Bandwidth-First ANN Refinement: Precision-on-Demand in Vector Databases

Teresa Zhang
teresaz@stanford.edu
Stanford University
Stanford, CA, USA

Abstract

Modern vector databases and RAG pipelines increasingly bottleneck on DRAM bandwidth during the refinement stage of approximate nearest neighbor (ANN) search, leaving abundant CPU/GPU FLOPs idle. We present a bandwidth-first, representation-preserving refinement stage that reduces bytes per candidate without re-encoding vectors or modifying indexes. The stage performs precision-on-demand with early rejection: it first reads a reduced-precision prefix and upgrades to full precision only for survivors. To enable safe pruning from reduced-precision reads, we derive two families of early-rejection bounds for both cosine similarity and Euclidean distance: one deterministic and sign-aware that guarantees zero false negatives, and one Hoeffding-based that provides a single control parameter to trade recall for bandwidth. To make partial-precision access effective under commodity DRAM, we introduce a disaggregated, bit-plane layout with lightweight bit-wise reordering that improves compressibility and leverages existing hardware lossless decompression on modern CPUs/GPUs. Across real-world datasets, precision-on-demand with early rejection reduces ANN search refinement bandwidth by up to 60%, with an additional ~1.8× reduction from lossless compression, while maintaining high recall. Orthogonal to index choice (e.g., HNSW, IVF, PQ) and representation-preserving, this method can be integrated into existing vector-database stacks to alleviate DRAM-bandwidth bottlenecks and increase query throughput.

ACM Reference Format:
Teresa Zhang. 2025. Bandwidth-First ANN Refinement: Precision-on-Demand in Vector Databases. In *Proceedings of International Conference on Extending Database Technology (EDBT).* ACM, New York, NY, USA, 13 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Nearest neighbor search, particularly approximate nearest neighbor (ANN) search, is a cornerstone of numerous high-value applications such as information retrieval [1, 2], recommendation systems [3, 4], retrieval-augmented generation (RAG) [5, 6], and anomaly detection [7, 8]. In high-dimensional spaces, the curse of dimensionality [9] renders exact nearest neighbor search practically infeasible, making ANN methods indispensable in practice. State-of-the-art ANN systems, e.g., HNSW (Hierarchical Navigable Small World) [10] often combined with dimension-reduction techniques [11–13], have been designed primarily for *computational* efficiency. While these techniques can incidentally reduce memory traffic, they do not treat memory bandwidth as a first-class optimization target. This limitation is increasingly misaligned with modern computing infrastructure: CPUs and GPUs today sustain hundreds of tera-FLOPS alongside TB/s-scale DRAM bandwidth. Consequently, with arithmetic intensity $O(1)$, the post-candidate-generation distance evaluation is typically memory-bound, leaving substantial compute capacity underutilized.

Taking a memory-centric view, this paper presents design techniques that minimize data transfer volume for post–candidate-generation distance evaluation while preserving ANN search accuracy. Empirically, most distance computations merely confirm rejection of a candidate. For example, Gao *et al.* [14] report that 83% of HNSW distance computations and 99% of IVF (inverted file index) distance computations result in candidate rejection. This suggests that precise full-precision distance calculations are often unnecessary, provided we can reliably avoid rejecting points that should be retained. Prior work has exploited this tolerance for approximation by modifying vector representations (e.g., product quantization [15], space projection [11]), thereby reducing both compute and bandwidth.

Unlike most existing solutions, which rely on altering vector representations through compression or dimensionality reduction, our work focuses on reducing memory bandwidth usage while preserving the original in-memory representation (i.e., dimensionality and floating-point precision). At a high level, our method performs *precision-on-demand with early rejection*: rather than fetching full-precision vectors up front, we first read a reduced-precision prefix (most-significant bits), compute an approximate score, and prune unlikely candidates; the remainder is fetched only for survivors. This design raises two central challenges: (1) how to construct early-rejection mechanisms that substantially cut memory traffic without compromising ANN accuracy, and (2) how to achieve real DRAM bandwidth savings given the fixed-burst, word-oriented access constraints of commodity memory systems.

We address the first challenge by developing a mathematically grounded framework for *adaptive early-rejection thresholds* that applies to both cosine similarity and Euclidean distance. We model the effect of reduced-precision reads (retaining sign and exponent while truncating mantissa bits) and relate an approximate score to the true distance. We provide three complementary families of cushions: (i) conservative global ℓ_1 and ℓ_2 cushions that bound the deviation via $\sum_i |q_i|\,\Delta_i$ and $\|\Delta\|_2$, respectively, both zero-miss by construction and useful as baselines; (ii) a deterministic, sign-aware cushion that exploits truncation toward zero to tighten the

EDBT, Tampere, Finland

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/XXXXXXX.XXXXXXX

admissible error while *still guaranteeing no false negatives*; and (iii) a Hoeffding-based probabilistic cushion that introduces a single, interpretable parameter to *trade recall for bandwidth* and that typically tightens with dimension (whereas the ℓ_1/ℓ_2 cushions remain comparatively loose). The arithmetic overhead of all cushions is lightweight (simple accumulations with precomputed error radii), so the guard checks themselves do not become a new bottleneck. In summary, ℓ_1/ℓ_2 provide safety and analytical clarity, the sign-aware cushion improves tightness without sacrificing safety, and the Hoeffding cushion exposes a practical control knob when small recall losses are acceptable.

To address the second challenge on turning partial reads into real DRAM bandwidth savings under fixed-burst, word-oriented memory, we introduce a *disaggregated in-memory placement* that physically separates each floating-point value into sign, retained upper bits, and truncated remainders, stored in distinct DRAM regions. This organization allows the system to fetch only the upper bits in the first stage and defer remainder fetches to the few survivors, aligning transferred bytes with the precision actually used. We further propose *bit-wise shuffling* that groups statistically similar high-order bits across dimensions, improving compressibility of the upper-bit segments. Because modern CPUs/GPUs already provide hardware-assisted lossless decompression, these segments can be decompressed at memory-throughput rate, amplifying effective bandwidth with negligible runtime overhead. Evaluations on real-world datasets under both cosine and Euclidean metrics show that precision-on-demand with early rejection alone reduces memory traffic by up to 60% while maintaining $\geq$ 99% recall, and that combining the disaggregated layout with compression yields an additional $\sim$1.8$\times$ reduction without loss of search accuracy.

Together, these techniques form a bandwidth-first, representation-preserving optimization stack that integrates seamlessly with existing ANN pipelines. By cutting data movement, they convert otherwise idle compute capacity into higher query throughput, without retraining, re-encoding, or specialized hardware. The approach is *orthogonal to index choice* (e.g., HNSW, IVF, PQ) and leaves floating-point representations intact, making it straightforward to adopt in current vector-database systems. More broadly, the results position high-dimensional similarity search as a *memory-centric* problem and invite further work on bandwidth-aware designs that push the efficiency frontier beyond compute-centric techniques.

2 Background and Related Work

2.1 Basics and Motivation

ANN search aims to retrieve the top-K vectors most similar to a query $q \in \mathbb{R}^D$ from a dataset $\mathbf{S} \subset \mathbb{R}^D$, under a distance metric such as Euclidean distance or cosine similarity. Over the past two decades, a wide range of ANN algorithms have been developed, which can be broadly categorized into four families: graph-based methods (e.g., HNSW [10]), quantization-based methods (e.g., product quantization (PQ) [15] and its successors), tree-based methods (e.g., KD-trees [16], cover trees [17]), and hashing-based methods (e.g., LSH [1]). These approaches differ primarily in how they implement the candidate-generation stage of ANN search. Among them, graph-based methods have become the de facto standard in practice due to their robustness and empirical accuracy, while tree- and hash-based methods often degrade in very high-dimensional spaces, a direct consequence of the curse of dimensionality.

Irrespective of the candidate-generation strategy, all pipelines end with a post-candidate-generation distance-evaluation phase ("refinement") that compares q to each candidate $c \in C$ and maintains a top-K list. Formally, given an index I that returns a candidate set $C = I(q)$ with $|C| \ll |\mathbf{S}|$, the system computes $d(q, c)$ for every $c \in C$ under the chosen metric and admits c iff $d(q, c)$ is better than $d(q, w)$ for the current worst element w in the heap. On modern CPUs/GPUs this phase is typically memory-bound: for a D-dimensional candidate, the computation performs $\Theta(D)$ floating-point operations but also transfers $\Theta(D)$ numbers from DRAM (i.e., $D \cdot b$ bytes with $b = 2$ for FP16 or $b = 4$ for FP32), yielding a low arithmetic intensity of $O(1)$. Consequently, throughput is limited by bytes moved rather than available FLOPs. Moreover, prior measurements report that the large majority of comparisons ultimately reject the candidate (e.g., high rejection rates for HNSW and IVF [14]), implying that full-precision evaluation of every candidate is often wasteful provided true top-K neighbors are not mistakenly discarded. A common way to exploit this tolerance has been to change the representation via quantization (e.g., PQ [15]) or dimensionality reduction (e.g., PCA/projections [11–13]), which can reduce both compute and memory traffic but ties efficiency to re-encoding choices and alters the in-memory vectors. In contrast, we adopt a bandwidth-first view that seeks to reduce the bytes per candidate while preserving both dimensionality and floating-point precision of the stored vectors. The next subsection reviews related work aimed at lowering the cost of this final phase and explains why memory traffic has generally been treated as secondary rather than as a first-class objective.

2.2 Prior Work on Refinement Efficiency

Prior approaches to accelerating the refinement stage of ANN pipelines largely fall into two well-studied categories: vector compression and dimensionality reduction.

Vector compression (quantization). Quantization methods, most notably product quantization (PQ) [15], preserve dimensionality but approximate each vector with compact codes drawn from learned codebooks, enabling fast distance estimation via table lookups. In PQ, a D-dimensional vector is partitioned into m sub-vectors; each sub-vector is mapped to the nearest centroid in a sub-codebook, and only the m centroid indices are stored. This design significantly reduces memory footprint and allows approximate distances to be computed by summing m precomputed lookup values per candidate. Many variants extend this idea to improve reconstruction accuracy or adapt to hardware constraints, including optimized/adaptive PQ, additive and composite quantization, and recent code-centric formulations such as RaBitQ [18], which encodes D-dimensional vectors into D-bit strings with probabilistic error control, and LVQ [19], which integrates scalar quantization with layouts that are prefetch- and SIMD-aware. While these techniques can reduce both compute and memory traffic during the refinement phase, they require learning codebooks or per-dataset calibration, and their accuracy-efficiency trade-offs are governed by quantization granularity. Moreover, distance estimation still operates on whole-code units (e.g., bytes/words per sub-vector) and therefore cannot adapt the number

of transferred *bits* below the code's word granularity.

Dimensionality reduction. Dimensionality-reduction methods embed high-dimensional vectors into a lower-dimensional space to cut both computation and data movement while approximately preserving neighborhood structure. Classical linear techniques include PCA [13] and random projections [11]; they directly lower the per-candidate arithmetic cost and reduce transferred bytes by operating in $d \ll D$ dimensions. More recent work explores multi-resolution embeddings, such as Matryoshka [20], where vectors are represented at multiple granularities so that coarse layers provide fast filtering and finer layers refine results as needed. These approaches provide a flexible accuracy-efficiency spectrum, but they also require transforming (or retraining) embeddings and their effectiveness depends on how well the reduced space preserves the original neighborhood relationships. In settings with fixed or legacy embeddings, re-encoding can be costly or undesirable.

Discussion. Both compression and dimensionality reduction can be highly effective, yet they *alter the representation* and implicitly make accuracy a function of the chosen codes or embedding dimensionality. In all cases, reductions in memory traffic arise as a *by-product* of coding or projection choices rather than from an explicit, first-class optimization of transferred bytes during refinement. By contrast, our work takes a *bandwidth-first* perspective: we keep the original floating-point representation intact and optimize data movement itself. In particular, we introduce a precision-on-demand mechanism that initially fetches only a reduced-precision prefix of each value and escalates to full precision only when necessary, coupled with decision-theoretic cushions that make partial evaluation safe or tunably conservative. This representation-preserving approach is orthogonal to the above families and can be layered atop existing indices and embedding choices.

2.3 Distinctions from Prior Work

Our approach differs from prior methods in several respects while remaining complementary to existing ANN pipelines. First, we treat *bandwidth* as a first-class optimization target while preserving original floating-point vectors; by contrast, compression and dimensionality-reduction methods change the representation (and hence induce accuracy trade-offs tied to codebooks or reduced spaces), and pruning/routing methods primarily focus on reducing the count of distance evaluations rather than the bytes transferred. Second, we enable fine-grained precision control below the word level so that only the most significant bits of each value are fetched initially, ensuring that savings materialize as fewer DRAM bytes rather than as full words (or full compressed codes) per candidate. These choices make the technique orthogonal to established designs and straightforward to layer on top of them.

(1) **Representation preservation with a bandwidth-first objective.** Compression and quantization methods (e.g., PQ, RaBitQ, LVQ) improve efficiency but re-encode vectors and alter their representation [15, 18, 19]. Dimensionality-reduction methods (e.g., PCA, Matryoshka) [13, 20] similarly transform embeddings to reduce computation and storage. Pruning and routing methods (e.g., ADSampling, FINGER, Probabilistic Routing) [21–23] preserve floating-point vectors but focus on reducing the number of candidate evaluations. In contrast, we *explicitly optimize data movement* while keeping the original floating-point representation intact, so that accuracy is not coupled to re-encoding choices.

(2) **Precision-on-demand at sub-word granularity.** Prior systems typically operate on full words or fixed-size codes, with no way to adapt precision below that granularity. Our design performs *precision-on-demand*: it fetches only a reduced-precision prefix (most-significant bits) sufficient for early rejection and retrieves the remainder only when necessary. This is enabled by a *disaggregated, bit-plane* layout that aligns with DRAM bursts, so that partial reads translate into *actual* fewer bytes transferred, and by lightweight bit-wise shuffling that further improves compressibility for hardware-supported lossless decompression.

(3) **Guaranteed or tunable safety for early rejection.** Several routing/pruning techniques apply heuristic admission rules to reduce candidate evaluations [21–23]. We instead provide a hierarchy of mathematically grounded cushions that relate partial-precision scores to the true distance: conservative global ℓ_1/ℓ_2 cushions (zero-miss baselines), a *deterministic sign-aware* cushion that tightens the bound while still guaranteeing *no* false negatives, and a *Hoeffding-based* probabilistic cushion that exposes a single, interpretable parameter to trade recall for bandwidth. This makes the aggressiveness of partial evaluation explicit and controllable.

(4) **Ease of adoption and orthogonality.** Because we preserve dimensionality and floating-point precision, our method can be dropped into existing stacks that already ingest FP16/FP32 embeddings and standard indices (e.g., HNSW and IVF). It can also co-exist with representation-changing methods (e.g., PQ or multi-resolution embeddings) by applying precision-on-demand within their final comparison stage, turning bandwidth savings into higher throughput without retraining or re-encoding.

Table 1 further summarizes these distinctions. Our approach is bandwidth-first, preserves full-precision representations, supports adaptive early rejection via formal cushions, aligns to DRAM access granularity through a bit-plane layout, and leverages lossless compression, while remaining drop-in compatible with existing ANN pipelines.

Table 1: Representative related methods vs. this work (✓: explicitly supported/claimed; —: not targeted).

Method	Preserve FP	Bandwidth -first	Early reject / routing	DRAM-aligned	Lossless comp.
RaBitQ [18]	—	—	—	—	—
ADSampling [21]	✓	—	✓	—	—
FINGER [22]	✓	—	✓	—	—
Prob. Routing [23]	✓	—	✓	—	—
LVQ [19]	—	Indirect	—	Partial	—
This work	✓	✓	✓	✓	✓

Notes. "Indirect" indicates that bandwidth reduction arises as a side-effect of representation changes rather than from an explicit bytes-first objective. "Partial" denotes partial alignment with DRAM

access patterns (e.g., structure-of-arrays layouts) without sub-word, bit-plane-level fetch.

3 Proposed Design Solutions

We target the post-candidate-generation distance evaluation, where throughput is limited by the memory bandwidth. Our approach is *representation-preserving* and *bandwidth-first*: it reduces transferred bytes per candidate without re-encoding vectors or changing dimensionality/precision. At a high level, we perform *precision-on-demand with early rejection*: rather than fetching full-precision vectors up front, we first read a reduced-precision prefix (most-significant bits), compute an approximate score, and prune unlikely candidates; the remainder is fetched only for survivors.

Two challenges arise under this design approach: (1) **Safe and effective partial evaluation.** Using reduced-precision values, we must bound the gap between the approximate and true distances. We therefore derive three families of *cushions*: conservative global ℓ_1/ℓ_2 cushions (zero-miss baselines), a *deterministic sign-aware* cushion that tightens bounds while still guaranteeing no false negatives, and a *Hoeffding-based probabilistic* cushion that exposes a single parameter to trade recall for bandwidth. (2) **Making savings materialize at DRAM granularity.** Because commodity DRAM fetches fixed-size bursts, we design a *disaggregated, bit-plane layout* with *bit-wise shuffling* so that partial-precision reads actually reduce bytes transferred and pair naturally with hardware-assisted lossless decompression on modern CPUs/GPUs.

Classical decision rule (unified). We define a score $d(q,c)$ so that larger is better:

$$d(q,c) = \begin{cases} \langle q,c\rangle, & \text{cosine (dot product)},\\ -\|q-c\|_2^2, & \text{Euclidean}, \end{cases} \qquad \tau \;=\; d(q,w),$$

where w is the current worst element in the top-K set. A candidate c is admitted iff $d(q,c) \geq \tau$.

With a reduced-precision prefix $\tilde{c}$ (mantissa truncated; write $c = \tilde{c} + \varepsilon$ with per-coordinate bounds $|\varepsilon_i| \leq \Delta_i$), we compute an approximate score $\hat{d} = d(q,\tilde{c})$ and guard early rejection with a cushion. For the deterministic case we use an *upper* cushion $U(q,\tilde{c},\Delta)$ such that $d(q,c) \leq \hat{s} + U$, and reject whenever

$$\hat{d} \;+\; U(q,\tilde{c},\Delta) \;<\; \tau.$$

For the probabilistic case we employ a Hoeffding-style cushion $T(\delta;\, q,\tilde{c},\Delta)$ and reject when

$$\hat{d} \;+\; T(\delta;\, q,\tilde{c},\Delta) \;<\; \tau,$$

which exposes a single parameter δ to trade recall for bandwidth. The remainder of this section detail these cushions and the memory layout that makes partial reads translate into real DRAM-byte savings.

3.1 Candidate Point Early Rejection: Cosine Similarity

Under cosine similarity we develop a set of *early-rejection cushions* that certify when a candidate can be discarded using only a reduced-precision prefix of its coordinates. We provide three deterministic cushions, two conservative global bounds (ℓ_1 and ℓ_2) and a *sign-aware* bound that leverages truncation toward zero, to guarantee zero false negatives, and one probabilistic (Hoeffding-based) cushion that exposes a single parameter to trade recall for bandwidth. Each cushion is computable in a single streaming pass and is friendly to SIMD/vectorized implementations. We first fix notation (Section 3.1.1) and formalize the floating-point truncation error model that supplies per-coordinate bounds Δ_i (Section 3.1.2). Building on this setup, Sections 3.1.3–3.1.5 develop the ℓ_1/ℓ_2, sign-aware, and Hoeffding cushions and instantiate rejection tests. Each cushion is computable in a single streaming pass and is amenable to SIMD/vectorized implementations.

3.1.1 Setup and Notation. Let $q \in \mathbb{R}^D$ denote the query vector, normalized so that $\|q\|_2 = 1$, and let $c \in \mathbb{R}^D$ denote a candidate vector from the dataset, also normalized so that $\|c\|_2 = 1$. Vectors are stored in full precision in memory, but in our precision-on-demand scheme we first fetch a *reduced-precision* prefix $\tilde{c}$ by truncating each coordinate's mantissa from M bits to $M' < M$ bits, retaining the original sign and exponent; the remainder is fetched only if needed.

The cosine similarity between q and c is

$$d_S(q,c) = \langle q,c\rangle = \sum_{i=1}^{D} q_i c_i. \tag{1}$$

When fetching a reduced-precision prefix of c, define

$$\hat{d}_S = \langle q,\tilde{c}\rangle, \qquad \varepsilon_i = c_i - \tilde{c}_i, \tag{2}$$

so that

$$d_S(q,c) = \hat{d}_S + E, \qquad E = \sum_{i=1}^{D} q_i \varepsilon_i. \tag{3}$$

Let $\tau_S = d_S(q,w)$ denote the current heap threshold (the similarity of the worst element w in the top-K set). A candidate is admitted iff $d_S(q,c) \geq \tau_S$. Our goal is to certify *early rejection* using only $(q,\tilde{c})$ and per-coordinate error bounds, i.e., to decide that $d_S(q,c) < \tau_S$ without fetching the mantissa remainders.

3.1.2 Floating-point Truncation Error Model. We assume IEEE-754 floating-point for concreteness, though the analysis extends to other formats (e.g., bfloat16 or custom FP). For normalized values ($1 \leq e_i \leq 2^{N_e} - 2$),

$$c_i = (-1)^{\sigma_i}\left(1 + \frac{m_i}{2^M}\right) 2^{\,e_i - B}, \tag{4}$$

and for subnormals ($e_i = 0$),

$$c_i = (-1)^{\sigma_i}\left(\frac{m_i}{2^M}\right) 2^{\,1-B}, \tag{5}$$

where σ_i is the sign bit, e_i the stored exponent, N_e the number of exponent bits, B the bias, and m_i the M-bit mantissa. Mantissa-only truncation to M' bits produces

$$m_i' = \left\lfloor \frac{m_i}{2^{M-M'}} \right\rfloor \cdot 2^{M-M'}, \tag{6}$$

which zeroes the $(M-M')$ least significant bits and *truncates toward zero* while preserving sign and exponent. The absolute truncation error satisfies

$$|\varepsilon_i| \;\leq\; \Delta_i := \begin{cases} 2^{\,e_i - B - M'} & \text{(normalized)},\\ 2^{\,1-B-M'} & \text{(subnormal)}, \end{cases} \tag{7}$$

so Δ_i depends only on (e_i, M') and not on the discarded bits. Because the reduced prefix $\tilde{c}$ includes sign and exponent, Δ_i is available as soon as $\tilde{c}$ is fetched. Truncation preserves the sign, including the IEEE-754 ±0 convention, i.e., $\text{sign}(\tilde{c}_i) = \text{sign}(c_i)$; this will be used by the sign-aware cushion.

3.1.3 Deterministic ℓ_1 and ℓ_2 Cushions. To safely reject candidates using reduced-precision data, we must bound the error term E in (3). Deterministic cushions do so by considering worst-case deviations, ensuring that no true top-K candidate is prematurely discarded.

From (3), the worst-case absolute error satisfies

$$|E| \;\le\; \sum_{i=1}^{D} |q_i|\,|\varepsilon_i| \;\le\; \sum_{i=1}^{D} |q_i|\,\Delta_i \;=:\; B_{\ell_1}(q,\Delta), \tag{8}$$

which yields the conservative ℓ_1 cushion. A second bound follows from Cauchy–Schwarz. Since $\|q\|_2 = 1$,

$$|E| \;\le\; \|q\|_2\,\|\varepsilon\|_2 \;\le\; \|\Delta\|_2 \;=:\; B_{\ell_2}(\Delta), \tag{9}$$

where the last inequality uses $|\varepsilon_i| \le \Delta_i$ coordinatewise. Combining these bounds gives the following *zero-miss* rejection tests:

$$\text{Reject if} \quad \hat{d}_S + B_{\ell_1}(q,\Delta) < \tau_S, \tag{10}$$

$$\text{or if} \quad \hat{d}_S + B_{\ell_2}(\Delta) < \tau_S. \tag{11}$$

Tightness and practical use. Because $\|q\|_2 = 1$, Cauchy–Schwarz implies $B_{\ell_1}(q,\Delta) \le B_{\ell_2}(\Delta)$ for every candidate. Thus the ℓ_1 cushion is never looser than the ℓ_2 cushion and is typically the more effective deterministic baseline. We keep B_{ℓ_2} for completeness (it is q-independent and mirrors the Euclidean case), but in practice (10) dominates (11). Both cushions are intentionally conservative; Section 3.1.4 tightens them using sign information, and Section 3.1.5 introduces a tunable probabilistic alternative.

3.1.4 Sign-Aware Deterministic Cushion. The ℓ_1 and ℓ_2 cushions bound E conservatively by assuming that every coordinate deviation $|\varepsilon_i|$ could *increase* the similarity. In practice, mantissa truncation shifts each coordinate toward zero *without changing its sign bit*. Hence the potential *upward* contribution of $q_i\varepsilon_i$ depends on the sign alignment between q_i and the (IEEE-754) sign of the stored prefix. To make this precise we adopt signed-zero semantics and write $\text{sign}^\star(x) \in \{-1, 0, +1\}$ as the function that returns the IEEE sign even when $x = \pm 0$. With truncation toward zero, $\text{sign}^\star(\tilde{c}_i) = \text{sign}^\star(c_i)$. We partition coordinates by sign alignment with the query:

$$\mathcal{P}(q,\tilde{c}) \;=\; \{\, i : \text{sign}(q_i) = \text{sign}^\star(\tilde{c}_i) \,\}, \tag{12}$$

$$\mathcal{N}(q,\tilde{c}) \;=\; \{\, i : \text{sign}(q_i) \ne \text{sign}^\star(\tilde{c}_i) \,\}. \tag{13}$$

Only indices in $\mathcal{P}$ can increase d_S via $q_i\varepsilon_i$; indices in $\mathcal{N}$ can only decrease it. This yields separate one-sided bounds:

$$U(q,\tilde{c},\Delta) \;=\; \sum_{i\in\mathcal{P}} |q_i|\,\Delta_i, \qquad L(q,\tilde{c},\Delta) \;=\; -\sum_{i\in\mathcal{N}} |q_i|\,\Delta_i, \tag{14}$$

and the range

$$\hat{d}_S \;+\; L \;\le\; d_S(q,c) \;\le\; \hat{d}_S \;+\; U. \tag{15}$$

Accordingly, we obtain the *sign-aware* rejection rule:

$$\text{reject } c \quad \text{if} \quad \hat{d}_S \;+\; U(q,\tilde{c},\Delta) \;<\; \tau_S. \tag{16}$$

This rule guarantees zero false negatives and is strictly tighter than the global cushions whenever $\mathcal{N} \ne \emptyset$, since

$$U(q,\tilde{c},\Delta) \;=\; \sum_{i\in\mathcal{P}} |q_i|\Delta_i \;\le\; \sum_{i=1}^{D} |q_i|\Delta_i \;=\; B_{\ell_1}(q,\Delta), \tag{17}$$

with equality only when $\mathcal{N} = \emptyset$.

Edge cases (zeros and subnormals). We note that truncation may produce $\tilde{c}_i = \pm 0$ for very small magnitudes. Using $\text{sign}^\star$ ensures that we still account for the true sign of c_i (available from the stored sign bit) when forming $\mathcal{P}$ and $\mathcal{N}$. Subnormal handling follows the bound Δ_i in (7); the sign-aware construction remains valid because truncation continues to move values toward zero without flipping the sign bit.

3.1.5 Hoeffding-Based Probabilistic Cushion. An alternative to deterministic cushions is to interpret the error term $E = \sum_{i=1}^{D} q_i\varepsilon_i$ as a bounded random sum. Since each summand satisfies $|q_i\varepsilon_i| \le |q_i|\,\Delta_i$, Hoeffding's inequality [24, 25] yields

$$\Pr(E \ge t) \;\le\; \exp\!\left(-\frac{t^2}{2\sum_{i=1}^{D}(q_i\Delta_i)^2}\right). \tag{18}$$

Given a user-specified level $\delta \in (0,1)$, inverting (18) produces the cushion

$$t_q(\delta) \;=\; \sqrt{2\log(1/\delta)\,\sum_{i=1}^{D}(q_i\Delta_i)^2}\,, \tag{19}$$

and the corresponding rejection rule

$$\text{Reject if } \hat{d}_S + t_q(\delta) < \tau_S, \tag{20}$$

where $\hat{d}_S = \langle q,\tilde{c}\rangle$ and $\tau_S = d_S(q,w)$ is the current heap threshold. (The sum in (19) can be accumulated in the same streaming pass used to form $\hat{d}_S$.)

Interpretation and tuning. With the deterministic mantissa truncation, ε_i is one-sided (truncation toward zero) and not independent of $\tilde{c}_i$, so δ in (19) should be viewed as a *tuning parameter* that controls the pruning aggressiveness, rather than as a strict tail probability. Smaller δ enlarges the cushion (safer but less aggressive), whereas larger δ shrinks it (more aggressive pruning at the cost of a nonzero false-negative rate). In practice, we found a simple rule of thumb effective: for $D \gtrsim 10^3$, choosing δ in the range 10^{-6} to 10^{-2} typically maintains high recall while providing noticeable bandwidth savings (Section 4).

Dimensional scaling. A key advantage of the Hoeffding cushion is that it tightens with dimension. For isotropic queries, $\mathbb{E}[q_i^2] \approx 1/D$, and for fixed kept mantissa length M' we have $\Delta_i = \Theta(2^{-M'})$ up to exponent factors, giving the approximation

$$t_q(\delta) \;\approx\; 2^{-M'}\sqrt{\frac{2\log(1/\delta)}{D}}. \tag{21}$$

Thus $t_q(\delta)$ shrinks like $D^{-1/2}$ as dimensionality grows, while the in-heap worst similarity τ_S concentrates toward zero at roughly $\Theta(\sqrt{\log N}/\sqrt{D})$ for N database points. In contrast, the deterministic cushions B_{ℓ_1}, B_{ℓ_2}, and the sign-aware U remain essentially flat with D, making them comparatively conservative in high dimensions. Consequently, the Hoeffding cushion often achieves smaller

cushions, fewer false positives, and higher bandwidth savings on modern, high-dimensional embeddings, while exposing the single, interpretable knob δ to balance recall against efficiency.

3.2 Candidate Point Early Rejection: Euclidean Distance

As with cosine similarity, we derive four classes of early-rejection cushions for Euclidean distance: conservative global bounds (ℓ_1, ℓ_2), a *sign-aware* bound that exploits the one-sided nature of truncation errors, and a *Hoeffding-based* probabilistic cushion that offers a tunable balance between recall and bandwidth savings.

Let $q \in \mathbb{R}^D$ be the query vector and $c \in \mathbb{R}^D$ a candidate. We first fetch a reduced-precision prefix $\tilde{c}$ by mantissa truncation and write

$$c \;=\; \tilde{c} + \varepsilon, \qquad \varepsilon = (\varepsilon_1, \ldots, \varepsilon_D). \tag{22}$$

The approximate and exact Euclidean distances are

$$\hat{d}_E \;=\; \|q - \tilde{c}\|_2, \tag{23}$$

$$d_E \;=\; \|q - c\|_2 \;=\; \|(q - \tilde{c}) - \varepsilon\|_2. \tag{24}$$

Each coordinate error is bounded by

$$|\varepsilon_i| \;\le\; \Delta_i, \tag{25}$$

where Δ_i is the ULP bound determined by the exponent of c_i and the kept mantissa length (Section 3.1.2). In the derivations below we work with *squared* distances to avoid square roots and to obtain additive cushions; thresholds are squared accordingly. Our objective is a *safe lower bound* on d_E (or d_E^2) that exceeds the heap threshold, yielding early rejection without fetching the truncated remainders. Sections 3.2.1–3.2.3 instantiate these ideas for the ℓ_1/ℓ_2, sign-aware, and Hoeffding cushions.

3.2.1 Deterministic ℓ_1 and ℓ_2 Cushions. To safely reject candidates under Euclidean distance using reduced-precision data, we bound the deviation introduced by the truncation error ε in (22). Working with *squared* distance, expand

$$d_E^2 \;=\; \|a - \varepsilon\|_2^2 \;=\; \|a\|_2^2 + \|\varepsilon\|_2^2 - 2\,a^\top \varepsilon \;\ge\; \hat{d}_E^2 \;-\; 2\,|a^\top \varepsilon|, \tag{26}$$

where $a = q - \tilde{c}$ and $\hat{d}_E = \|a\|_2$ as in (23). Thus a *lower bound* on d_E^2 follows from any upper bound on $|a^\top \varepsilon|$.

Bounding the cross term gives two conservative cushions. By Hölder's inequality,

$$|a^\top \varepsilon| \;\le\; \sum_{i=1}^{D} |a_i|\,|\varepsilon_i| \;\le\; \sum_{i=1}^{D} |a_i|\,\Delta_i \;=:\; B_{\ell_1}, \tag{27}$$

and from (26) we obtain the *zero-miss* rejection test

$$\text{Reject if} \quad \hat{d}_E^2 \;-\; 2\,B_{\ell_1} \;>\; \tau_E^2, \qquad \tau_E \;=\; d_E(q, w). \tag{28}$$

Alternatively, the reverse triangle inequality yields

$$d_E \;=\; \|a - \varepsilon\|_2 \;\ge\; \|a\|_2 - \|\varepsilon\|_2 \;\ge\; \hat{d}_E \;-\; B_{\ell_2},\; B_{\ell_2} \;=\; \|\Delta\|_2, \tag{29}$$

which implies the (unsquared) rejection test

$$\text{Reject if} \quad \hat{d}_E \;-\; B_{\ell_2} \;>\; \tau_E. \tag{30}$$

Discussion. Both ℓ_1 and ℓ_2 cushions are safe by construction (no false negatives). The ℓ_1 form adapts to the per-coordinate residual magnitudes $|a_i|$ and typically tightens when the residual is concentrated on few coordinates or when Δ_i varies across exponents; the ℓ_2 form depends only on Δ (hence only on exponents and M') and is query/candidate independent but often looser in practice. As in the cosine case, these global cushions are intentionally conservative; Section 3.2.2 tightens them using one-sided (sign-aware) structure of truncation errors, and Section 3.2.3 introduces a tunable Hoeffding cushion that is especially effective at large D.

3.2.2 Sign-Aware Deterministic Cushion. Mantissa truncation moves each coordinate toward zero without flipping its sign, therefore the per–coordinate error is one–sided:

$$\varepsilon_i \in \begin{cases} [0, \Delta_i], & \tilde{c}_i \ge 0, \\ [-\Delta_i, 0], & \tilde{c}_i < 0, \end{cases} \tag{31}$$

where Δ_i is the ULP bound from Section 3.1.2. Let $a = q - \tilde{c}$ and set $s_i = \operatorname{sign}(\tilde{c}_i) \in \{-1, 0, +1\}$; define the aligned residual $b_i = a_i s_i$ to indicate whether a_i points *with* ($b_i > 0$) or *against* ($b_i < 0$) the allowable direction.

Per-coordinate minimum via one-sided projection. For each i, the admissible error set is the interval I_i in (31). Since $(a_i - \varepsilon_i)^2$ is convex in ε_i, the minimum over I_i is attained at the Euclidean projection $\varepsilon_i^\star = \operatorname{proj}_{I_i}(a_i) = \min(\max(a_i, \ell_i), u_i)$ for $I_i = [\ell_i, u_i]$. Equivalently,

$$\varepsilon_i^\star \;=\; s_i \cdot \min\big(\max(b_i,\, 0),\, \Delta_i\big), \tag{32}$$

which yields the exact per-coordinate minimum contribution to the *squared* distance:

$$\ell_i(a_i, \Delta_i, s_i) = \begin{cases} a_i^2, & b_i \le 0, \\ 0, & 0 < b_i < \Delta_i, \\ (|a_i| - \Delta_i)^2, & b_i \ge \Delta_i. \end{cases} \tag{33}$$

Summing over coordinates gives a safe lower bound

$$L_{\text{coord}}^{\text{sign-aware}}(q, \tilde{c}, \Delta) \;=\; \sum_{i=1}^{D} \ell_i(a_i, \Delta_i, s_i) \;\le\; d_E(q, c)^2. \tag{34}$$

Let $\tau_E = d_E(q, w)$ denote the Euclidean heap threshold. The *zero–miss* rejection test is then

$$\text{Reject if} \qquad L_{\text{coord}}^{\text{sign-aware}}(q, \tilde{c}, \Delta) \;>\; \tau_E^2. \tag{35}$$

Tightness. The symmetric model permits $\varepsilon_i \in [-\Delta_i, \Delta_i]$, while the one–sided set I_i in (31) is a subset. By convexity,

$$\min_{\varepsilon_i \in I_i} (a_i - \varepsilon_i)^2 \;\ge\; \min_{\varepsilon_i \in [-\Delta_i, \Delta_i]} (a_i - \varepsilon_i)^2, \tag{36}$$

so (34) dominates the symmetric coordinate-wise bound. The gain is largest when many $b_i \le 0$ (residuals point against the allowable direction), where $\ell_i = a_i^2$ and cancellation is impossible. Consequently, the sign-aware bound is stricter than the global ℓ_1/ℓ_2 cushions in common cases, yet preserves the zero-miss guarantee.

3.2.3 Hoeffding-Based Probabilistic Cushion. To enable more aggressive pruning, we treat the linear error term in the squared-distance expansion

$$d_E^2 \;=\; \hat{d}_E^2 \;+\; \|\varepsilon\|_2^2 \;-\; 2\sum_{i=1}^{D} a_i\varepsilon_i, \tag{37}$$

as a bounded random sum. Since each summand obeys $|a_i\varepsilon_i| \le |a_i|\,\Delta_i$, Hoeffding's inequality yields

$$\Pr\left\{\sum_{i=1}^{D} a_i\varepsilon_i \;\ge\; t\right\} \;\le\; \exp\left(-\frac{t^2}{2\sum_{i=1}^{D}(a_i\Delta_i)^2}\right). \tag{38}$$

Setting the right-hand side to a user-specified level $\delta \in (0,1)$ and solving for t gives the *Hoeffding cushion*

$$t_a(\delta) \;=\; \sqrt{2\log(1/\delta)\,\sum_{i=1}^{D}(a_i\Delta_i)^2}\,. \tag{39}$$

Substituting into (37) provides the lower bound

$$d_E^2 \;\ge\; \hat{d}_E^2 \;-\; 2\,t_a(\delta), \tag{40}$$

and hence the rejection test

$$\text{Reject if} \quad \hat{d}_E^2 \;-\; 2\,t_a(\delta) \;>\; \tau_E^2. \tag{41}$$

Interpretation and scaling. Because truncation makes ε_i one-sided and coupled to $\tilde{c}_i$, δ should be viewed as a *tuning knob* rather than a strict tail probability: smaller δ enlarges the cushion (safer, less pruning), larger δ tightens it (more pruning, nonzero false-negative rate). For fixed M' and typical high-dimensional settings, $\Delta_i = \Theta(2^{-M'})$ up to exponent factors and $\mathbb{E}[a_i^2] = \Theta(1)$ after centering by $\tilde{c}$, giving the approximation

$$t_a(\delta) \;\approx\; 2^{-M'}\sqrt{\frac{2\log(1/\delta)}{D}}, \tag{42}$$

which shrinks as $D^{-1/2}$. In contrast, the worst in-heap Euclidean threshold scales as $\tau_E = \Theta(\sqrt{D})$ (up to dataset-dependent norms). Thus the gap between the cushion and the threshold widens with D, and the Hoeffding test in (41) retains strong pruning power where global deterministic cushions become uninformative.

In summary, the Euclidean Hoeffding cushion supplies a single, interpretable parameter δ to navigate the recall–bandwidth trade-off, complements the zero-miss deterministic cushions, and is particularly effective for modern high-dimensional embeddings.

Summary of Early-Rejection Cushions (Cosine and Euclidean). Table 2 summarizes the forms, rejection tests, safety guarantees, and per-candidate costs for all cushions used in this paper. Symbols: $\hat{d}_S = \langle q, \tilde{c}\rangle$, $\tau_S = d_S(q,w)$, $\hat{d}_E = \|q-\tilde{c}\|_2$, $\tau_E = d_E(q,w)$, $a = q - \tilde{c}$, Δ_i from Section 3.1.2, $B_{\ell_1} = \sum_i |a_i|\Delta_i$, $B_{\ell_2} = \|\Delta\|_2$, U and L as in Section 3.1.4, $L_{\text{coord}}^{\text{sign-aware}}$ as in Section 3.2.2, $t_q(\delta)$ and $t_a(\delta)$ as in Section 3.1.5 and 3.2.3.

Table 2: Cushion summary: Form, Rejection Test, Safety, and Per-candidate Cost.

Metric	Cushion (form)	Rejection test	Safety	Per-candidate cost
Cosine	ℓ_1: $B_{\ell_1}(q,\Delta) = \sum_i \lvert q_i\rvert\Delta_i$	$\hat{d}_S + B_{\ell_1} < \tau_S$	Zero-miss	1 pass: sum $\lvert q_i\rvert\Delta_i$
Cosine	ℓ_2: $B_{\ell_2}(\Delta) = \lVert\Delta\rVert_2$	$\hat{d}_S + B_{\ell_2} < \tau_S$	Zero-miss	1 pass: sum Δ_i^2, 1 sqrt
Cosine	Sign-aware: $U = \sum_{i\in\mathcal{P}} \lvert q_i\rvert\Delta_i$	$\hat{d}_S + U < \tau_S$	Zero-miss	1 pass with sign mask
Cosine	Hoeffding: $t_q(\delta) = \sqrt{2\log(1/\delta)\sum_i (q_i\Delta_i)^2}$	$\hat{d}_S + t_q(\delta) < \tau_S$	Tunable (δ)	1 pass: sum $(q_i\Delta_i)^2$
Euclidean	ℓ_1: $B_{\ell_1} = \sum_i \lvert a_i\rvert\Delta_i$	$\hat{d}_E^2 - 2B_{\ell_1} > \tau_E^2$	Zero-miss	1 pass: sum $\lvert a_i\rvert\Delta_i$
Euclidean	ℓ_2: $B_{\ell_2} = \lVert\Delta\rVert_2$	$\hat{d}_E - B_{\ell_2} > \tau_E$	Zero-miss	1 pass: sum Δ_i^2, 1 sqrt
Euclidean	Sign-aware: $L_{\text{coord}}^{\text{sign-aware}}$	$L_{\text{coord}}^{\text{sign-aware}} > \tau_E^2$	Zero-miss	1 pass: piecewise per coord.
Euclidean	Hoeffding: $t_a(\delta) = \sqrt{2\log(1/\delta)\sum_i (a_i\Delta_i)^2}$	$\hat{d}_E^2 - 2t_a(\delta) > \tau_E^2$	Tunable (δ)	1 pass: sum $(a_i\Delta_i)^2$

3.3 Disaggregated Data Placement

Precision-on-demand only translates into *real* bandwidth savings if the memory layout supports selective-precision access. A naive approach is to store two copies of every vector, one full-precision for exact evaluation and one reduced-precision for early rejection. While flexible, this notably increases the memory consumption and can be prohibitive at billion–trillion–vector scale. A more economical idea is to store only full-precision vectors and derive a reduced-precision prefix on demand. However, commodity DRAM operates at coarse granularity: controllers issue fixed *burst* transfers (typically 8–16 contiguous bytes) within pages of size 2–8 KB [26]. Consequently, requesting "just a few high-order bits" still pulls entire FP16/FP32 words over DRAM, nullifying apparent savings unless the *layout* itself is changed. We therefore redesign the placement so that partial-precision reads map to fewer DRAM bytes.

To realize actual bandwidth savings, we adopt a *disaggregated (bit–plane)* layout that stores each floating-point value as separate components, sign, retained upper bits, and truncated remainder, so they can be fetched independently. Let an n-bit floating-point number be written as $[b_n, b_{n-1}, \ldots, b_1]$, where b_n is the sign bit, $[b_{n-1}, \ldots, b_{n-m}]$ encodes the m-bit exponent, and the remaining $n-m-1$ bits form the mantissa. For reduced precision, we truncate the k least-significant mantissa bits, and represent coordinate j as

$$\varphi^{(j)} \;=\; \big(b_n^{(j)},\; \omega^{(j)},\; \nu^{(j)}\big), \tag{43}$$

where $b_n^{(j)}$ is the sign bit, $\omega^{(j)}$ collects the $(n-k-1)$ retained upper bits (sign + exponent + high-order mantissa), and $\nu^{(j)}$ is the k-bit remainder. These three components are stored in *separate, contiguous pages across dimensions* (structure-of-arrays), as illustrated in Fig. 1. For a D-dimensional vector $c = [\varphi^{(1)}, \ldots, \varphi^{(D)}]$, the layout becomes:

- a *sign-bit page*: $[\, b_n^{(1)}, \ldots, b_n^{(D)} \,]$,
- an *upper-bit (reduced-precision) page*: $[\, \omega^{(1)}, \ldots, \omega^{(D)} \,]$,
- a *remainder page*: $[\, \nu^{(1)}, \ldots, \nu^{(D)} \,]$.

Burst fetches now land entirely within a single bit-plane page, so issuing only the sign/upper-bit pages in the first stage directly reduces DRAM bytes; the remainder page is touched only for survivors.

During the first-stage screening, we fetch only the *sign-bit* and *upper-bit* pages; the *remainder* page is touched only if a candidate survives and requires full-precision evaluation. As a result, the bytes transferred scale with the precision actually consumed (sign + exponent + retained mantissa bits), rather than with a fixed FP16/FP32 word size. Because each component is stored contiguously across dimensions, burst transfers land within a single bit-plane page, improving burst efficiency and avoiding fetches that straddle unrelated words. The separation also aligns with modern memory controllers that can issue concurrent bursts to different pages/banks, enabling parallel activation of sign/upper-bit pages while deferring remainder traffic.

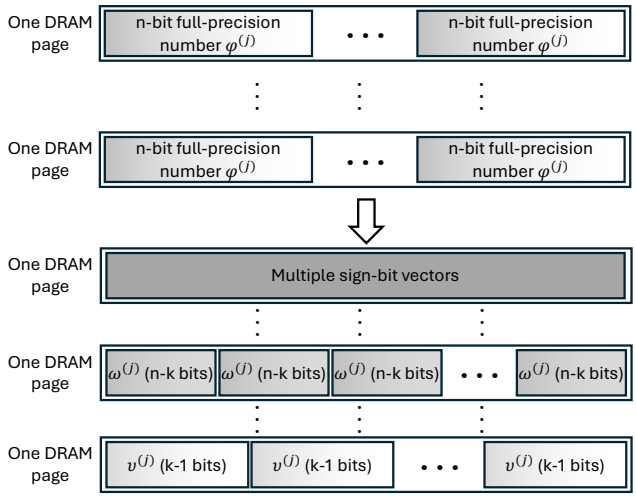

Figure 1: Transition from naive contiguous placement to disaggregated placement, enabling selective precision fetches.

To further amplify savings, we apply *lossless compression* to the upper-bit pages $[\,\omega^{(1)}, \ldots, \omega^{(D)}\,]$. These retained bits are often highly correlated across dimensions (e.g., shared exponent ranges and similar high-order mantissa patterns), which yields strong compressibility. Modern hardware makes decompression inexpensive: recent CPUs (e.g., Intel Sapphire Rapids with QuickAssist) and GPUs (e.g., NVIDIA Blackwell) provide native block decompression that operates at memory-throughput rates, so the overhead is negligible relative to the bytes saved. Because ANN corpora are typically static or slowly changing, compression can be performed offline and amortized over many queries. For random access, upper-bit pages can be compressed in fixed-size chunks (e.g., 4–16 KB) with per-chunk offsets, so a first-stage fetch decompresses only the relevant chunk rather than an entire page.

Compression effectiveness depends strongly on layout. Storing each $\omega^{(j)}$ contiguously (array-of-structures) limits the redundancy visible to a compressor. We therefore introduce *bit-wise shuffling*: each upper-bit page is reorganized into $(n-k)$ aligned *slices*, as illustrated in Fig. 2, where slice r contains the r-th most significant bit from all D dimensions. This structure-of-arrays interleaving clusters statistically similar patterns (e.g., shared exponents and correlated high-order mantissae) that conventional compressors exploit via entropy coding and dictionary/byte deduplication.

4 Evaluation

We evaluate our design under both cosine similarity and Euclidean distance across eight representative corpora (Table 3), spanning token-level word vectors, document/passage embeddings, and image descriptors. All vectors use FP16 (1-bit sign, 5-bit exponent, 10-bit mantissa). We vary the number of truncated mantissa bits $t \in \{0, \ldots, 10\}$ and report two outcomes: Recall@K and the per-candidate false-positive rate (FPR) p_{fp}. Unless stated otherwise, queries are hot in cache and only candidate vectors are read from

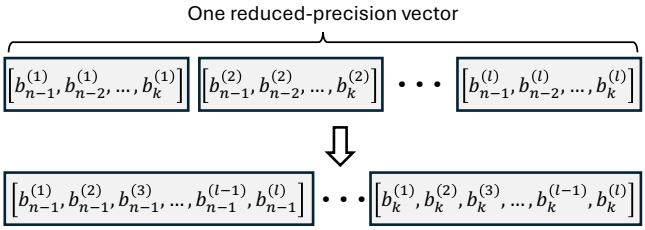

Figure 2: Bit-wise shuffling reorganizes reduced-precision vectors into aligned slices across dimensions, improving compressibility.

DRAM. For the Hoeffding cushions we select δ from a logarithmic grid to achieve high recall (typically $\geq 90\%$) while providing noticeable bandwidth savings.

Table 3: Datasets used in evaluation.

Cosine Similarity		
Dataset	**Type**	**Dimension**
GloVe	Token-level word vectors (co-occurrence based)	200
WikiNews	News articles (document embeddings)	300
FineWeb	Web pages (general-domain document embeddings)	1,024
MS MARCO	IR passages (dense passage embeddings)	3,072
20 Newsgroups	Usenet posts (topic-oriented document embeddings)	3,072
DBPEDIA	Wikipedia entity abstracts (KB-text document embeddings)	3,072
Euclidean Distance		
Dataset	**Type**	**Dimension**
SIFT	Local image keypoint descriptors	128
GIST	Global image/scene descriptors	960
GloVe†	Token-level word vectors	200

† GloVe is evaluated under both cosine similarity and Euclidean distance.

Bytes-per-query model and projected throughput. To visualize the first-stage effect of truncation, we use a simple per-candidate accounting model:

$$\text{BW}_s \;\approx\; \frac{t}{16} \;-\; p_{\text{fp}}, \tag{44}$$

where $t/16$ is the fraction of *mantissa* bits skipped (relative to FP16) and p_{fp} is the fraction of candidates that still trigger a full-precision fetch.[1] When distance evaluation is bandwidth-bound, reducing bytes per query by a fraction $s := \text{BW}_s$ ideally improves throughput by QPS $\text{gain}_{\text{ideal}} \approx 1/(1-s)$. In mixed regimes where only a fraction f of query time is bandwidth-bound, the overall speedup is $1/((1-f)+f(1-s))$.

Section 3 introduced four early-rejection mechanisms applicable to both metrics: a probabilistic Hoeffding cushion with tunable miss parameter δ, an ℓ_1 cushion, an ℓ_2 cushion, and a sign-aware cushion. Diagnostic results in Fig. 3 show that the global ℓ_1 and ℓ_2 cushions suffer consistently high FPR once more than six mantissa bits are truncated. For example, on MS MARCO with eight bits removed, ℓ_1 and ℓ_2 reach 12% and 25% FPR, whereas the Hoeffding and sign-aware methods remain well below 5%. Such inflated FPRs imply poor bandwidth savings under (44); accordingly, the remainder

[1] Eq. (44) abstracts away fixed sign/exponent bits and burst/alignment effects; Section 3.3 explains how the bit-plane layout makes partial-precision reads translate into fewer DRAM bytes.

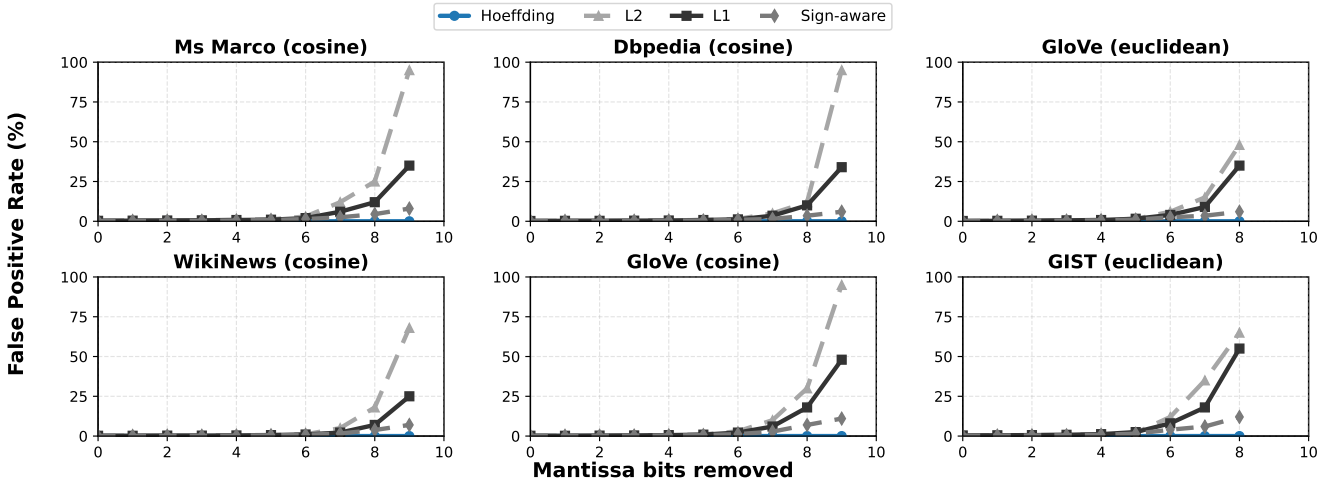

Figure 3: Measured false positive rate while removing different number of mantissa bits.

of our evaluation focuses on the sign-aware deterministic rule (zero-miss, but with higher FPR) and the Hoeffding cushion (a single-parameter trade-off between recall and bandwidth). In the following subsections we evaluate these two methods under both cosine similarity and Euclidean distance.

4.1 Cosine Similarity Evaluation

Figs. 4–9 present the recall vs. memory bandwidth saving and false positive rate vs. mantissa bits removed for the six cosine datasets listed in Table 3. We begin with the sign-aware method, which by construction never excludes a true top-K neighbor: its inflated, sign–aware bound guarantees 100% recall across all datasets. This guarantee, however, comes at a cost. As truncation grows (t increases), the cushion loosens, admitting more borderline non-neighbors and driving the false positive rate upward. As a result, the memory bandwidth saving BW_s plateaus around 0.4−0.6, with vertical markers in the plots indicating the best observed operating points. In contrast, Hoeffding-based methods often achieve higher bandwidth savings by tolerating small recall losses. The variation in their effectiveness across datasets stems from the size of the cushion, $U(q, \tilde{c}, \Delta) = \sum_{i:\,\mathrm{sign}(q_i)=\mathrm{sign}(\tilde{c}_i)} |q_i|\,\Delta_i$,, which grows large when query and database coordinates are strongly sign-aligned or when the maximum per-coordinate truncation error is high (i.e., under aggressive mantissa truncation or coarse exponent scaling).

Compared with the sign-aware method, Hoeffding-based method provides a tunable probabilistic cushion governed by the parameter δ. Smaller δ widens the bound, preserving recall but weakening pruning and hence increasing false positive rate and reducing memory bandwidth saving; larger δ tightens the cushion, reducing false positive rate and raising bandwidth savings while introducing certain recall loss. The qualitative effect of δ is consistent across datasets, though the appropriate range of parameter δ depends on the vector dimensionality and inter-vector distance distribution. Therefore, the reported results use distinct δ values for each dataset, chosen by grid search over logarithmically spaced values (10^{-12} to 10^{-1}) to achieve relatively high recall (at least 90%). As

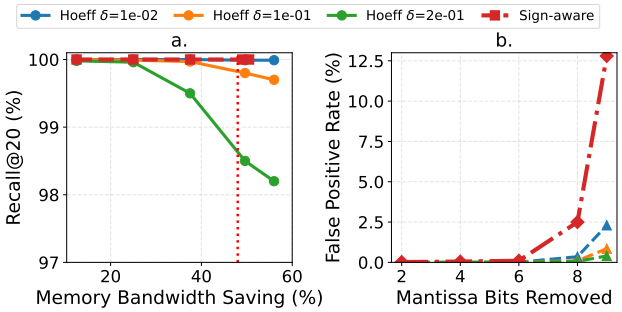

Figure 4: GloVe (cosine). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

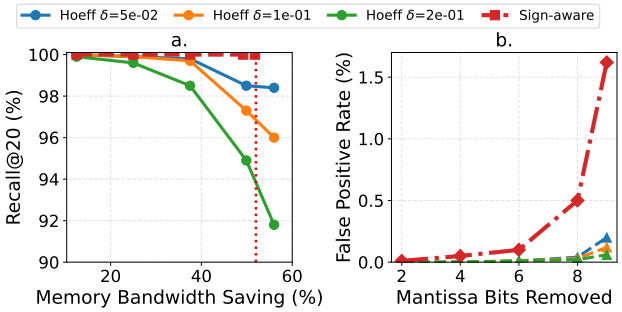

Figure 5: FineWeb (cosine). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

mentioned earlier, the structure of the datasets plays a crucial role, and we highlight two structural effects that shape performance. First, the *dimension effect*: high-dimensional document embeddings (MS MARCO, 20 Newsgroups, DBPEDIA at 3,072D; FineWeb at

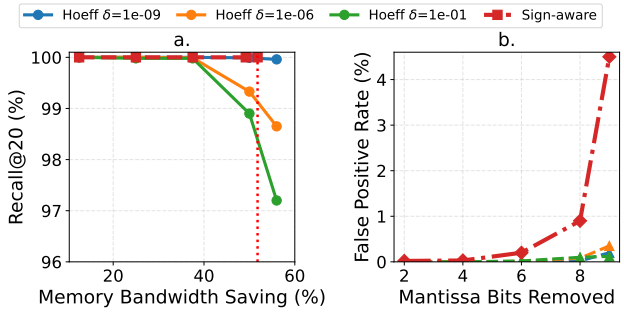

Figure 6: MS MARCO (cosine). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

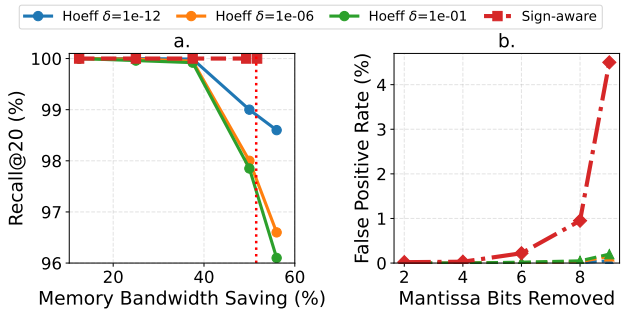

Figure 7: 20 Newsgroups (cosine). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

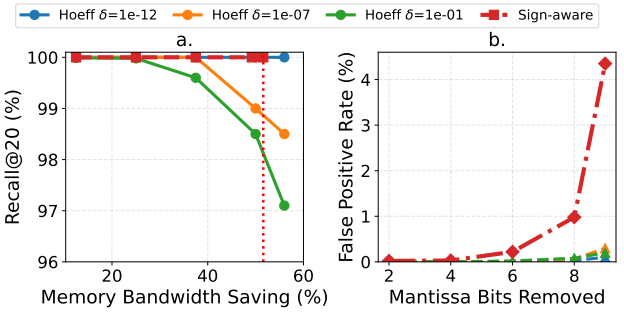

Figure 8: DBPEDIA (cosine). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

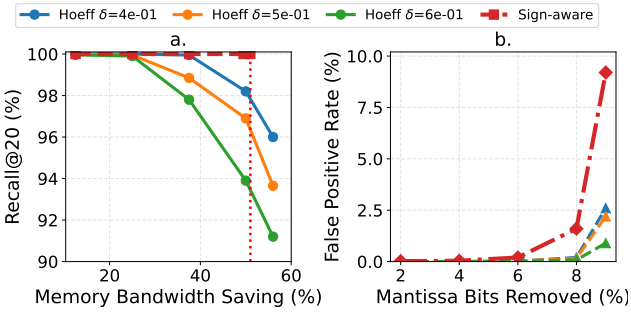

Figure 9: WikiNews (cosine). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

1,024D) sustain higher recall and lower false positive rate at the same $\{t, \delta\}$ than lower-dimensional word or short–text embeddings. For example, on WikiNews (~300D), Hoeffding achieves ~ 90% recall at ~ 60% memory bandwidth saving (removing $t = 9$ bits) with false positive rate of ~ 2% (Fig. 9), whereas DBPEDIA (3,072D) achieves both higher recall and lower false positive rate under comparable truncation (Fig. 8). This effect can be explained as follows: In higher-dimensional embeddings, each coordinate contributes only a small part to the total similarity. Truncation noise in individual coordinates tends to *average out*, so the overall error is smaller relative to the true distance. This makes pruning rules more reliable, yielding lower false positives and better recall at the same truncation level. In other words, high dimensions act like natural noise-cancelers: random errors get diluted when spread across many coordinates. Second, the *margin effect*: datasets with clearer separation between true neighbors and distractors tolerate larger δ without recall loss. For instance, while 20 Newsgroups has low dimensionality, it also has a low margin effect, which allows it to achieve high recall rates with low false positive rates. The margin is the gap between a true neighbor's similarity score and that of its closest distractor. When this gap is large, even approximate or noisy distance estimates are sufficient to distinguish the true neighbor. But when the margin is small, small truncation errors can flip the ordering, leading to false positives or recall loss.

In summary, the cosine experiments suggest a practical rule of thumb. The sign-aware method is appropriate when recall must remain perfect, since its deterministic bound guarantees that no true top-K neighbor is ever lost. However, this guarantee comes with a steep trade-off: bandwidth savings saturate early, and false positive rates grow quickly with aggressive truncation. Hoeffding cushions, by contrast, expose a tunable parameter that consistently delivers higher savings whenever small, controllable recall loss is acceptable. The effectiveness of Hoeffding scales especially well in high-dimensional embeddings, where truncation errors average out, and in datasets with strong neighbor–distractor separation, where larger margins make pruning more reliable. These findings imply that system designers can tailor early-rejection strategies to application needs: sign-aware cushions are best for strict-recall scenarios such as safety-critical retrieval, while Hoeffding cushions offer a flexible knob for trading minimal accuracy against substantial bandwidth efficiency in large-scale search engines or recommendation systems.

4.2 Euclidean Distance Evaluation

We now turn to Euclidean distance on three datasets: GIST, GloVe, and SIFT. Figs. 10–12 present recall vs. memory bandwidth saving and false positive rate vs. mantissa bits removed, comparing the sign-aware rule and the Hoeffding cushion. The sign-aware

method leverages the one-sided nature of truncation errors under Euclidean distance, which ensures that no true top-K neighbor is ever discarded. This guarantee is evident in Figs. 10a–12a, where recall remains identically 100% across all datasets. However, this conservatism comes at a cost. As mantissa truncation becomes more aggressive, the bound loosens, allowing distractors to pass through and rapidly driving up false positive rates. The severity of this increase varies by dataset. For instance, both GloVe and SIFT reach at most $\sim$ 50% bandwidth saving under sign-aware, but their false positive rate profiles diverge: in GloVe, FPR remains under 2%, while in SIFT it escalates past 10%, as shown in Figs. 11b and 12b.

Hoeffding provides the tunable alternative. Its probabilistic cushion is governed by the parameter δ, which directly controls the aggressiveness of pruning. With small δ, the cushion widens, ensuring that recall remains nearly perfect but at the cost of weaker pruning and lower bandwidth savings. As δ increases, the cushion tightens: more candidates are rejected early, memory traffic decreases, and bandwidth savings improve, though at the expense of a controlled loss in recall. Intuitively, δ acts as a single scalar knob that lets system designers navigate the trade-off space between accuracy and efficiency: conservative settings resemble the behavior of sign-aware, while larger values push toward aggressive pruning with higher savings. As in the cosine case, we tune δ separately for each dataset using a logarithmic grid search to maintain recall above 90%. This procedure reflects the fact that the optimal operating point depends strongly on dataset structure, including dimensionality and neighbor separation, and highlights the flexibility of Hoeffding cushions in adapting to diverse workloads.

Two structural effects again emerge. First, the *dimension effect*: higher-dimensional datasets with relatively independent coordinates average truncation noise across many dimensions, yielding smoother curves and typically lower false positive rates for the same t. For example, although GIST has a nominal dimensionality of 960, its heavy redundancy reduces the effective dimension, weakening this averaging benefit. As a result, GIST exhibits substantially higher false positive rate than GloVe (200D), despite its larger dimension (see Figs. 10b and 11b). Second, the *margin effect*: datasets with stronger separation between true neighbors and distractors can tolerate larger δ without recall loss. SIFT, for instance, maintains high recall with relatively low false positive rate under aggressive pruning, whereas tighter-margin datasets such as GIST incur much higher false positive rate to achieve comparable recall (see Figs. 12b vs. 10b).

In summary, Euclidean results reinforce the cosine story. The sign-aware method is suitable when absolute recall is mandatory, but it saturates bandwidth savings early and suffers runaway false positive rate. Hoeffding offers the only meaningful trade-off lever: by tolerating small, controllable recall loss, it consistently delivers higher savings and lower false positives.

4.3 Lossless Compression Results

Finally, Fig. 13 presents the measured compression ratios achieved by applying GZIP and ZSTD to the *bit-wise shuffled upper-bit pages* (sign/exponent + retained mantissa) of each candidate's reduced-precision vector. Across all six datasets, the compression ratio[2] ranges from approximately 1.4× to 2.0×, with ZSTD slightly outperforming GZIP in most cases and averaging around 1.8×. This translates into as much as $\sim$ 45% additional reduction for the first-stage fetch of reduced-precision data.

The benefit arises because exponent ranges and high-order mantissa patterns are correlated across dimensions. Bit-wise shuffling makes these correlations explicit by aligning statistically similar bits across dimensions, creating long runs and repeated symbols that

[2]Compression ratio is defined as (uncompressed bytes)/(compressed bytes).

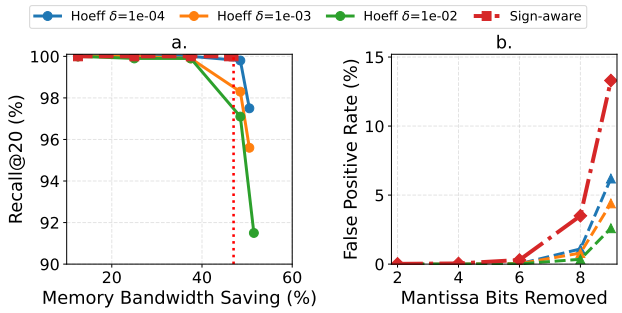

Figure 10: GIST (Euclidean). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

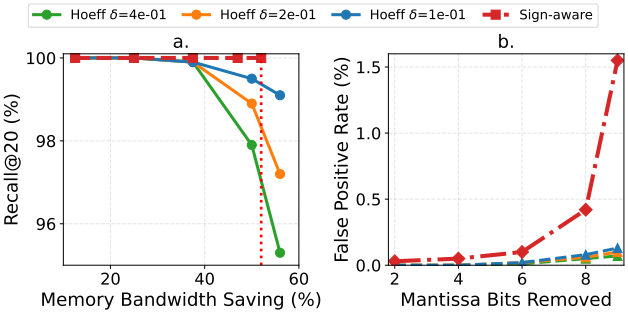

Figure 11: GloVe (Euclidean). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

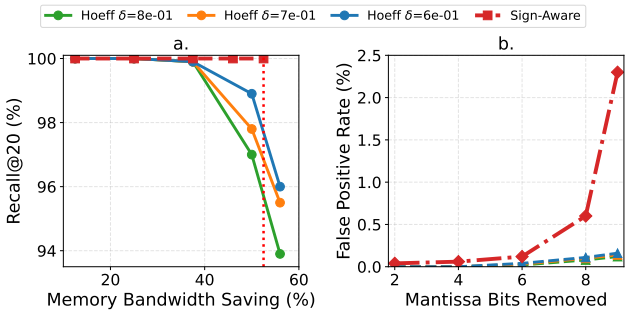

Figure 12: SIFT (Euclidean). Normalized Recall@20 vs. bandwidth saving (left) and false positive rate vs. mantissa bits removed (right).

standard dictionary/entropy encoders exploit. In practice, reduced-precision vectors—already smaller due to truncation—become even more lightweight when paired with lossless block compression. Equally important, this stage integrates naturally with existing hardware: modern CPUs (e.g., Intel Sapphire Rapids with QuickAssist) and GPUs (e.g., NVIDIA Blackwell) support hardware-assisted block decompression at memory-throughput rates, so decode overhead is negligible relative to the bytes saved. Because ANN corpora are typically static or slowly changing, compression can be performed offline and amortized over many queries.

Taken together, lossless compression complements the disaggregated placement and early-rejection design. If early rejection reduces first-stage bytes by a fraction s (e.g., s=0.60) and the shuffled upper-bit pages compress by a factor R (e.g., $R\approx$1.8), the combined reduction is

$$s_{\text{total}} \;=\; 1 - \frac{1-s}{R} \quad\Rightarrow\quad s_{\text{total}} \approx 1 - \frac{0.40}{1.8} \approx 0.78,$$

placing aggregate savings in the $\sim$ 70–80% range. This layered strategy highlights a key advantage of a bandwidth-centric approach: orthogonal optimizations compose cleanly to deliver efficiency gains beyond what any single technique achieves in isolation.

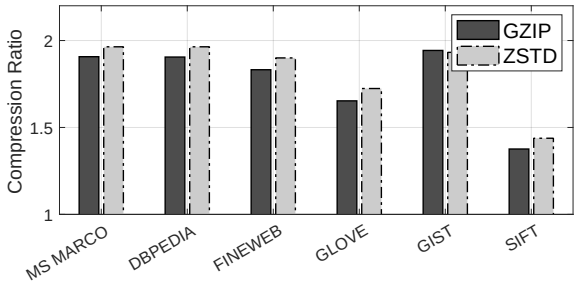

Figure 13: Compression ratio of the bit-wise shuffled exponents.

Computation overhead. Cushion evaluation adds only a *single streaming pass* of simple accumulations per candidate—the same pass that forms $\hat{d}$. For cosine, ℓ_1 and sign-aware accumulate terms in $|q_i|$; ℓ_2 does not depend on q (only on Δ); and the Hoeffding cushion accumulates $(q_i\Delta_i)^2$ (optionally reusing a cached q_i^2 vector per query). For Euclidean, ℓ_1 uses $|a_i| = |q_i - \tilde{c}_i|$, ℓ_2 again depends only on Δ, the sign-aware form applies the piecewise rule in Eq. (33), and the Hoeffding cushion accumulates $(a_i\Delta_i)^2$. In all cases the arithmetic intensity remains a small *constant* ($O(1)$), so *bytes-per-query reduction* dominates end-to-end impact in bandwidth-bound regimes (cf. Eq. (44)).

5 Future Work

We highlight three concrete directions that build directly on this study, targeting improved recall–bandwidth control, tighter theoretical guarantees, and end-to-end integration in vector databases.
• *Adaptive precision control.* Develop online policies that set (M', t, δ) per query (and per collection) using lightweight statistics such as margin estimates and exponent histograms. The goal is to meet target recall bands while minimizing bytes per query and keeping tail latency within SLOs.
• *Sharper cushions with data awareness.* Investigate data-aware bounds (e.g., Bernstein/Bentkus-style) that leverage observed error structure and anisotropy to tighten cushions without sacrificing transparent guarantees. A key metric is reduction in cushion size at fixed recall.
• *Index- and system-level integration.* Integrate precision-on-demand with common indices (HNSW/IVF) and a vector-DB stack, aligning posting/graph layouts with bit-plane pages to realize DRAM-byte savings end to end. A prototype should report wall-clock QPS/latency, hardware DRAM counters, and the overhead (or benefit) of hardware block decompression.

6 Conclusion

We presented a bandwidth-first, representation-preserving framework for ANN distance evaluation that reduces bytes per query without altering embeddings. The approach performs precision-on-demand with early rejection, supported by mathematically grounded cushions—conservative ℓ_1/ℓ_2 baselines, a sign-aware deterministic cushion with a zero-miss guarantee, and a Hoeffding-based probabilistic cushion with a single parameter δ to trade recall for efficiency—and extends classical error bounds and concentration tools to progressive similarity computation. To make savings materialize at DRAM granularity, we introduced a bit-plane layout with bit-wise shuffling and leveraged hardware-assisted lossless compression; across diverse datasets, early rejection alone reduces first-stage bytes by up to 60%, and compression adds roughly 1.4×–2.0× more, placing aggregate savings in the $\sim$ 70–80% range and mapping to throughput (QPS) gains in bandwidth-bound regimes. Because the method preserves floating-point vectors and is orthogonal to index choice (e.g., HNSW/IVF/PQ), it is drop-in for modern vector-database stacks and high-dimensional retrieval systems, and it motivates future work on adaptive precision control, data-aware cushions, and end-to-end index/system integration.

References

[1] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the twentieth annual symposium on Computational geometry*, 2004, pp. 253–262.
[2] H. Schütze, C. D. Manning, and P. Raghavan, *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008, vol. 39.
[3] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl, "Item-based collaborative filtering recommendation algorithms," in *Proceedings of the international conference on World Wide Web*, 2001, pp. 285–295.
[4] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The adaptive web: methods and strategies of web personalization*. Springer, 2007, pp. 291–324.
[5] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.
[6] Z. Jiang, F. F. Xu, L. Gao, Z. Sun, Q. Liu, J. Dwivedi-Yu, Y. Yang, J. Callan, and G. Neubig, "Active retrieval augmented generation," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, 2023, pp. 7969–7992.
[7] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, pp. 85–126, 2004.
[8] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM computing surveys (CSUR)*, vol. 41, no. 3, pp. 1–58, 2009.
[9] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the ACM symposium on Theory of computing*, 1998, pp. 604–613.

[10] Y. Malkov and D. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[11] S. Deegalla and H. Bostrom, "Reducing high-dimensional data by principal component analysis vs. random projection for nearest neighbor classification," in *International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2006, pp. 245–250.

[12] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification." *Journal of machine learning research*, vol. 10, no. 2, 2009.

[13] M. Du, S. Ding, and H. Jia, "Study on density peaks clustering based on k-nearest neighbors and principal component analysis," *Knowledge-Based Systems*, vol. 99, pp. 135–145, 2016.

[14] J. Gao and C. Long, "High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations," *Proceedings of the ACM on Management of Data*, vol. 1, no. 2, pp. 1–27, 2023.

[15] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 1, pp. 117–128, 2010.

[16] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

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

[18] M. Ding, Z. Zhang, Y. Huang, Y. Sun, Y. He, L. Song, and J. Wang, "Rabitq: Memory efficient and low-latency approximate nearest neighbor search via randomized bit quantization," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2024, pp. 1665–1679.

[19] J. Zhou, C. Song, D. Zhang, J. Li, J. Tang, W. Wang, Y. Li, B. Ding, and Q. Li, "Compressed indices for memory-efficient approximate nearest neighbor search," in *Proceedings of the VLDB Endowment*, vol. 16, no. 12, 2023, pp. 3497–3510.

[20] A. Kusupati, G. Bhatt, A. Rege, M. Wallingford, A. Sinha, V. Ramanujan, W. Howard-Snyder, K. Chen, S. Kakade, P. Jain, and A. Farhadi, "Matryoshka representation learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 233–30 249, 2022.

[21] H. Chen, P. Zhao, and M.-L. Zhang, "Reliable and efficient distance comparison for approximate nearest neighbor search," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2023, pp. 1792–1806.

[22] X. Wang, Z. Li, J. Luo, X. Chen, Y. Fu, Z. Li, S. Wang, B. Wang, C. Guo, and D. Lin, "Finger: Fast inference for graph-based approximate nearest neighbor search," in *Proceedings of the Web Conference (WWW)*, 2023, pp. 408–419.

[23] X. Zhang, S. Li, W. Wang, Y. Jiang, and G. Chen, "Probabilistic routing for fast graph-based approximate nearest neighbor search," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2024, pp. 41 415–41 428.

[24] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *Journal of the American Statistical Association*, vol. 58, no. 301, pp. 13–30, 1963.

[25] S. Boucheron, G. Lugosi, and P. Massart, *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

[26] B. Jacob, D. Wang, and S. Ng, *Memory systems: cache, DRAM, disk*. Morgan Kaufmann, 2010.