, Beygelzimer A, d'Alché-Buc F, Fox EB, Garnett R (eds) Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada, pp 13,748–13,758, URL https://proceedings.neurips.cc/paper/2019/hash/09853c7fb1d3f8ee67a61b6bf4a7f8e6-Abstract.html
87. Szegedy C, Liu W, Jia Y, Sermanet P, Reed SE, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2014) Going deeper with convolutions. CoRR abs/1409.4842, URL http://arxiv.org/abs/1409.4842, 1409.4842
88. Tang J, Liu J, Zhang M, Mei Q (2016) Visualizing large-scale and high-dimensional data. In: Bourdeau J, Hendler J, Nkambou R, Horrocks I, Zhao BY (eds) Proceedings of the 25th International Conference on World Wide Web, WWW 2016, Montreal, Canada, April 11 - 15, 2016, ACM, pp 287–297, DOI 10.1145/2872427.2883041, URL https://doi.org/10.1145/2872427.2883041
89. Toussaint GT (1980) The relative neighbourhood graph of a finite planar set. Pattern Recognit 12(4):261–268, DOI 10.1016/0031-3203(80)90066-7, URL https://doi.org/10.1016/0031-3203(80)90066-7
90. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser L, Polosukhin I (2017) Attention is all you need. In: Guyon I, von Luxburg U, Bengio S, Wallach HM, Fergus R, Vishwanathan SVN, Garnett R (eds) Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA, pp 5998–6008, URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
91. Wang J, Yi X, Guo R, Jin H, Xu P, Li S, Wang X, Guo X, Li C, Xu X, Yu K, Yuan Y, Zou Y, Long J, Cai Y, Li Z, Zhang Z, Mo Y, Gu J, Jiang R, Wei Y, Xie C (2021) Milvus: A purpose-built vector data management system. In: Li G, Li Z, Idreos S, Srivastava D (eds) SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021, ACM, pp 2614–2627, DOI 10.1145/3448016.3457550, URL https://doi.org/10.1145/3448016.3457550
92. Wang M, Xu X, Yue Q, Wang Y (2021) A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. Proc VLDB Endow 14(11):1964–1978, DOI 10.14778/3476249.3476255, URL http://www.vldb.org/pvldb/vol14/p1964-wang.pdf
93. Wang M, Lv L, Xu X, Wang Y, Yue Q, Ni J (2023) An efficient and robust framework for approximate nearest neighbor search with attribute constraint. In: Oh A, Naumann T, Globerson A, Saenko K, Hardt M, Levine S (eds) Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, URL http://papers.nips.cc/paper_files/paper/2023/hash/32e41d6b0a51a63a9a90697da19d235d-Abstract-Conference.html
94. Wang Z, Wang P, Palpanas T, Wang W (2023) Graph- and tree-based indexes for high-dimensional vector similarity search: Analyses, comparisons, and future directions. IEEE Data Eng Bull 46(3):3–21, URL http://sites.computer.org/debull/A23sept/p3.pdf
95. Wang Z, Wang Q, Cheng X, Wang P, Palpanas T, Wang W (2024) Steiner-hardness: A query hardness measure for graph-based ANN indexes. CoRR abs/2408.13899, DOI 10.48550/ARXIV.2408.13899, URL https://doi.org/10.48550/arXiv.2408.13899, 2408.13899
96. Wei C, Wu B, Wang S, Lou R, Zhan C, Li F, Cai Y (2020) Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. Proc VLDB Endow 13(12):3152–3165, DOI 10.14778/3415478.3415541, URL http://www.vldb.org/pvldb/vol13/p3152-wei.pdf
97. Wu W, He J, Qiao Y, Fu G, Liu L, Yu J (2022) HQANN: efficient and robust similarity search for hybrid queries with structured and unstructured constraints. In: Hasan MA, Xiong L (eds) Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022, ACM, pp 4580–4584, DOI 10.1145/3511808.3557610, URL https://doi.org/10.1145/3511808.3557610
98. Xu X, Li C, Wang Y, Xia Y (2020) Multiattribute approximate nearest neighbor search based on navigable small world graph. Concurr Comput Pract Exp 32(24), DOI 10.1002/CPE.5970, URL https://doi.org/10.1002/cpe.5970

99. Xu Y, Gao J, Gou Y, Long C, Jensen CS (2024) irangegraph: Improvising range-dedicated graphs for range-filtering nearest neighbor search. CoRR abs/2409.02571, DOI 10.48550/ARXIV.2409.02571, URL https://doi.org/10.48550/arXiv.2409.02571, 2409.02571
100. Yang W, Li T, Fang G, Wei H (2020) PASE: postgresql ultra-high-dimensional approximate nearest neighbor search extension. In: Maier D, Pottinger R, Doan A, Tan W, Alawini A, Ngo HQ (eds) Proceedings of the 2020 International Conference on Management of Data, SIGMOD Conference 2020, online conference [Portland, OR, USA], June 14-19, 2020, ACM, pp 2241–2253, DOI 10.1145/3318464.3386131, URL https://doi.org/10.1145/3318464.3386131
101. Zhang J, Ma R, Song T, Hua Y, Xue Z, Guan C, Guan H (2021) Hierarchical satellite system graph for approximate nearest neighbor search on big data. Trans Data Sci 2(4):32:1–32:15, DOI 10.1145/3488377, URL https://doi.org/10.1145/3488377
102. Zhang Q, Xu S, Chen Q, Sui G, Xie J, Cai Z, Chen Y, He Y, Yang Y, Yang F, Yang M, Zhou L (2023) VBASE: unifying online vector similarity search and relational queries via relaxed monotonicity. In: Geambasu R, Nightingale E (eds) 17th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2023, Boston, MA, USA, July 10-12, 2023, USENIX Association, pp 377–395, URL https://www.usenix.org/conference/osdi23/presentation/zhang-qianxi
103. Zhao P, Zhang H, Yu Q, Wang Z, Geng Y, Fu F, Yang L, Zhang W, Cui B (2024) Retrieval-augmented generation for ai-generated content: A survey. CoRR abs/2402.19473, DOI 10.48550/ARXIV.2402.19473, URL https://doi.org/10.48550/arXiv.2402.19473, 2402.19473
104. Zhao S, Yang Y, Wang Z, He Z, Qiu L, Qiu L (2024) Retrieval augmented generation (RAG) and beyond: A comprehensive survey on how to make your llms use external data more wisely. CoRR abs/2409.14924, DOI 10.48550/ARXIV.2409.14924, URL https://doi.org/10.48550/arXiv.2409.14924, 2409.14924
105. Zhao W, Tan S, Li P (2022) Constrained approximate similarity search on proximity graph. CoRR abs/2210.14958, DOI 10.48550/ARXIV.2210.14958, URL https://doi.org/10.48550/arXiv.2210.14958, 2210.14958
106. Zoumpatianos K, Lou Y, Ileana I, Palpanas T, Gehrke J (2018) Generating data series query workloads. VLDB J 27(6):823–846, DOI 10.1007/S00778-018-0513-X, URL https://doi.org/10.1007/s00778-018-0513-x
107. Zuo C, Qiao M, Zhou W, Li F, Deng D (2024) Serf: Segment graph for range-filtering approximate nearest neighbor search. Proc ACM Manag Data 2(1):69:1–69:26, DOI 10.1145/3639324, URL https://doi.org/10.1145/3639324