



Revisiting Filtered ANN Benchmarks: A Hardness-Controlled Benchmark Generator for Realistic Evaluation

Mintaek Lim
Seoul National University
Seoul, South Korea
victorlim@snu.ac.kr

Dogeun Kim
Seoul National University
Seoul, South Korea
kdg6245@snu.ac.kr

Minwoo Kim
Seoul National University
Seoul, South Korea
danny5969@snu.ac.kr

Jaeyoung Do*
Seoul National University
Seoul, South Korea
jaeyoung.do@snu.ac.kr

ABSTRACT

Filtered approximate nearest neighbor (FANN) search must satisfy both vector similarity and structured predicates, yet evaluations remain brittle because real hybrid workloads are rarely shareable and existing benchmarks rely on ad-hoc synthetic or semi-real constructions. We argue that realism hinges on execution-driven query difficulty: failures in early filtering trigger over-fetching of additional candidates, shaping latency, throughput, and recall. Building on this insight, we propose α -Hardness, a query-level hardness metric that models the conditional execution chain via the over-fetch factor and extends naturally to strategy-conditioned settings. Across diverse datasets and hybrid strategies, α -Hardness exhibits strong monotonic alignment with empirical performance, while common proxies such as selectivity or attribute-vector correlation are frequently unstable or strategy-inconsistent. We further introduce HCBGen, a hardness-controlled benchmark generator that uses α -Hardness as an explicit control signal to synthesize workloads under coarse bias modes or to match a target hardness profile. Our experiments show that widely used benchmarks occupy a narrow, relatively easy portion of the hardness spectrum, masking robustness gaps that emerge under harder queries. Finally, we demonstrate that matching hardness distributions enables privacy-preserving proxy workloads that closely reproduce performance trends, bridging research benchmarks and real evaluation.

PVLDB Reference Format:

Mintaek Lim, Dogeun Kim, Minwoo Kim, and Jaeyoung Do. Revisiting Filtered ANN Benchmarks: A Hardness-Controlled Benchmark Generator for Realistic Evaluation. PVLDB, 19(10): 2763 - 2776, 2026.
doi:10.14778/3828612.3828630

PVLDB Artifact Availability:

The source code, data, and/or other artifacts have been made available at https://github.com/AIDASLab/FANN_Benchmark.git.

1 INTRODUCTION

Vector search [23, 25, 33] has become a first-class primitive in modern data management, powering retrieval-augmented generation (RAG) [27], recommendation engines[9, 11, 35], e-commerce search[28, 61], and multi-modal analytics[32, 38]. In production, however, similarity alone is insufficient: queries must also enforce

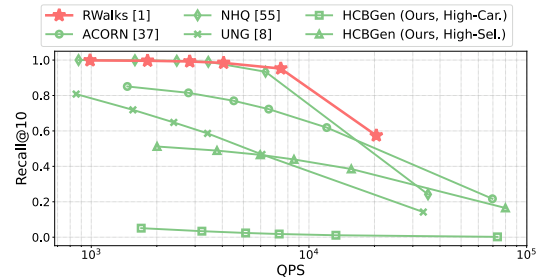


Figure 1: Recall-QPS trade-off curves of the RWalks index evaluated on six synthetic hybrid workloads built from the same vector-only dataset under different label-synthesis policies (four adopted from prior studies and two generated by our hardness-controlled framework: High Cardinality and Highly Selective). Although the workload configuration used in the RWalks paper (red) achieves near-perfect Recall@10 at high throughput, the other workload designs induce substantially different trade-offs, demonstrating that hybrid-index performance can be highly sensitive to workload construction, even with the underlying vector corpus and index fixed.

structured predicates such as category, time, or product attributes[1, 13, 42, 55, 57] that are non-negotiable for correctness and safety. For example, in ANN-based scholarly search systems, users issue queries in natural language, which are then embedded to retrieve relevant papers based on vector similarity. However, users may also require results restricted to specific venues or publication years. In such cases, the query must include structured predicates in addition to the embeddings, such as "year = 2026 and venue = VLDB". Without these constraints, the system may return semantically relevant but out-of-scope papers. This requires the system to index data containing both structured attributes and unstructured embeddings, and to optimize retrieval so that the returned points satisfy the filter predicates while remaining closest to the query embedding.

While algorithmic progress has been rapid, the field faces a quieter but consequential bottleneck: we still lack a reliable and practical way to benchmark hybrid indexing methods [20, 45]. Reported results often fail to transfer across datasets, query patterns, and deployment contexts, making it difficult for practitioners to trust which method to adopt and for researchers to draw durable conclusions [20, 55]. The root cause is not that hybrid queries are uncommon, but that real-world hybrid workloads are rarely shareable due to privacy constraints [4, 5], proprietary concerns [11, 14, 19], and limited public coverage [1]. As a result, most studies evaluate hybrid indices using synthetic or semi-real workloads, typically

*Corresponding author.

This work is licensed under the Creative Commons BY-NC-ND 4.0 International License. Visit <https://creativecommons.org/licenses/by-nc-nd/4.0/> to view a copy of this license. For any use beyond those covered by this license, obtain permission by emailing info@vldb.org. Copyright is held by the owner/author(s). Publication rights licensed to the VLDB Endowment.

Proceedings of the VLDB Endowment, Vol. 19, No. 10 ISSN 2150-8097.
doi:10.14778/3828612.3828630

constructed by taking a vector-only dataset and synthesizing labels, attributes, and filtered queries [54, 55, 59].

Benchmarks therefore exist and are widely used, but they suffer from a fundamental reliability issue: a “hybrid workload” is not uniquely determined by the underlying vector dataset. Instead, it depends on benchmark design choices such as how attributes are assigned to vectors, how filters are composed, and how queries are generated, which are often implicit or inconsistent across studies [8, 37, 45]. Even when the base vectors are identical, seemingly minor design changes can substantially alter effective workload difficulty and thus the observed ranking of methods. Figure 1 makes this sensitivity concrete: RWalks [1], evaluated on multiple synthetic hybrid workloads built from the same vector-only dataset, SIFT1M [22], under different label-synthesis policies, exhibits widely different recall-QPS trade-offs, implying that without a principled way to characterize and control workload difficulty, hybrid benchmarking may inadvertently favor particular strategies. Because current synthetic benchmarks neither explicitly model nor sufficiently control query difficulty, such variability is often masked by aggregate reporting based on average-case metrics [8, 55]. In practice, however, real hybrid workloads span a wide range of difficulty, and a small fraction of hard queries can dominate tail latency and expose system-level failures [14, 31, 50, 56, 58]. Consequently, current benchmarking practices hinder reliable assessment and pose a practical barrier to the adoption of FANN techniques in commercial vector database systems [3, 7, 58].

This challenge stems from the absence of a principled notion of query-level hardness for hybrid search. Existing proxies such as selectivity and correlation [30, 37] capture only partial aspects of difficulty and cannot explain the large performance variance among queries with similar proxy values (Section 3.1). As a result, prior work mainly controls dataset properties such as cardinality, skew, correlation, and missingness [8, 37, 55], while query workloads are generated through simple heuristics such as random attribute combinations or selectivity targets [45].

To overcome these limitations, we propose α -Hardness, a principled definition of query-level hardness for hybrid search, and introduce a hardness-controlled benchmarking methodology in which the construction and evaluation of workloads are directly guided by hardness. Our key insight is that query difficulty arises from the execution process itself, rather than from isolated static properties. We model FANN execution as a conditional chain, capturing how vector retrieval, over-fetching, and conditional filtering interact and how computational cost accumulates along the execution path [54]. This execution-driven formulation yields a unified hardness metric that (i) closely aligns with empirical search performance and (ii) remains robust to previously unmodeled factors, since new influences naturally manifest as changes within the same execution structure rather than requiring ad-hoc proxy extensions.

Hardness is not only diagnostic; it enables a new benchmarking methodology by making workload difficulty explicit and controllable. Building on this model, we introduce a hardness-controlled benchmark generator (HCBGen) that treats hardness as an explicit control knob for workload construction. Instead of generating queries via ad-hoc rules (e.g., targeting selectivity alone or sampling random filter combinations) the generator systematically

synthesizes workloads with prescribed hardness profiles. In particular, it can (i) construct workloads spanning RANDOM-, Low-, and HIGH-hardness regimes for controlled evaluation, and (ii) approximate realistic workloads by matching a target hardness distribution. This design unlocks three practical capabilities: (1) **Fair comparison**: it evaluates methods under comparable difficulty regimes, reducing the risk that workload construction implicitly advantages a specific strategy. (2) **Systematic stress testing**: it deliberately instantiates high-hardness workloads to expose failure modes and tail-latency behavior, rather than only average-case performance. (3) **Privacy-preserving workload sharing**: it enables organizations to share hardness specifications (e.g., hardness distributions) in lieu of raw query logs or sensitive attributes, while still supporting reproducible and predictive performance characterization. Finally, our extensive experiments show that widely used synthetic hybrid benchmarks occupy only a narrow—and relatively easy—portion of the hardness spectrum, highlighting the need for hardness-controlled benchmarking to obtain results that generalize beyond favorable workloads. Our contributions are as follows:

- We define α -Hardness, an execution-driven metric for query-level difficulty in FANN that aligns with empirical performance.
- We revisit common hybrid benchmarking practices and show that many evaluations are biased toward easy queries, masking tail behavior and limiting generalizability.
- We present a hardness-controlled benchmark generator (HCBGen) that enables controlled workload construction, fair and index-agnostic benchmarking, stress testing under targeted hardness regimes, and privacy-preserving approximation of real workloads through hardness specifications.

2 PRELIMINARY

2.1 Filtered ANNs

FANN search extends semantic search by jointly enforcing vector similarity and structured filter constraints. In standard ANN search, a query retrieves nearest neighbors based only on vector distance. In contrast, FANN additionally imposes predicates over structured attributes such as categorical labels, numeric ranges, or other metadata constraints [37, 55].

Formally, let the base data be defined as

$$\mathcal{D} = \{(\mathbf{v}_i, \mathbf{a}_i) \mid i = 1, \dots, N\},$$

where $\mathbf{v}_i \in \mathbb{R}^d$ denotes the d dimensional embedding vector of data point i , and $\mathbf{a}_i$ represents its associated set of structured attributes. A FANN (hybrid) query q is specified by a vector predicate $q_v \in \mathbb{R}^d$ and a set of filter predicates q_f , which together define the similarity requirement and the desired attribute constraints of the query. The goal of FANN search is to retrieve the top- K nearest neighbors to q_v among only those data points satisfying the filter constraint q_f :

$$\text{FANN}(q_{\text{hybrid}}) = \arg \min_{\substack{S \subseteq \{i \mid q_f \subseteq \mathbf{a}_i\} \\ |S|=K}} \sum_{i \in S} d(q_v, \mathbf{v}_i).$$

where $d(\cdot, \cdot)$ denotes a distance function defined on the embedding space. The additional filter constraints beyond traditional ANN search make FANN fundamentally a two-dimensional retrieval problem, forcing the system to trade off which condition to prioritize

during pruning, and thus requiring the index to simultaneously handle unstructured vector similarity and structured evaluation.

2.2 Filtered ANN Approaches

With the growing attention in FANN search, a range of approaches have been proposed, from simple baselines to hybrid-native indexing methods. The two most common baselines are Post-Filtering and Pre-Filtering. Post-Filtering first retrieves candidates using only vector similarity to q_v and then applies the filter predicate. Since the fraction of filter-satisfying candidates is unknown, it typically relies on over-fetching with a vector-only index such as HNSW [33]. Pre-Filtering applies the predicate before vector search and considers only filter-satisfying points. While this avoids over-fetching and guarantees full recall, the absence of query-specific indices often reduces it to a brute-force scan over the filtered subset, limiting scalability and throughput. Beyond these baselines, hybrid-native indices tightly integrate vector similarity and filtering during search, including NHQ [55], ACORN [37], UNG [8], RWalks [1], and Filtered-DiskANN (FDANN) [18]. NHQ augments vector distance with attribute-based distance to enable joint navigation over vector and attribute spaces. ACORN densifies the HNSW graph at build time and restricts traversal to filter-satisfying nodes, approximating an oracle query-specific index. UNG groups data points by shared attribute values and builds a unified graph that models both intra-group and inter-group connectivity. RWalks propagates attribute information over the graph, reminiscent of graph neural network propagation [62], and uses the resulting attribute matrix to guide search toward valid candidates. FDANN starts traversal from label-aware entry points and explores only filter-satisfying nodes during search.

3 QUERY-LEVEL HARDNESS MODELING

FANN (hybrid) query performance variance is dominated not only by the dataset and index, but also by the *execution dynamics* induced by enforcing both constraints. We therefore model *query-level hardness* as an execution-grounded quantity that predicts how much *work* a system must perform to return K valid neighbors. This section (i) explains why conventional proxy measures fail to model query hardness, (ii) proposes an execution-grounded hardness model, and (iii) extends the model to be strategy-conditioned.

3.1 Why Proxy-Based Hardness Fails

Unlike relational database systems [7, 21, 26, 51], query-level hardness in FANN search has been relatively under explored [30, 63]. Prior work therefore relies on two intuitive proxy measurements for hybrid query difficulty: selectivity, the fraction of base points satisfying the filter, and correlation, the degree to which filter-satisfying points align with the query vector’s neighborhood [30, 37]. In general, lower selectivity tends to increase difficulty because an index must examine more candidates to obtain K valid results, degrading throughput and/or recall [1]. Likewise, higher correlation typically makes queries easier: when filter-satisfying points are spatially clustered near the query vector, graph-based indices are more likely to reach valid targets through graph paths, improving recall while shrinking the explored search space and increasing throughput.

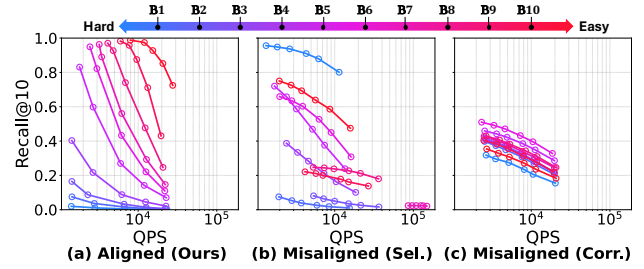


Figure 2: Success and failure cases of query ordering for recall-QPS trade-offs. For each proxy, queries are sorted by the estimated score and split into 10 equal-sized percentile bins (B1=hardest 10% → B10=easiest 10%). We plot recall-QPS curves for each bin on a synthetic SIFT1M-based benchmark using RWalks: (a) Successfully aligned (Ours), (b) Misaligned (Selectivity), and (c) Misaligned (Correlation).

However, selectivity and correlation only partially describe the *execution* of FANN. Both are one dimensional, static, whereas the actual difficulty is shaped by execution dynamics (i.e., how candidate retrieval, over-fetching, and filtering interact until K valid neighbors are obtained.) Consequently, the same global selectivity can induce very different *local* availability of filter-satisfying points around the query, and correlation does not directly determine how many candidates must be explored before K valid results are found. As a result, the ordering implied by selectivity or correlation can differ from the ordering induced by empirical recall-QPS trade-offs.

Figure 2 shows representative failure cases of this misalignment. On SIFT1M [22], we synthesize a hybrid workload with three categorical attributes (cardinality 6 each) and 10,000 queries, following Section 3.4. For each proxy, we sort queries, split them into 10 bins, and evaluate each bin with RWalks [1]. Lower bin indices indicate lower proxy values, interpreted as harder queries, while curves closer to the upper-right indicate better recall-QPS trade-offs. Figure 2(a), sorted by α -Hardness, serves as a reference where bin-wise performance is consistently ordered. In contrast, Figures 2(b) and (c) show that neither selectivity nor correlation produces an ordering aligned with actual performance. Thus, query hardness cannot be reliably captured by static, measurement-based proxies alone.

More broadly, defining a single scalar hardness by composing multiple measurements has two fundamental limitations. First, such measurements provide only indirect signals and do not explicitly model search execution, so their alignment with performance is not guaranteed. Second, they are inherently incomplete: newly identified hardness-related factors require revising both the definition and the measurement set. This motivates an execution-grounded hardness definition anchored in the search process rather than ad-hoc proxy statistics.

3.2 Execution-Grounded Hardness via Conditional Over-Fetching

To overcome the limitations of measurement-based proxies, we define query-level hardness by explicitly modeling FANN execution as a *conditional chain*. The key challenge in hybrid search is that the system must jointly satisfy vector similarity and a structured

predicate; when early-stage filtering fails to produce enough valid results, the search must *over-fetch* and continue exploring, a behavior commonly observed in practical systems (e.g., post-filtering pipelines) [54, 55]. Our formulation captures this execution dynamic through an over-fetch factor, yielding (i) stronger alignment with empirical performance and (ii) a structurally complete definition whose functional form does not require revision when new hardness-related phenomena are discovered.

Query and execution setting. A hybrid query is $q = (q_v, q_f)$ where q_v is the query vector and q_f is a set of structured predicates. Let K be the target number of returned neighbors. Let V_0 denote the full base data and $V_f \subseteq V_0$ the subset satisfying q_f .

Key idea. Hybrid query difficulty is governed by *how many vector-ranked candidates must be examined before execution outputs K filter-satisfying results*. To make this execution-grounded notion concrete, we start from a canonical execution chain in which over-fetching is explicit (post-filtering), and then extend the model to be strategy-conditioned in Section 3.3.

3.2.1 A canonical conditional execution chain. We instantiate query-level hardness by tracing a simple yet representative *post-filtering* execution chain: (1) retrieve candidates according to vector similarity; (2) scan candidates and keep only those satisfying q_f until K valid neighbors are obtained. This chain is methodological rather than restrictive: it is the simplest setting in which over-fetching is explicit, making it a clean basis for defining and estimating hardness, and it matches the vector-centric execution behavior studied throughout this section.

Definition 3.1 (Over-fetch factor). For a hybrid query $q = (q_v, q_f)$ and target K , the over-fetch factor $\alpha(q; K)$ is the number of vector-ranked candidates that must be examined so that the scan phase outputs K results satisfying q_f .

3.2.2 α -Hardness as conditional execution cost. To account for the conditional dependence between stages, we define α -Hardness using a multiplicative execution-cost model. The key intuition is that over-fetching not only increases predicate-checking work, but also amplifies the candidate retrieval effort required to maintain a fixed operating point.

Let $H_{\text{fetch}}(q_v | m)$ denote the cost of generating the top- m vector-ranked candidates for q_v from V_0 . Let $H_{\text{scan}}(q_f | q_v, m, K)$ denote the cost of scanning these m candidates and extracting K filter-satisfying results. Then the α -Hardness of q is defined as

$$H_\alpha(q|K) \triangleq H_{\text{fetch}}(q_v | \alpha(q; K)) \times H_{\text{scan}}(q_f | q_v, \alpha(q; K), K). \quad (1)$$

Under post-filtering, predicate checking incurs an (approximately) constant per-candidate cost, so scan hardness scales linearly with the number of examined candidates:

$$H_{\text{scan}}(q_f | q_v, m, K) = C_{\text{scan}} \cdot m. \quad (2)$$

Plugging $m = \alpha(q; K)$ into Eq. (2) and substituting into Eq. (1) yields

$$H_\alpha(q | K) \propto H_{\text{fetch}}(q_v | \alpha(q; K)) \cdot \alpha(q; K). \quad (3)$$

Monotonicity and alignment with performance. Hardness should increase monotonically with chain cost: larger α raises scanning cost and drives retrieval deeper, increasing $H_{\text{fetch}}(q_v | \alpha)$. Thus, higher H_α implies lower throughput and/or recall under a fixed budget. Equivalently, on a given dataset $\mathcal{D}$, H_α is designed to be

positively associated with $\left(\frac{1}{\text{Recall}} \cdot \frac{1}{\text{QPS}}\right)$, as validated in Section 3.4.

Reducing H_{fetch} to vector-only difficulty. Conditioned on the required depth $m = \alpha(q; K)$, the fetch component depends only on the vector predicate, reducing to vector search up to depth m . Thus, α -Hardness generalizes Steiner-Hardness [56] from vector-only to hybrid search: H_{fetch} captures the Steiner-Hardness of the full graph at depth $\alpha(q; K)$, while the overall formulation also accounts for filter-induced over-fetching. Since $\alpha(q; K)$ is not known *a priori*, we estimate it with $\hat{\alpha}(q; K)$ and substitute $\hat{\alpha}(q; K)$ into H_α .

3.2.3 Estimating $\hat{\alpha}(q; K)$ via selectivity and local availability. Semantically, $\hat{\alpha}(q; K)$ represents how many vectors must be traversed around the query vector q_v in order to obtain K filter-satisfying results. A natural first signal for this quantity is the *global selectivity* of the filter, $s = |V_f|/|V_0|$. When this fraction is small, valid results are rare in the dataset as a whole, and $\hat{\alpha}$ is therefore likely to increase. However, global selectivity alone is insufficient because it ignores how V_f is distributed around q_v . Vectors and labels can be correlated; for example, papers from a specific year may cluster around certain topics in the embedding space. As a result, even when a filter is globally selective, the filter-satisfying vectors may still be concentrated near the query, in which case the effective over-fetching required to collect K valid results can remain small.

To capture this effect, we estimate the over-fetching needed to obtain K valid results using both *global selectivity* and a *local availability (density) correction*:

$$\hat{\alpha}(q; K) \triangleq \frac{1}{s} \cdot \frac{1}{\rho(q)} \cdot K, \quad (4)$$

where $\rho(q)$ quantifies how much denser filter-satisfying points are around q_v compared to the full data.

To operationalize local availability, we use a query-local density proxy based on the ratio of K -NN distances in V_0 and V_f :

$$\hat{\alpha}(q; K) = \frac{|V_0|}{|V_f|} \cdot \frac{d(q_v, K\text{-NN} | V_f)}{d(q_v, K\text{-NN} | V_0)} \cdot K, \quad (5)$$

where $d(q_v, K\text{-NN} | V)$ denotes the distance from q_v to its K -th nearest neighbor within set V . Eq. (5) increases $\hat{\alpha}$ when (i) the filter is globally selective (small $|V_f|$) and/or (ii) valid points are locally sparse near the query (large $d(\cdot | V_f)$ relative to $d(\cdot | V_0)$).

Finally, substituting $\hat{\alpha}(q; K)$ into Eq. (3) yields a practical hardness score that can be computed *without running* the target hybrid index, while remaining tightly coupled to the execution chain. We empirically validate that ordering queries by H_α aligns well with recall-QPS trends (Section 3.4).

Completeness and extensibility. This formulation supports a *structurally complete* notion of hybrid query hardness: H_α is determined by the execution chain and its cost components, rather than by an open-ended set of ad-hoc proxies. Thus, new hardness-related phenomena can be incorporated by refining estimators of the existing chain variables, most notably $\hat{\alpha}(q; K)$, within the same framework. Our results in Section 3.4 show that this estimation already captures the dominant sources of over-fetching in practice and aligns well with empirical performance.

3.2.4 Extending the Execution Chain to Complex Predicates. Eq. (2) is sufficient for the categorical-predicate regime targeted by existing

FANN indices (e.g., UNG, RWalk, NHQ, and FDANN), where treating per-candidate predicate-check cost as approximately constant is appropriate. However, real applications may involve more complex predicates supported by systems such as pgvector [24], including polygon and regex matching. To accommodate future indices that may support such queries, we show that Eq. (2) can be naturally extended to cases where predicate-evaluation cost varies substantially across queries. The framework handles this by refining the scan-cost term while preserving the hardness formulation:

$$H_{\text{scan}}(q_f | q_v, \hat{\alpha}(q; K), K) = \hat{\alpha}(q; K) \cdot c_{\text{scan}}(q_f), \quad (6)$$

where $c_{\text{scan}}(q_f)$ is the average per-candidate predicate-evaluation cost. A natural refinement is to write

$$c_{\text{scan}}(q_f) = c(q_f) \cdot \phi(q_f), \quad (7)$$

where $c(q_f)$ is a predicate-type-specific unit cost, calibrated empirically [44], and $\phi(q_f)$ captures predicate complexity in terms of required unit operations [17]. For example, $\phi(q_f)$ may correspond to the number of vertices in a polygon predicate [36] or the number of wildcards in a regular-expression predicate [12, 49]. For simple categorical predicates, $c_{\text{scan}}(q_f)$ is approximately constant, recovering Eq. (2). This natural extension preserves the main advantage of the execution-grounded framework: richer hardness-related phenomena can be incorporated by refining existing chain variables, rather than by introducing an unrelated proxy. Experimental validation of this extension is provided in Section 3.4.4.

3.3 Strategy-Conditioned Hardness Estimation

Although H_α is defined by modeling the post-filtering execution chain from a vector-centric viewpoint, existing FANN approaches are not limited to this family. As discussed in prior surveys on FANN indices [30], existing methods can be broadly categorized into *vector-centric* and *filter-centric* pruning families¹. Filter-centric strategies do not follow the post-filtering chain; instead, they first restrict the search space using the filter predicate and then perform vector search within the filtered subset. Because hybrid-query difficulty is defined through the execution chain, it is inherently *strategy-dependent*: different indices enforce vector and filter constraints in different orders, leading to different dominant cost drivers. Accordingly, we denote the resulting score as $H(q | \text{strategy})$.

Pre-filtering execution chain. For filter-centric approaches such as UNG, FANN execution can be modeled as first inducing a filter-satisfying subgraph from the full graph index, and then performing vector search within that subgraph. Let G_0 denote the original graph over the full dataset V_0 , and let G_f denote the induced subgraph over V_f . The corresponding execution chain is

$$H_{\text{PreFiltering}}(q | K) \triangleq H_{\text{restrict}}(q_f | G_0) \cdot H_{\text{search}}(q_v | G_f),$$

where H_{restrict} denotes the cost of identifying the filter-satisfying search space and H_{search} the cost of vector search within G_f . This hardness can be interpreted as an execution-based approximation to the Steiner-Hardness [56] of the filter-induced subgraph G_f , extending the same perspective in Section 3.2.3 to the filter-centric

¹Although indices such as UNG and ACORN incorporate both filter and vector predicates through hybrid pruning mechanisms, they can still be classified into one of the two families according to the dominant pruning behavior. This categorization is also consistently supported by our experimental results.

setting. Since the restriction step is relatively stable for a fixed index, and search over G_f degrades toward a linear scan as selectivity decreases, the dominant cost factor is the size of the filtered vectors:

$$H_{\text{PreFiltering}}(q | K) \propto H_{\text{search}}(q_v | G_f) \propto |V_f|.$$

By rearranging Eq. (4), we obtain

$$|V_f| = \frac{|V_0| K \rho(q)}{\hat{\alpha}(q; K)}.$$

Pre-filtering already restricts search to the filter-satisfying subgraph G_f , so the dominant family-level cost is governed mainly by the size of that restricted search space, $|V_f|$, rather than by the local-density term $\rho(q)$. Since $\rho(q)$ governs how quickly post-filtering accumulates valid results during vector-centric traversal—it ceases to be a primary cost driver once search is already confined to V_f . With K and $|V_0|$ fixed for a given benchmark setting, the dominant filter-centric cost follows the opposite monotone trend from vector-centric cost and decreases as the effective $\hat{\alpha}$ increases.

Why a simple inversion is sufficient. The above derivation establishes a monotone duality between the two execution families: queries that are hard for vector-centric execution tend to be easy for filter-centric execution, and vice versa. Rather than introducing separate hardness models per family, we therefore apply a monotone inversion to reuse the same execution-grounded score:

$$H(q | \text{strategy}) \triangleq \begin{cases} H_\alpha(q), & \text{if the strategy is vector-centric,} \\ \frac{1}{H_\alpha(q)}, & \text{if the strategy is filter-centric.} \end{cases}$$

This transformation is intentionally conservative: it only assumes that the two strategy families exhibit opposite monotone trends with respect to over-fetching-driven difficulty (as observed empirically), without committing to a specific internal implementation. As shown in Section 3.4, although this minimal strategy-aware transformation is not intended as an exact cost model for the filter-centric implementations, it is sufficient to achieve strong alignment with empirical performance across both families.

3.4 Hardness Validation

We empirically validate whether the proposed α -Hardness induces a query ordering that is consistent with observed *recall-throughput* trade-offs in FANN search. Our goal is not to fit a particular index, but to test whether the hardness definition remains predictive across (i) diverse dataset/workload characteristics and (ii) fundamentally different hybrid execution strategies.

3.4.1 Dataset & Workload Suite. Following prior hybrid benchmarking practice [8, 37, 55], we construct a suite of *synthetic* hybrid datasets by augmenting standard vector-only benchmarks with synthesized categorical labels, while systematically controlling the core label/workload factors: the source base vectors, number of attributes, attribute cardinalities, value distributions, per-attribute missing probabilities, and *data-label correlation (DLC)* between base vectors and base labels².

Query workloads are generated using the same synthesis policy used for the base data to ensure structural compatibility. All

²DLC is distinct from the query-vector correlation (QVC) used as a hardness proxy in Section 3.1; DLC characterizes correlation in the *data/label generation* process, whereas QVC is measured at the *query level*.

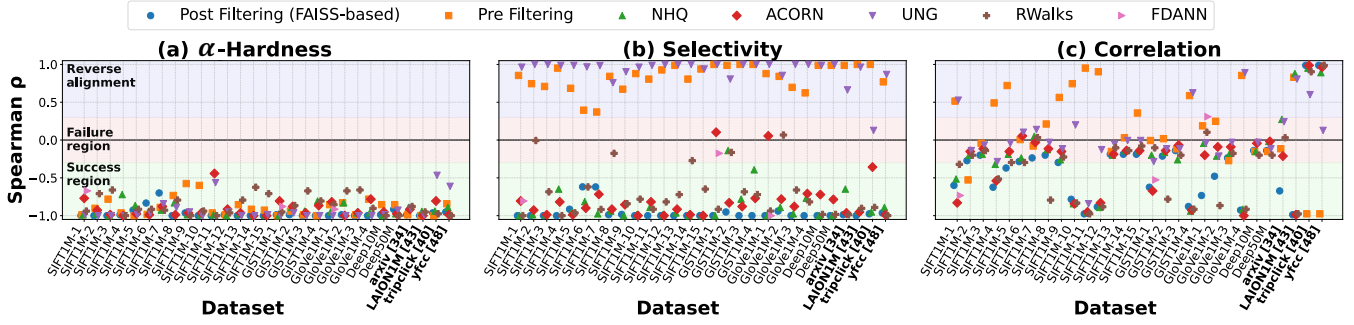


Figure 3: Spearman rank correlation between query hardness estimates (α -Hardness (ours) or baseline proxies: selectivity and correlation) and search performance across 29 datasets (25 synthetic and 4 semi-real), evaluated under seven hybrid query processing strategies. Semi-real workloads are constructed by taking a vector-only dataset and synthesizing labels, attributes, and filtered queries. Synthetic datasets are identified using the configuration IDs (Table 1). Values closer to -1 indicate stronger alignment (harder queries $\rightarrow$ worse performance), whereas values in $[0.3, 1]$ indicate unintended reverse alignment.

synthetic configurations are summarized in Table 1. In addition, to verify that α -Hardness is not an artifact of fully synthetic construction, we include four *semi-real* datasets whose base data—base vectors paired with corresponding labels—are drawn from real-world corpora, and whose query workloads are sampled or synthetically generated from the base data [34, 40, 43, 48].

We evaluate alignment under seven representative FANN processing strategies: FAISS[23]-based Post-Filtering and Pre-Filtering, and five hybrid-native indices (NHQ [55], ACORN [37], UNG [8], RWalks [1]), and FDANN [18]³. For each method, we sweep its parameters to obtain recall-QPS trade-offs, following the recommended settings in the corresponding original papers. To further examine whether the observed behavior remains robust when the dataset size exceeds CPU cache capacity, we additionally include the Deep10M and Deep50M datasets, which are subsets of Deep1B [60]. For additional experimental details refer to Section 5.1.

3.4.2 Alignment Protocol and Metrics. FANN search performance is commonly summarized by recall-QPS trade-offs under varying parameters [23, 45]. Accordingly, a hardness measure is *well-aligned* if sorting queries by hardness yields groups whose trade-off curves degrade monotonically. We use Recall@10 and refer to it simply as recall throughout this section. For each dataset and strategy, we compute $H(q \mid \text{strategy})$ (Section 3.3), sort queries by increasing hardness, and partition them into $B=10$ equal-sized batches. This operationalizes the ordering soundness requirement: later batches should contain queries that are no easier than earlier ones⁴. For each batch, we evaluate the target strategy across operating points, yielding one recall-QPS trade-off curve. To summarize each curve while preserving the trade-off, we compute a Pareto-optimal score by extracting the Pareto frontier, averaging recall and QPS over frontier points, and taking their product [16]. Higher scores indicate more efficient query processing.

Let $\bar{H}_b$ be the average hardness of batch b , and let SCORE_b be the Pareto-optimal score of that batch. We quantify alignment by the

³Since FDANN can handle only queries with a single filter attribute, we report results only for SIFT1M-1, GIST1M-1, and GloVe1M-1.

⁴We use $B=10$ to balance resolution and stability; our conclusions are unchanged for nearby choices such as 5 or 20.

Table 1: Parameter configurations for synthetic datasets. The configuration ID encodes the source base vectors (prefix). For brevity, a single value indicates a setting applied uniformly to all attributes; mixed settings list per-attribute values.

ID	#Attr	Cardi.	Distribution	Missing Prob.	DLC
SIFT1M-1	1	12	Zipf	0.5	0.0
SIFT1M-2	3	6			
SIFT1M-3	3	12			
SIFT1M-4	12	1			
SIFT1M-5	12	3			
SIFT1M-6	12	6			
SIFT1M-7	12	12			
SIFT1M-8	3	12	Random	0.5	0.0
SIFT1M-9	12	3			
SIFT1M-10	3	12	Zipf	0.5	0.5
SIFT1M-11					1.0
SIFT1M-12					mix
SIFT1M-13	3	12	Zipf	0.0	0.0
SIFT1M-14				0.8	
SIFT1M-15				mix	
GIST1M-1	1	12	Zipf	0.5	0.0
GIST1M-2	3	12	Random	0.5	0.0
GIST1M-3	3	12	Zipf	0.5	0.0
GIST1M-4	3	12	Zipf	0.0	0.5
GloVe1M-1	1	12	Zipf	0.5	0.0
GloVe1M-2	3	12	Random	0.5	0.0
GloVe1M-3	3	12	Zipf	0.5	0.0
GloVe1M-4	3	12	Zipf	0.0	0.5
Deep10M	3	12	Zipf	0.5	0.0
Deep50M	3	12	Zipf	0.5	0.0

Spearman rank correlation between $\{\bar{H}_b\}_{b=1}^B$ and $\{\text{SCORE}_b\}_{b=1}^B$ [47]. Because higher hardness should correspond to lower performance, stronger alignment is reflected by Spearman’s ρ closer to -1 . For conventional hardness proxies (selectivity and correlation), we negate them before batching so that larger values consistently indicate higher difficulty, matching the direction of $H(\cdot)$ ⁵.

3.4.3 Results and Takeaways. Figure 3 reports Spearman’s ρ across datasets and strategies. Across diverse dataset/strategy combinations, the proposed hardness exhibits strong monotonic alignment with empirical search performance (typically $\rho \leq -0.7$), whereas

⁵We intentionally do *not* choose proxy directions per strategy, since that would leak strategy knowledge into the baseline and artificially improve its alignment.

Table 2: Alignment comparison between Extended and Original α -Hardness across predicate workloads. Predicate complexity is grouped by calibrated unit scan cost $c(q_f)$. Improvement is measured as $\Delta|\rho| = |\rho_{\text{ADAPTIVE}}| - |\rho_{\text{ORIGINAL}}|$.

Complexity	Dataset	$c(q_f)$	ρ (Original)	ρ (Extended)	$\Delta \rho $
Simple	categorical	1.56	-0.99	-0.99	+0.00
	range	2.18	-0.96	-0.92	-0.04
Medium	p-simple	3.75	-0.82	-0.98	+0.16
	p-medium	3.75	-0.62	-1.00	+0.38
	p-complex	3.75	-0.59	-1.00	+0.41
	set-overlap	10.76	-0.90	-1.00	+0.10
Complex	r-simple	287.92	-0.89	-0.99	+0.10
	r-complex	287.92	-0.66	-0.95	+0.29

proxy measurements are frequently near zero or unstable, indicating that they do not reliably order query difficulty.

The baseline *query–vector correlation* (QVC) becomes informative only when strong positive *data–label correlation* (DLC) is explicitly enforced in the synthetic construction (e.g., SIFT1M-10, -11, -12, GIST1M-4, and GloVe1M-4), a condition that is neither universal nor guaranteed in realistic workloads; outside such regimes, QVC provides limited explanatory power.

Selectivity exhibits opposite signs of ρ across strategy families because it is intrinsically entangled with the pruning order, leading to reversed alignment with the intended ordering under certain strategies. For vector-centric strategies, lower selectivity typically makes queries harder by increasing over-fetching and predicate checks, whereas for filter-centric strategies it can make queries easier by shrinking the pre-filtered search space and reducing downstream vector traversal. Without explicit knowledge of the execution family, which is typically unavailable to a benchmark designer, selectivity cannot even determine a *consistent direction* of difficulty, and in many settings yields weak alignment.

If we treat $\rho \in [-0.3, 0.3]$ as an alignment failure case [10], α -Hardness incurs no such failures in any of our evaluated settings, including scaled-out regimes, whereas selectivity fails in a non-trivial fraction of cases. This gap arises because selectivity isolates only the filter predicate while ignoring the vector predicate and their joint interaction along the execution chain, thereby violating the validity and ordering soundness criteria discussed in Section 3.2.

3.4.4 Validation on Complex Predicates. We further validate the extended formulation (Eq. (6)) on complex-predicate workloads built on SIFT100K, a randomly sampled subset of SIFT1M. The evaluated predicates include range filters, polygon matching with 4–12 (p-simple), 30–70 (p-medium), or 150–250 (p-complex) vertices, regex matching over strings of length 100 with either no wildcard (r-simple) or 1–8 wildcards (r-complex), and set-overlap predicates with a vocabulary pool of size 100. Since no existing hybrid index directly supports this predicate set, we use HNSW-based post-filtering as the representative execution strategy.

Table 2 shows that the original α -Hardness aligns well when predicate complexity is low, but its ordering quality degrades as per-candidate validation cost becomes more query-dependent, especially for polygon and regex workloads. By contrast, the extended formulation remains aligned across all workloads, with $\rho < -0.9$

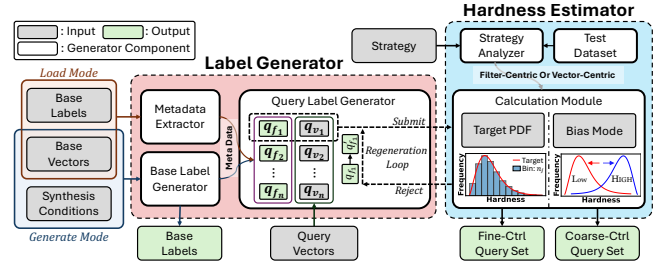


Figure 4: Overview of the hardness-controlled benchmark generator (HCBGen). The generator supports both Load and Generate modes for constructing the base data, and controls query-level hardness using either a target hardness profile or coarse-grained bias modes (HIGH, Low, RANDOM).

throughout, and yields the largest gains on structurally complex predicates such as p-medium, p-complex, and r-complex. These results show that α -Hardness extends naturally to compute-bound settings, where the dominant cost shifts from over-fetching to candidate scanning, without changing the underlying conditional chain.

4 BENCHMARK GENERATOR

Because real-world hybrid workloads are rarely shareable, FANN evaluations often rely on synthetic benchmarks, whose hidden design choices can bias rankings and obscure robustness. We present HCBGen, which uses α -Hardness to synthesize hybrid workloads with controllable, interpretable, and strategy-consistent difficulty. The generator supports (i) reproducible, hardness-neutral fixed benchmarks for *fair comparison*, (ii) *stress testing* by emphasizing hard queries, and (iii) *workload approximation* by matching a hardness profile that can be shared without exposing raw query logs.

4.1 Architecture

Figure 4 separates the generator into a *Label Generator* and a *Hardness Estimator*. The label generator proposes candidate queries, while the hardness estimator enforces hardness-aware acceptance through a regeneration loop. This separation allows hardness control to remain strategy-consistent across heterogeneous indices.

4.1.1 Label Generator. The label generator supports two base data modes—*Load* and *Generate*—because hybrid benchmarking arises in two fundamentally different settings. In one setting, practitioners have a labeled base data (i.e., base vectors with associated base labels), but the corresponding query logs cannot be shared; the community often has only public vector-only corpora, where structured base labels are unavailable and must be synthesized to support hybrid (vector+filter) benchmarking. The two modes trade off *semantic fidelity* and *controlled synthesis*: LOAD preserves the semantics and structure of a given corpus, whereas GENERATE creates a synthetic label space with explicit control over workload properties.

In Load mode, the generator takes base vectors together with their existing base labels and uses a *Metadata Extractor* to infer the structural metadata needed for query construction—such as attribute domains, cardinalities, value frequencies, and missingness

Algorithm 1 Coarse-Grained Hardness-Controlled Query Generation (HIGH/LOW/RANDOM).

Require: Base vectors V , base labels L , mode M , target size N , Budget B

Ensure: Query set Q with $|Q| = N$.

```
1: Sample an initial query pool and compute hardness values
2: if  $M \in \{\text{HIGH}, \text{LOW}\}$  then
3:   Set threshold  $\tau \leftarrow \text{perc}_{80}(H)$  for HIGH,  $\text{perc}_{20}(H)$  for LOW
4: end if
5: for each query vector  $q_o$  in the initial pool do
6:   Initialize candidate set  $C \leftarrow \emptyset$ 
7:   for  $i = 1$  to  $B$  do
8:     Sample filter  $q_f$  and form  $q = (q_o, q_f)$ 
9:     if  $M = \text{RANDOM}$  or  $q$  satisfies  $\tau$  then
10:       $Q \leftarrow Q \cup \{q\}$ ; break
11:    end if
12:     $C \leftarrow C \cup \{q\}$ 
13:  end for
14:  if no query accepted then
15:     $Q \leftarrow Q \cup \{\arg \min_{q \in C} |H(q) - \tau|\}$ 
16:  end if
17: end for
18: return  $Q$ 
```

patterns. This metadata is forwarded to the *Query Label Generator*, which samples query labels that are schema-compatible with the loaded base labels. As a result, generated predicates respect the original label vocabulary and distributional structure, enabling realistic query synthesis when the original query workload is unavailable.

In Generate mode, the generator takes only base vectors and a set of label synthesis conditions, and uses a *Base Label Generator* to synthesize base labels from scratch. The synthesis conditions define the label space and its statistical properties, and also include an explicit control for vector-label alignment. Concretely, our default synthesizer partitions the vector space into clusters and assigns attribute values in a DLC-controlled manner so that vectors within the same cluster share values with a tunable strength. This is important because vector-label alignment strongly influences the effectiveness of pruning behaviors in hybrid indices, and thus determines the regime in which a benchmark evaluates robustness versus best-case performance. The synthesized labels are then summarized into the same type of metadata as in Load, ensuring that subsequent query generation remains structurally consistent.

Given the extracted or synthesized metadata and the query vectors, the *Query Label Generator* samples candidate predicates and forms hybrid queries. Since most hybrid-native indices primarily support categorical predicates, it generates predicates composed of one or more label = value conditions. Because FANN assumes single-table data with vector embeddings, cross-table operations such as joins are out of scope. Candidates are then passed to the hardness estimator, and rejected queries are returned for regeneration, forming a hardness-controlled regeneration loop (Section 4.2). This design decouples label construction from hardness control: the label generator defines **what** predicates are valid and realistic, while the hardness estimator decides **which** valid queries are retained to satisfy the target hardness regime.

4.1.2 Hardness Estimator. Hardness-based control requires strategy awareness because different hybrid indices enforce vector and filter constraints in different orders (Section 3.3). Accordingly, the hardness estimator consists of a *Strategy Analyzer* and a *Hardness Calculation Module*. The strategy analyzer first classifies the target

Algorithm 2 Fine-Grained Hardness-Controlled Query Generation (MATCH-PDF) with Budgeted Closest-Fill Fallback.

Require: Base vectors V , base labels L , target hardness PDF $\mathcal{P}$, target size N , bin count B .

Ensure: Query set Q with $|Q| = N$.

```
1: Initialize  $Q \leftarrow \emptyset$ .
2: Discretize hardness into  $B$  disjoint intervals  $\{I_j\}_{j=1}^B$  according to  $\mathcal{P}$ ; assign target counts  $n_j$ .
3: Initialize candidate pools  $R_j \leftarrow$  empty list for all bins  $j$ .
4: while  $|Q| < N$  do
5:   Sample candidate query  $q = (q_o, q_f)$ ; compute hardness  $h \leftarrow H(q)$ .
6:   Find bin  $j$  such that  $h \in I_j$ .
7:   if  $n_j > 0$  then
8:      $Q \leftarrow Q \cup \{q\}$ ;  $n_j \leftarrow n_j - 1$ .
9:   end if
10:  for each bins  $j$  with  $n_j > 0$  do
11:    if  $h \notin I_j$  then store  $(\text{dist}(h, I_j), q_i)$  in  $R_j$ .
12:  end for
13:  if  $q$  is the last candidate query then
14:    for each bin  $j$  with  $n_j > 0$  do
15:      fill remaining slots with arg min-dist unused candidates from  $R_j$ .
16:    end for
17:  end if
18: end while
19: return  $Q$ .
```

strategy into a pruning family, which fixes the strategy-consistent interpretation of hardness. The hardness calculation module then computes the strategy-conditioned hardness score of each candidate query (Section 3.3) and applies the user-specified control rule to decide whether to keep the query or reject it.

To classify a strategy with minimal overhead, we maintain two fixed probe datasets that were chosen empirically to exhibit the most opposing characteristics across the two pruning families. Specifically, these probes are reduced versions of SIFT1M-1 and SIFT1M-7 from Table 1, where both the number of base vectors and the number of queries are downsampled by a factor of ten. Given a target strategy, the analyzer builds the index if needed, runs searches on both probes, and records the resulting recall-QPS trade-off curves. It then summarizes each curve using a single Pareto-optimal score computed by the same procedure described in Section 3.4.1, and classifies the strategy by comparing the two scores.

This estimation correctly classifies not only Post-Filtering and Pre-Filtering, but also all hybrid-native FANN strategies, including ACORN, UNG, NHQ, RWalks, and FDANN, in exact agreement with the theoretical categorization in [30]. This is also reflected in Figure 3, where the ordering induced by α -Hardness remains consistently negative across all strategies.

To handle future implementations that may not show a clear preference for either probe workload, we classify a strategy as MIXED when the two probe scores are too close to distinguish. MIXED is a robust fallback for ambiguous cases: rather than relying on a single family decision, the generator enforces hardness constraints under both interpretations, reducing vulnerability to borderline misclassification. This keeps probing a one-time, cacheable process while preventing classification uncertainty from affecting hardness control. Because the index analyzer uses small probe datasets, this step requires at most one minute.

⁶UNG synthesizes labels by treating each attribute-value pair as a single label and assigning 12 such labels under a Zipfian distribution. To replicate this, we use 12 attributes of cardinality 1 and implement the Zipfian skew through missing probabilities, yielding a functionally equivalent labeling policy.

Table 3: Label-synthesis configurations used in prior hybrid indexing studies, summarizing the number of attributes, attribute cardinalities, value distributions, missing probabilities, and data-label correlation (DLC).

Method	#Attr	Cardi.	Distribution	Missing Prob.	DLC
NHQ	3	(3, 2, 2)	Random	0	No
ACORN	1	(12)	Random	0	No
UNG ⁶	12	(1, 1, ..., 1)	–	Zipf	No
RWalks	6	(100, 20, 10, 5, 3, 2)	Random	0	No

4.2 Hardness-Controlled Query Generation

The generator supports two hardness-control families. The first provides coarse-grained modes for stress testing and hardness-neutral fixed benchmarks, while the second enables fine-grained profile matching for workload approximation. Both use the same accept/reject loop and differ only in the acceptance rule.

4.2.1 Coarse-Grained Bias Modes. The coarse-grained modes bias the workload toward hard or easy queries (Algorithm 1). Rather than using a fixed, absolute hardness threshold, which can vary widely across datasets and label constructions, we derive a cutoff adaptively from a small initial sample of generated queries. For example, HIGH uses a cutoff corresponding to the top portion of the sampled α -Hardness, while LOW uses a cutoff corresponding to the bottom portion. A candidate query is accepted if its α -Hardness falls into the desired region (hard for HIGH, easy for LOW); RANDOM disables hardness-based selection and accepts randomly generated queries. RANDOM performs no hardness-based filtering, producing a reproducible, strategy-independent fixed benchmark. Because it retains queries spanning a broad range of hardness values, it yields a benchmark with diverse query difficulty that can be used for fair comparison across hybrid index designs. To ensure termination and stable generation time, we cap the number of regeneration attempts per query. If no candidate meets the desired condition within this retry limit, we accept the candidate whose difficulty score is closest to the cutoff, which guarantees completion while still preserving a strong bias.

4.2.2 Fine-Grained Target-Profile Matching. Fine-grained control targets a user-specified hardness profile to approximate an unknown real workload using α -Hardness as a shareable signal (Algorithm 2). Concretely, the generator first partitions the difficulty range into a fixed number of intervals and assigns a target number of queries to each interval according to the desired profile. During generation, each sampled candidate query is scored and placed into the corresponding interval; it is accepted only if that interval still needs more queries, and otherwise it is rejected and regeneration continues. To keep the procedure practical, we enforce a finite sampling budget. If generation stalls because some intervals are extremely unlikely under the current base data, we trigger explicit fallback mechanisms, such as filling the remaining slots with the closest available candidates. We validate this behavior in Section 5.4 by comparing both the similarity of the resulting difficulty profiles and the downstream search behavior.

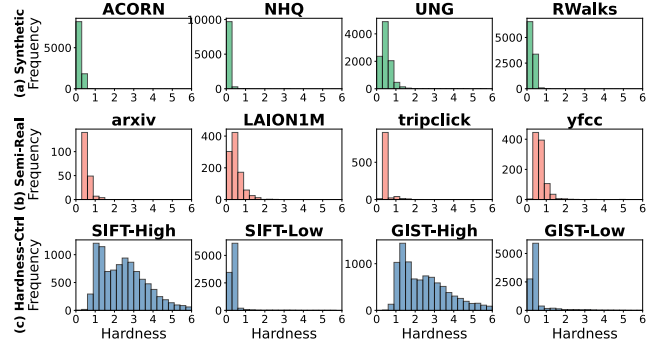


Figure 5: Comparison of α -Hardness distribution coverage across three workload classes: (a) synthetic workloads adopted from prior studies (SIFT1M-based), (b) semi-real workloads, and (c) hardness-controlled workloads generated by our framework.

5 EVALUATION

This section evaluates the proposed HCBGen from three perspectives: coverage, controllability, and workload approximation. We investigate the following research questions:

- **RQ1 (Hardness coverage):** How much of the hardness spectrum is covered by workloads used in prior FANN studies and by semi-real workloads?
- **RQ2 (Hardness controllability):** Can our generator reliably shift the workload toward easy vs. hard regimes under identical synthesis settings?
- **RQ3 (Robustness under controlled regimes):** How do representative strategies behave when evaluated across matched, systematically varied hardness regimes?
- **RQ4 (Hardness as a bridge):** If only a hardness profile can be shared, can the generator produce proxy workloads that induce similar performance trends?

5.1 Experimental Setup

We evaluate on three classes of hybrid workloads: (i) synthetic workloads adopted in prior hybrid indexing studies (NHQ, ACORN, UNG, and RWalks [1, 8, 37, 55]), (ii) four semi-real datasets, typically constructed by augmenting a vector-only dataset with synthesized labels, used in prior work (arxiv, LAION1M, tripclick, and yfcc [34, 40, 43, 48]), and (iii) four hardness-controlled benchmarks generated by our framework. We follow the original synthesis procedures and parameter recommendations to ensure that our comparisons isolate the effect of *workload design*, rather than retuning or re-implementing indices. In all cases, label synthesis uses predicates composed of categorical conditions of the form `label=value`, combined with conjunctions.

For performance benchmarking, we report recall-QPS trade-offs following the parameter sweeps recommended by the corresponding papers to ensure fair and representative evaluation. All experiments run on a single Amazon EC2 instance (c5a.8xlarge; 32 vCPUs @ 3.3 GHz; 64 GB RAM).

For synthetic workloads from prior studies, we fix the base vector dataset to SIFT1M [22] and vary only the label/query synthesis

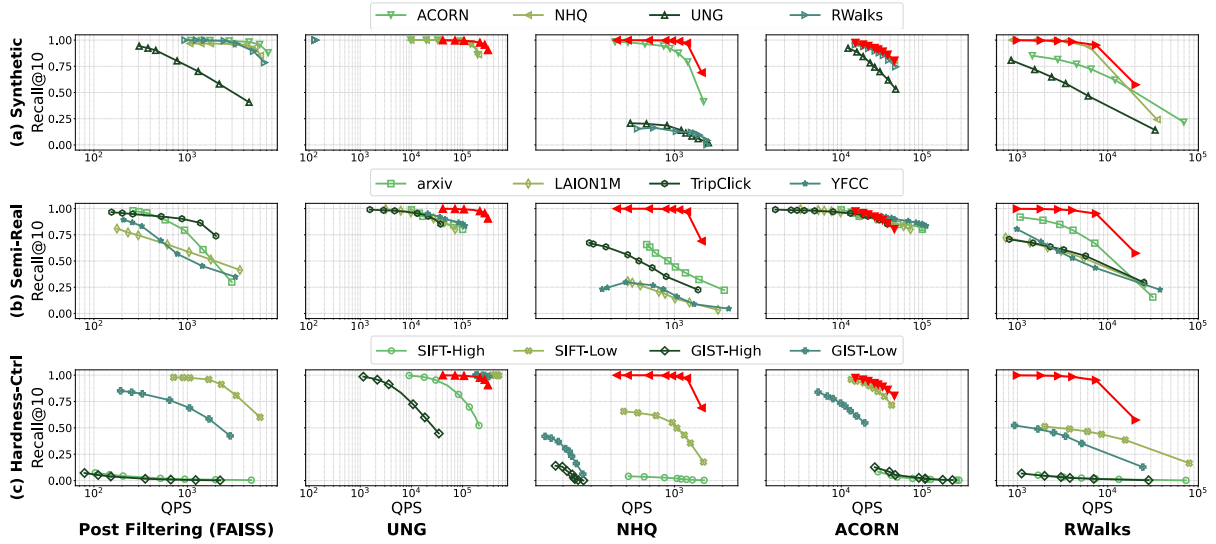


Figure 6: Recall-QPS trade-off curves for five hybrid query processing strategies (Post-Filtering and four hybrid-native indices) across 12 workloads spanning three workload families: (i) synthetic workloads adopted in prior hybrid indexing studies, (ii) semi-real workloads, and (iii) hardness-controlled workloads generated by our method. Red markers overlay the performance reported in each strategy’s original paper (when available), providing a reference anchor for cross-workload comparison.

policy, so that differences are attributable to workload construction. Label synthesis configurations are summarized in Table 3 and are adopted from the original papers. Query workloads follow the original generation procedures (e.g., single-predicate queries for RWalks and no missing-value constraint on NHQ query filters). The four semi-real benchmarks are widely used benchmarks in prior FANN studies [1, 8, 37]. They span diverse vector dimensionalities, and attribute configurations [45].

Using the proposed generator, we construct four representative hardness-controlled hybrid benchmarks. These benchmarks are based on two widely used vector datasets, SIFT1M [22] and GIST1M [22], whose vector dimensionalities are 128 and 960, respectively, chosen because SIFT1M has served as a standard baseline across prior hybrid indexing studies, whereas GIST1M offers a markedly higher-dimensional space that stresses vector-centric behavior. For each base data, we synthesize 3 attributes with a cardinality of 12 per attribute, following a Zipfian distribution with skewness parameter 1.5, a missing probability of 0.5, and 0.0 DLC (data-label correlation). To isolate the effect of hardness control, we generate two workload variants under identical synthesis settings using the coarse-grained Low and HIGH modes (Section 4.2). This yields **SIFT-Low**, **SIFT-High**, **GIST-Low**, and **GIST-High**.

5.2 Hardness Coverage and Control

We first examine how query hardness is distributed across different workloads and whether our generator enables systematic control.

5.2.1 Coverage Comparison (RQ1). Figure 5 compares α -Hardness distributions. Across these workloads, hardness is typically heavy-tailed: many queries fall into an easy regime with a smaller hard tail. Prior synthetic workloads, notably those used by ACORN and NHQ, are concentrated toward the easy end of the hardness spectrum,

Table 4: Indexing time and memory footprint of four methods on three different synthetic datasets.

Method	Indexing Time (seconds)			Memory Footprint (MB)		
	ACORN Syn. (Simple)	UNG Syn. (Medium)	RWalks Syn. (Complex)	ACORN Syn. (Simple)	UNG Syn. (Medium)	RWalks Syn. (Complex)
NHQ	56	40	23	684	695	689
ACORN	659	657	685	1202	1202	1202
UNG	38	40	281	695	693	731
RWalks	342	344	348	1110	1203	2073

and the semi-real workloads similarly occupy a narrow easy range with little mass in harder regimes. In contrast, our **SIFT-High** and **GIST-High** benchmarks span a substantially wider hardness range, including a visible fraction of queries with hardness > 1 and a hard tail extending beyond 6. This broader coverage is important because it enables stress-testing and robustness evaluation in challenging regimes that are weakly represented in commonly used benchmarks, reducing the risk that conclusions hinge on an implicitly “mild” workload slice.

5.2.2 Hardness-Aware Generation (RQ2). To verify controllability, we compare Low vs. HIGH workloads generated under identical synthesis settings. As shown in Figure 5(c), Low concentrates queries in the $[0, 1]$ hardness range, whereas HIGH shifts the mass sharply toward harder queries (over 99% of queries have hardness > 1). Crucially, this distributional shift translates into empirical behavior: Figure 6(c) shows that Low workloads yield consistently better recall-QPS trade-offs, while HIGH workloads shift curves toward lower recall under the same search budgets across all evaluated strategies. This validates that hardness is not merely a descriptive statistic: it is an actionable control signal that induces predictable differences in measured performance.

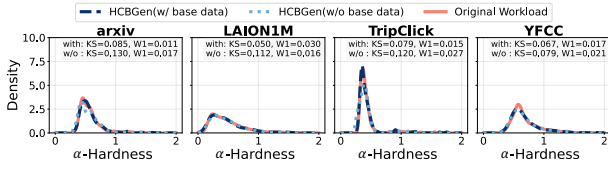


Figure 7: α -Hardness density on four semi-real datasets, comparing the original workload with two MATCH-PDF workloads generated by HCBGen to match the given distribution, with and without base data. Insets report KS and W1 distances to the original workload (lower is better).

5.3 Strategy Robustness (RQ3)

We next revisit the performance and indexing / maintenance overhead of representative hybrid query processing strategies under a common workload suite. A key challenge in interpreting the FANN literature is that each method is often evaluated on a workload designed with a method-specific label/query synthesis policy (Table 3), which complicates cross-paper comparison. Our goal here is not to claim any prior result is “wrong”; rather, we assess *sensitivity*: whether conclusions drawn under a narrow workload slice remain stable when the hardness regime is varied in a controlled way.

As shown in Figure 6, strategy rankings and trade-off shapes vary substantially across workloads. In particular, results reported in the original papers tend to align with easier regions of our workload space, suggesting that performance claims can be workload-sensitive and that robustness under harder regimes is not well captured by commonly used benchmarks. This sensitivity is most evident on HIGH workloads, where recall drops sharply for most strategies, often below 0.2, consistent with the known difficulty of extremely low-selectivity regimes (e.g., around or below 0.1%) in which hybrid indices struggle to find enough filter-satisfying candidates [1]. We also report indexing time and memory footprint in Table 4. These results are measured on the three synthetic datasets in Table 3, all built on the same SIFT1M base vectors. Since index construction is independent of the query workload, these costs are driven mainly by the base data label synthesis policy rather than by the α -Hardness. Overall, the table shows that indexing overhead is sensitive to label organization: some methods remain stable across datasets, whereas others vary despite sharing the same base vectors.

5.3.1 Strategy-Specific Behavior Analysis. While all methods degrade under hard workloads, the severity and underlying causes of degradation differ substantially across index designs.

Post Filtering relies exclusively on over-fetched vector candidates to satisfy filter predicates. Under high-hardness workloads, increasing the over-fetch factor has limited effect, as filter-satisfying candidates remain rare, leading to consistently low recall. As shown in Figures 5 and 6, higher α -Hardness directly translates to poorer search performance across datasets.

UNG is one of the most robust methods to dataset variation among those evaluated. Even extremely low-selectivity queries targeting rare labels, such as **SIFT-High** in Figure 6(c), can still maintain robust performance because sufficient graph connectivity is preserved, albeit at the cost of substantially higher index construction overhead. As shown in Table 4, this design requires UNG to preserve

connectivity for rare labels during indexing by partitioning the data into smaller groups, causing both indexing time and memory overhead to increase as the label distribution of the base data becomes more complex. These smaller groups can also improve search efficiency. However, in workloads such as RWalks with single-label filters, UNG must traverse a large number of groups, which results in lower QPS than in other workloads.

NHQ incorporates attribute similarity by averaging binary matches across attributes. As the number of attributes grows, each match contributes less, reducing recall under complex, high-cardinality labels. Interestingly, indexing time decreases as the number of attributes grows, suggesting that index construction may terminate early when too few base vectors share a label to form sufficient edges, resulting in weak graph connectivity and lower recall. This pattern is reflected in the recall drops on UNG, RWalks, and YFCC in Figure 6-NHQ. Consequently, NHQ is the least robust strategy under hard workloads.

ACORN evaluates filter satisfaction before traversal and expands only filter-qualified nodes, achieving strong performance under most workloads. However, its denser-HNSW strategy breaks down under high-hardness workloads, especially when selectivity falls below approximately 0.1%. In this regime, dense connectivity alone cannot preserve meaningful connectivity among filter-satisfying nodes, causing the search to terminate prematurely before reaching the true targets and thus inflating QPS despite failed retrieval. This behavior is especially evident in **SIFT-High** and **GIST-High**. Because this denser-HNSW strategy is predicate-agnostic, its indexing cost is largely insensitive to the base data label distribution, but it consistently requires the highest indexing time and memory footprint among the evaluated methods.

RWalks requires base labels to be flattened into binary vectors, making it sensitive to attribute count and missing probability and causing greater performance variation than in other strategies. RWalks also degrades under high-hardness workloads, consistent with its original finding that recall collapses at extremely low selectivity when index construction fails to establish effective traversal paths. Its binary label representation also increases memory overhead as the number of base labels grows, as in RWalks. However, this design is effective when there are few query filters and remains reliable even with large attribute cardinalities. This behavior aligns with the workload configuration of RWalks, enabling strong performance with few active labels and high-cardinality attributes.

5.4 Hardness as a Bridge (RQ4)

Another practical implication of α -Hardness and HCBGen is that they bridge research benchmarks and real-world hybrid workloads. Because such workloads are rarely released or standardized, we conduct a fidelity experiment by emulating the query workloads of four semi-real datasets. We consider two deployment scenarios. In the first, the base data is available but the original query workload is not, so the generator receives only the target hardness information; this corresponds to the *base labels loading* configuration with MATCH-PDF mode (*Generated with base data*). In the second, more challenging scenario, neither base data nor query logs are available, and the generator again receives only the target hardness information (*Generated w/o base data*). In this case, base

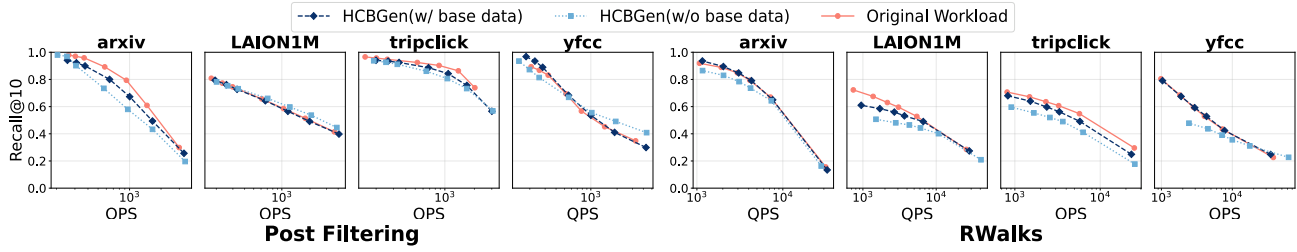


Figure 8: Recall–QPS trade-offs on four semi-real datasets for Post-Filtering and RWalks, comparing the original workload with two MATCH-PDF workloads generated by our generator with and without base data. The generated workloads largely preserve the performance trends of the original workload, including the workload generated without base data.

labels are synthesized using a fixed configuration with 2 attributes of cardinality 12. This extreme but practical setting demonstrates the generality of hardness as a standalone workload descriptor.

Figure 7 compares the hardness distributions of the original workloads (given) with those of the two proxy workloads generated by our method. MATCH-PDF enables the generator to closely fit the target hardness distribution across all four datasets. To quantify distributional similarity, we report (i) the *Kolmogorov–Smirnov statistic*[46] and (ii) the *1-Wasserstein distance*[41] between the generated and given hardness distributions. Both metrics consistently show that the generated workloads closely match the given ones.

We further test this argument by asking whether matching hardness distributions also matches performance. Figure 8 reports recall–QPS trade-offs for Post-Filtering and RWalks under three workloads per dataset. The generated workloads closely track the trade-off curves of the given workloads. Notably, even when the generator has no access to the base data, it still reproduces similar performance trends. This supports two conclusions: (i) α -Hardness is a strong proxy for performance-relevant workload difficulty, and (ii) hardness-controlled generation can faithfully emulate unseen workloads using only hardness profiles. These results show that our approach can generate privacy-preserving proxy benchmarks that retain the performance-critical properties of real-world hybrid workloads. This enables realistic benchmarking of hybrid search systems without access to sensitive user logs or proprietary query traces. More broadly, the results suggest that α -Hardness captures the dominant factors governing hybrid search performance, making it an effective surrogate for approximating unseen workloads.

6 RELATED WORKS

Vector Search Hardness Estimation. Prior work characterizes vector-search hardness through intrinsic data properties such as dimensionality, local density variation, and hubness, which shape nearest-neighbor structure and distance concentration [2, 6, 39]. Steiner Hardness further gives an execution-oriented measure for graph-based ANN by estimating the traversal effort needed to reach the target neighborhood [56]. Our α -Hardness extends this view to FANN by preserving traversal difficulty as the fetch component while additionally modeling over-fetching and filter-induced execution effects. To the best of our knowledge, no prior work explicitly defines query hardness for FANN.

FANN Benchmarking. Recent work has introduced FANN benchmarking studies. Shi et al. [45] construct semi-real datasets with controlled factors such as label length, selectivity, and dataset size, but sample query labels from the dataset, which biases queries toward easy cases under our hardness estimator. Lin et al. [30] discuss query-difficulty factors conceptually, but also rely on query sampling, inheriting label-distribution bias and limiting robustness evaluation. Iff et al. [20] introduce a semi-real hybrid dataset with diverse query labels, yet without an explicit notion of query-level difficulty, their representativeness remains unclear. Existing FANN benchmarks therefore lack query-level hardness modeling, leaving robustness under hard queries unexplored.

Benchmark Sensitivity in Database Systems. Benchmark brittleness is a recognized problem across database research. In relational systems, TPC-H and TPC-DS have been shown to miss real-world workloads in schema type, expression complexity, and operator structure [53], while cloud-scale production workloads exhibit write-heavy pipelines and long-tailed distributions absent from standard benchmarks [52]. In ANN search, algorithm rankings shift substantially across datasets [29] and also depend on whether queries are easy or hard [15]. Together, these results show that benchmark design choices often determine performance rankings.

7 CONCLUSION

In this paper, we presented α -Hardness, an execution-driven measure of hybrid-query difficulty that aligns consistently with observed FANN performance across datasets and strategies, unlike selectivity- or correlation-based proxies. Using α -Hardness as a control knob, HCBGen generates hardness-controlled benchmarks and can approximate real workloads by matching hardness profiles, revealing that many existing benchmarks are biased toward easy queries and can overstate robustness.

ACKNOWLEDGMENTS

This work was supported in part by the National Research Foundation of Korea (NRF) grant (RS-2025-00560762), the Institute of Information Communications Technology Planning Evaluation (IITP) grants (RS-2024-00454666, RS-2025-25442338, RS-2021-II211343), and Samsung Electronics Co., Ltd. (IO251215-14535-01). This research was also conducted under the Creative-Pioneering Researchers Program at Seoul National University (SNU). J. Do is affiliated with ASRI, SNU.

REFERENCES

- [1] Anas Ait Aomar, Karima Echihabi, Marco Arnaboldi, Ioannis Alagiannis, Damien Hilloulin, and Manal Cherkaoui. 2025. RWalks: Random Walks as Attribute Diffusers for Filtered Vector Search. *Proc. ACM Manag. Data* 3, 3, Article 212 (June 2025), 26 pages. <https://doi.org/10.1145/3725349>
- [2] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E. Houle, Ken-ichi Kawarabayashi, and Michael Nett. 2015. Estimating Local Intrinsic Dimensionality. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Sydney, NSW, Australia) (KDD '15). Association for Computing Machinery, New York, NY, USA, 29–38. <https://doi.org/10.1145/2783258.2783405>
- [3] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2020. ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Inf. Syst.* 87, C (Jan. 2020), 13. <https://doi.org/10.1016/j.is.2019.02.006>
- [4] Michael Barbaro and Tom Zeller. 2006. A Face Is Exposed for AOL Searcher No. 4417749. *The New York Times* (2006).
- [5] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breiteringer. 2016. Research-paper recommender systems: a literature survey. *Int. J. Digit. Libr.* 17, 4 (Nov. 2016), 305–338. <https://doi.org/10.1007/s00799-015-0156-0>
- [6] Kevin Beyer, Jonathan Goldstein, Raghu Ramakrishnan, and Uri Shaft. 1999. When is “nearest neighbor” meaningful? In *ICDT*. 217–235. https://doi.org/10.1007/3-540-49257-7_15
- [7] Peter A. Boncz, Marcin Zukowski, and Niels Nes. 2005. MonetDB/X100: Hyper-Pipelining Query Execution. In *CIDR*.
- [8] Yuzheng Cai, Jiayuan Shi, Yizhuo Chen, and Weiguo Zheng. 2024. Navigating Labels and Vectors: A Unified Approach to Filtered Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 6, Article 246 (Dec. 2024), 27 pages. <https://doi.org/10.1145/3698822>
- [9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishu Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Isipir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS '16)*. Association for Computing Machinery, New York, NY, USA, 7–10. <https://doi.org/10.1145/2988450.2988454>
- [10] Jacob Cohen. 1988. *Statistical Power Analysis for the Behavioral Sciences*. Lawrence Erlbaum Associates.
- [11] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys)*. 191–198. <https://doi.org/10.1145/2959100.2959190>
- [12] Russ Cox. 2007. Regular Expression Matching Can Be Simple And Fast. Online article. <https://swtch.com/~rsc/regexp/regexp1.html> Accessed: 2026-06-13.
- [13] Hanjun Dai, Bethany Yixin Wang, Xingchen Wan, Bo Dai, Sherry Yang, Azade Nova, Pengcheng Yin, Phitchaya Mangpo Phothilimthana, Charles Sutton, and Dale Schuurmans. 2024. UQE: A Query Engine for Unstructured Databases. *arXiv:2407.09522 [cs.DB]* <https://arxiv.org/abs/2407.09522> Accessed: 2026-06-13.
- [14] Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM* 56, 2 (Feb. 2013), 74–80. <https://doi.org/10.1145/2408776.2408794>
- [15] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2018. The Lernaean Hydra of Data Series Similarity Search: An Experimental Evaluation of the State of the Art. *Proc. VLDB Endow.* 12, 2 (Oct. 2018), 112–127. <https://doi.org/10.14778/3282495.3282498>
- [16] Joshua Engels, Benjamin Landrum, Shangdi Yu, Laxman Dhulipala, and Julian Shun. 2024. Approximate Nearest Neighbor Search with Window Filters. *arXiv:2402.00943 [cs.DS]* <https://arxiv.org/abs/2402.00943> Accessed: 2026-06-13.
- [17] Simon F. Goldsmith, Alex S. Aiken, and Daniel S. Wilkerson. 2007. Measuring empirical computational complexity. In *Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (Dubrovnik, Croatia) (ESEC-FSE '07)*. Association for Computing Machinery, New York, NY, USA, 395–404. <https://doi.org/10.1145/1287624.1287681>
- [18] Siddharth Gollapudi, Neel Karia, Varun Sivashankar, Ravishankar Krishnaswamy, Nikit Begwani, Swapnil Raz, Yiyong Lin, Yin Zhang, Neelam Mahapatro, Premkumar Srinivasan, Amit Singh, and Harsha Vardhan Simhadri. 2023. Filtered-DiskANN: Graph Algorithms for Approximate Nearest Neighbor Search with Filters. In *Proceedings of the ACM Web Conference 2023 (WWW '23)*. Association for Computing Machinery, New York, NY, USA, 3406–3416. <https://doi.org/10.1145/3543507.3583552>
- [19] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning (ICML '20)*. JMLR.org, Article 364, 10 pages.
- [20] Patrick Iff, Paul Brügger, Marcin Chrapek, Maciej Besta, and Torsten Hoefler. 2025. Benchmarking Filtered Approximate Nearest Neighbor Search Algorithms on Transformer-based Embedding Vectors. *arXiv preprint arXiv:2507.21989* (2025). <https://arxiv.org/abs/2507.21989> Accessed: 2026-06-13.
- [21] Yannis E. Ioannidis. 1996. Query Optimization. *Comput. Surveys* 28, 1 (1996), 121–123.
- [22] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 1 (Jan. 2011), 117–128. <https://doi.org/10.1109/TPAMI.2010.57>
- [23] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data* 7, 3 (2021), 535–547. <https://doi.org/10.1109/TBDATA.2019.2921572>
- [24] Andrew Kane. 2024. pgvector: Open-source vector similarity search for Postgres. <https://github.com/pgvector/pgvector>. GitHub repository, accessed 2026-04-19.
- [25] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.). Association for Computational Linguistics, Online, 6769–6781. <https://doi.org/10.18653/v1/2020.emnlp-main.550>
- [26] Viktor Leis, Andrey Gubichev, Atanas Mirchev, Peter Boncz, and Thomas Neumann. 2015. How Good Are Query Optimizers, Really? *Proceedings of the VLDB Endowment* 9, 3 (2015), 204–215. <https://doi.org/10.14778/2850583.2850594>
- [27] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) (NIPS '20). Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.
- [28] Sen Li, Fuyu Lv, Taiwei Jin, Guli Lin, Keping Yang, Xiaoyi Zeng, Xiao-Ming Wu, and Qianli Ma. 2021. Embedding-based Product Retrieval in Taobao Search. *arXiv:2106.09297 [cs.LG]* <https://arxiv.org/abs/2106.09297> Accessed: 2026-06-13.
- [29] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data — Experiments, Analyses, and Improvement. *IEEE Transactions on Knowledge & Data Engineering* 32, 08 (Aug. 2020), 1475–1488. <https://doi.org/10.1109/TKDE.2019.2909204>
- [30] Yanjun Lin, Kai Zhang, Zhenying He, Yinan Jing, and X. Sean Wang. 2025. Survey of Filtered Approximate Nearest Neighbor Search over the Vector-Scalar Hybrid Data. *arXiv preprint arXiv:2505.06501* (2025). <https://arxiv.org/abs/2505.06501> Accessed: 2026-06-13.
- [31] Eric Lo, Nick Cheng, Wilfred W. Lin, Wing-Kai Hon, and Byron Choi. 2014. MyBenchmark: generating databases for query workloads. *The VLDB Journal* 23, 6 (Dec. 2014), 895–913. <https://doi.org/10.1007/s00778-014-0354-1>
- [32] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. 2019. ViLBERT: Pretraining Task-Agnostic Visiolinguistic Representations for Vision-and-Language Tasks. *arXiv:1908.02265 [cs.CV]* <https://arxiv.org/abs/1908.02265> Accessed: 2026-06-13.
- [33] Yu A. Malkov and D. A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42, 4 (2020), 824–836. <https://doi.org/10.1109/TPAMI.2018.2889473>
- [34] Malteos. 2022. *Aspect Paper Embeddings*. <https://huggingface.co/datasets/malteos/aspect-paper-embeddings> Accessed: 2026-01-01.
- [35] Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilya Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. *arXiv:1906.00091 [cs.LG]* <https://arxiv.org/abs/1906.00091> Accessed: 2026-06-13.
- [36] Joseph O'Rourke. 1998. *Computational Geometry in C* (2nd ed.). Cambridge University Press, USA.
- [37] Liana Patel, Peter Kraft, Carlos Guestrin, and Matei Zaharia. 2024. ACORN: Performant and Predicate-Agnostic Search Over Vector Embeddings and Structured Data. *Proc. ACM Manag. Data* 2, 3, Article 120 (May 2024), 27 pages. <https://doi.org/10.1145/3654923>
- [38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020 [cs.CV]* <https://arxiv.org/abs/2103.00020> Accessed: 2026-06-13.
- [39] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. 2010. Hubs in Space: Popular Nearest Neighbors in High-Dimensional Data. *J. Mach. Learn. Res.* 11 (Dec. 2010), 2487–2531.
- [40] Navid Rekasaz, Oleg Lesota, Markus Schedl, Jon Brasley, and Carsten Eickhoff. 2021. TripClick: The Log Files of a Large Health Web Search Engine. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2507–2513. <https://doi.org/10.1145/3404835.3463242>
- [41] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. 2000. The Earth Mover's Distance as a Metric for Image Retrieval. *International Journal of Computer Vision* 40, 2 (2000), 99–121. <https://doi.org/10.1023/A:1026543900054>

- [42] R.S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. 1996. Role-based access control models. *Computer* 29, 2 (1996), 38–47. <https://doi.org/10.1109/2.485845>
- [43] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. 2021. LAION-400M: Open Dataset of CLIP-Filtered 400 Million Image–Text Pairs. *arXiv preprint arXiv:2111.02114* (2021). <https://arxiv.org/abs/2111.02114> Accessed: 2026-06-13.
- [44] Patricia G. Selinger, Morton M. Astrahan, Donald D. Chamberlin, Raymond A. Lorie, and Thomas G. Price. 1979. Access Path Selection in a Relational Database Management System. In *Proceedings of the 1979 ACM SIGMOD International Conference on Management of Data*. ACM, 23–34. <https://doi.org/10.1145/582095.582099>
- [45] Jiayang Shi, Yuzheng Cai, and Weiguo Zheng. 2025. Filtered Approximate Nearest Neighbor Search: A Unified Benchmark and Systematic Experimental Study [Experiment, Analysis & Benchmark]. *arXiv:2509.07789* [cs.DB] <https://arxiv.org/abs/2509.07789> Accessed: 2026-06-13.
- [46] N. V. Smirnov. 1948. Table for estimating the goodness of fit of empirical distributions. *The Annals of Mathematical Statistics* 19, 2 (1948), 279–281. <https://doi.org/10.1214/aoms/1177730256>
- [47] Charles Spearman. 1904. The proof and measurement of association between two things. *The American Journal of Psychology* 15, 1 (1904), 72–101. <https://doi.org/10.2307/1412159>
- [48] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. 2016. YFCC100M: The New Data in Multimedia Research. *Commun. ACM* 59, 2 (Jan. 2016), 64–73. <https://doi.org/10.1145/2812802>
- [49] Ken Thompson. 1968. Programming Techniques: Regular Expression Search Algorithm. *Commun. ACM* 11, 6 (1968), 419–422. <https://doi.org/10.1145/363347.363387>
- [50] Transaction Processing Performance Council. 2024. *TPC-H Benchmark Specification*. <https://www.tpc.org/tpch/> Accessed: 2026-06-13.
- [51] Immanuel Trummer, Junxiong Wang, Ziyun Wei, Deepak Maram, Samuel Moseley, Saehan Jo, Joseph Antonakakis, and Ankush Rayabhari. 2021. SkinnerDB: Regret-bounded Query Evaluation via Reinforcement Learning. *ACM Trans. Database Syst.* 46, 3, Article 9 (Sept. 2021), 45 pages. <https://doi.org/10.1145/3464389>
- [52] Alexander van Renen, Dominik Horn, Pascal Pfeil, Kapil Vaidya, Wenjian Dong, Murali Narayanaswamy, Zhengchun Liu, Gaurav Saxena, Andreas Kipf, and Tim Kraska. 2024. Why TPC is Not Enough: An Analysis of the Amazon Redshift Fleet. *Proc. VLDB Endow.* 17, 11 (July 2024), 3694–3706. <https://doi.org/10.14778/3681954.3682031>
- [53] Adrian Vogelsgesang, Michael Haubenschild, Jan Finis, Alfons Kemper, Viktor Leis, Tobias Muehlbauer, Thomas Neumann, and Manuel Then. 2018. Get Real: How Benchmarks Fail to Represent the Real World. In *Proceedings of the Workshop on Testing Database Systems* (Houston, TX, USA) (*DBTest '18*). Association for Computing Machinery, New York, NY, USA, Article 1, 6 pages. <https://doi.org/10.1145/3209950.3209952>
- [54] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) (*SIGMOD '21*). Association for Computing Machinery, New York, NY, USA, 2614–2627. <https://doi.org/10.1145/3448016.3457550>
- [55] Mengzhao Wang, Lingwei Lv, Xiaoliang Xu, Yuxiang Wang, Qiang Yue, and Jiongkang Ni. 2023. An efficient and robust framework for approximate nearest neighbor search with attribute constraint. In *Proceedings of the 37th International Conference on Neural Information Processing Systems* (New Orleans, LA, USA) (*NIPS '23*). Curran Associates Inc., Red Hook, NY, USA, Article 692, 14 pages.
- [56] Zeyu Wang, Qitong Wang, Xiaoxing Cheng, Peng Wang, Themis Palpanas, and Wei Wang. 2024. Steiner-Hardness: A Query Hardness Measure for Graph-Based ANN Indexes. *Proc. VLDB Endow.* 17, 13 (2024), 4668–4682. <https://doi.org/10.14778/3704965.3704974>
- [57] Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. 2020. AnalyticDB-V: a hybrid analytical engine towards query fusion for structured and unstructured data. *Proc. VLDB Endow.* 13, 12 (Aug. 2020), 3152–3165. <https://doi.org/10.14778/3415478.3415541>
- [58] Tianming Wu and Dixin Tang. 2025. PathFinder: Efficiently Supporting Conjunctions and Disjunctions for Filtered Approximate Nearest Neighbor Search. *arXiv:2511.00995* [cs.DB] <https://arxiv.org/abs/2511.00995> Accessed: 2026-06-13.
- [59] Xiaoliang Xu, Chang Li, Yuxiang Wang, and Yixing Xia. 2020. Multiattribute Approximate Nearest Neighbor Search Based on Navigable Small World Graph. *Concurrency and Computation: Practice and Experience* 32, 24 (2020), e5970. <https://doi.org/10.1002/cpe.5970>
- [60] Artem Babenko Yandex and Victor Lempitsky. 2016. Efficient Indexing of Billion-Scale Datasets of Deep Descriptors. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2055–2063. <https://doi.org/10.1109/CVPR.2016.226>
- [61] Han Zhang, Songlin Wang, Kang Zhang, Zhiling Tang, Yunjiang Jiang, Yun Xiao, Weipeng Yan, and Wen-Yun Yang. 2020. Towards Personalized and Semantic Retrieval: An End-to-End Solution for E-commerce Search via Embedding Learning. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Virtual Event, China) (*SIGIR '20*). Association for Computing Machinery, New York, NY, USA, 2407–2416. <https://doi.org/10.1145/3397271.3401446>
- [62] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2021. Graph Neural Networks: A Review of Methods and Applications. *arXiv:1812.08434* [cs.LG] <https://arxiv.org/abs/1812.08434> Accessed: 2026-06-13.
- [63] Zijian Zhu, Yongli Wang, and Dongmei Liu. 2025. Hybrid Query of Boolean Filter and Vector Similarity Search: Benchmark, Comparison and Direction. In *2025 5th Asia Conference on Information Engineering (ACIE)*. 117–124. <https://doi.org/10.1109/ACIE64499.2025.00026>