NORM-AWARE RESIDUAL HASHING FOR FAST NEAREST NEIGHBOR SEARCH

Anonymous authors
Paper under double-blind review

ABSTRACT

Approximate nearest neighbor (ANN) search in high-dimensional spaces plays an important role in modern AI applications. To reduce the cost of high-dimensional distance computations in ANN search, binary hashing represents vectors as compact binary codes and uses inexpensive Hamming distances to select candidates. However, we observe that vectors in several real-world datasets are concentrated within a narrow angular region. When such vectors are encoded, the resulting bits can be imbalanced and have low entropy, limiting the ability of binary codes to distinguish true neighbors from other candidates. Our experiments further show that more balanced bits alone do not ensure accurate L2 candidate ranking, since Hamming distance does not capture vector lengths. We propose Norm-Aware Residual Hashing (NARH), a mean-centered binary distance estimator designed to address these limitations. Our key idea is to center vectors before encoding to reduce angular concentration and make their directions more distinguishable. Since centering changes angles but preserves Euclidean distances, NARH uses angles estimated from centered binary codes and each vector's distance to the center to estimate original squared Euclidean distances. We further extend the estimator to inverted-file search with a separate mean for each posting list, using precomputed center projections to reduce expensive query-time projection computations. Our analysis provides a high-probability bound on squared-distance estimation error in terms of code length and residual norms, and a sufficient condition for preserving pairwise rankings. Experiments on four real-world datasets show that NARH reduces bit imbalance, improves recall at fixed reranking budgets, and achieves better search performance than binary hashing baselines based on random projection and Iterative Quantization (ITQ).

1 INTRODUCTION

K-nearest neighbor search in high-dimensional spaces (Indyk & Motwani, 1998) plays an important role in the era of AI, powering a wide range of applications, including recommendation systems (Liu et al., 2025; Yuan et al., 2025), image retrieval (Omama et al., 2025), and retrieval-augmented generation (RAG) (Asai et al., 2024) for large language models (Wang et al., 2018; Lewis et al., 2020). Data such as Text, images, audio, and video are mapped into high-dimensional vector spaces, where similar items have similar vector representations. Search is then performed on these vectors to retrieve similar data. However, the curse of dimensionality (Indyk & Motwani, 1998) makes finding exact nearest neighbors computationally expensive and can require scanning the entire dataset. Approximate nearest neighbor (ANN) search has therefore become a widely used solution in practice, sacrificing some accuracy to substantially reduce search costs.

Improving ANN search efficiency involves several factors, one of which is the cost of distance computation (Gao & Long, 2023; Song et al., 2025). Computing the distance between a query vector and a database vector is a basic operation in vector search, but it becomes expensive in high-dimensional spaces. To reduce this cost, a common strategy is to transform vectors into representations that allow cheaper distance estimation, use these estimates to select candidates, and then rerank the candidates using the original vectors. Hashing and quantization are two widely studied approaches to this strategy (Wang et al., 2018; Matsui et al., 2018). Its effectiveness depends on estimation accuracy: inaccurate estimates require larger candidate sets to retain true neighbors, and the resulting reranking cost can offset the savings from inexpensive comparisons.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

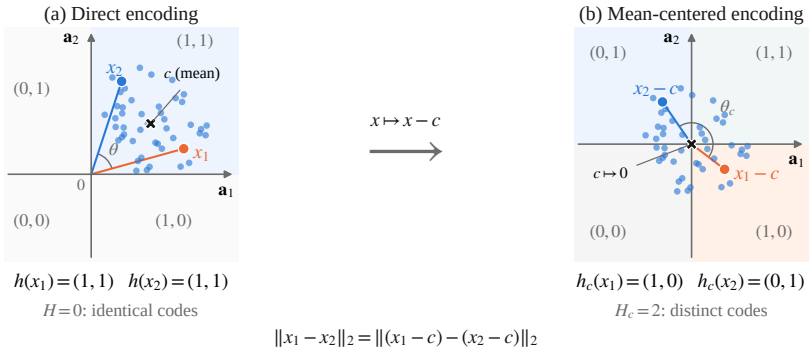

Figure 1: Effect of mean centering on binary codes. (a) Before centering, all points receive the same code. (b) After centering, the same projection directions produce distinct codes, while pairwise Euclidean distances remain unchanged.

Among hashing-based methods, binary hashing is particularly attractive: it maps high-dimensional vectors to binary codes—short sequences of zeros and ones—and uses Hamming distance to select candidates. These codes require little storage and support fast comparisons using XOR and popcount operations. SIMD (Single Instruction, Multiple Data) instructions can further accelerate these comparisons by processing multiple codes in parallel. The compact codes also reduce the amount of data loaded during candidate selection. To translate these computational advantages into faster search, the codes need to preserve similarity well enough to select a small but accurate candidate set. Binary hashing methods are commonly categorized as data-independent or data-dependent (Wang et al., 2018). Random-hyperplane locality-sensitive hashing (LSH) samples projection directions independently of the data (Charikar, 2002), whereas data-dependent methods such as Iterative Quantization (ITQ) learn a rotation from data to reduce binary quantization error (Gong et al., 2013).

Among these methods, random-hyperplane hashing (Charikar, 2002) is particularly attractive for its simplicity and its ability to estimate vector angles through Hamming distance. It generates binary codes by recording the signs of projections onto random directions. Nearby directions are likely to receive similar codes, and normalized Hamming distance is an unbiased estimator of the angle divided by π. Cai (2021) found that this method was among the leading performers in their comparison of hashing methods.

However, we observe that vectors in several real-world datasets point in similar directions, with pairwise angles concentrated within a small range. Applying random-hyperplane hashing directly to these vectors produces many bits that take the same value on most data points, resulting in *bit imbalance and low entropy*. Bits that take the same value on most data points contribute little to distinguishing those points through Hamming distance. With a limited number of bits, the codes may therefore fail to resolve the angular differences needed to distinguish true neighbors from other candidates. As a result, some true neighbors may be excluded from the candidate set when the reranking budget is limited and cannot be recovered by exact reranking. Improving bit balance alone, however, is insufficient for L2 search. Our controlled experiments show that the same binary codes can yield substantially different recall depending on whether candidate scoring accounts for vector lengths. Section 4 examines these limitations through a toy example, angle and bit statistics on four datasets, and controlled comparisons of candidate-scoring rules.

We propose *Norm-Aware Residual Hashing* (NARH) to address this problem. Our key idea is to reduce angular concentration and spread vector directions more widely, enabling binary codes to better distinguish true neighbors from other candidates. Specifically, we translate database and query vectors by subtracting a common center before applying random-hyperplane hashing. We use the arithmetic mean as a simple default center to reduce the shared directional component and make the directions after centering more distinguishable. On the datasets studied, this translation broadens the angular distributions and produces more balanced bits with higher entropy. Figure 1 illustrates this idea: centering preserves Euclidean distances while allowing the same projection directions to produce distinct binary codes. Since centering changes angles, we combine the angles estimated from binary codes with the distances of both vectors from the center to estimate their original Euclidean distance. This allows us to exploit the more distinguishable directions after centering while still finding nearest neighbors according to the original distances.

We implement NARH in exhaustive binary scanning (SCAN) and inverted-file search (IVF). SCAN uses a global mean as its center. IVF instead uses the mean of each posting list as a *list-local center* and scans only selected lists. Using these list-local centers introduces a query-time challenge: naively projecting $q - c_j$ separately for every probed list repeats an expensive matrix–vector multiplication. We avoid this cost through $A(q - c_j) = Aq - Ac_j$: projected centers Ac_j are computed and stored during index construction, and each query computes Aq only once. Each subsequent list requires only a vector subtraction and binary thresholding to obtain its query code.

Our contributions are threefold:

- **Analysis.** We examine directional concentration, bit imbalance, and low entropy on real-world datasets, and show through controlled experiments that higher bit entropy alone does not ensure accurate L2 candidate ranking. We also derive a finite-bit error bound for our distance estimator and a sufficient condition for preserving pairwise rankings.
- **Method.** We propose NARH, a mean-centered binary distance estimator designed to improve candidate ranking when vector directions are concentrated. We extend this estimator to IVF search with list-local centers, using precomputed center projections to reduce expensive query-time projection computations.
- **Experiments.** Experiments on four real-world datasets demonstrate that NARH improves search performance compared with binary hashing baselines based on random projection and ITQ.

2 Problem Setup and Preliminary Concepts

Notation. Let $X = \{x_i\}_{i=1}^N \subset \mathbb{R}^d$ denote a fixed database of N vectors with dimensionality d, and let $q \in \mathbb{R}^d$ be a query. We use B for the binary code length, K for the number of returned neighbors, and $L \geq K$ for the number of candidates retained for exact reranking. The Euclidean distance is denoted by $\delta(q, x) = \|q - x\|_2$; computations use its square $\delta^2(q, x) = \|q - x\|_2^2$. The indicator $\mathbb{I}[\cdot]$ equals 1 when its condition holds and 0 otherwise, and is applied coordinate-wise to vector conditions. Other notation is defined at first use.

Nearest neighbor search. Given X, q, and δ, K-nearest neighbor search returns the set

$$\mathcal{N}_K(q) \in \operatorname*{arg\,min}_{S \subseteq X,\, |S|=K} \sum_{x \in S} \delta(q, x). \tag{1}$$

Due to the curse of dimensionality, exact K-nearest neighbor search generally requires computing the distance from the query to every database vector, which becomes prohibitively expensive at scale. Approximate nearest neighbor (ANN) search therefore trades a small loss in result accuracy for substantially lower search cost. Let $\widehat{\mathcal{N}}_K(q)$ denote the K points returned by ANN. Search quality is measured by $\mathrm{Recall@}\, K = |\mathcal{N}_K(q) \cap \widehat{\mathcal{N}}_K(q)|/K$. A method provides a better recall–efficiency trade-off if it achieves higher recall in shorter time.

Binary codes. A binary code maps each vector $x \in \mathbb{R}^d$ to a compact string $b(x) \in \{0, 1\}^B$. The Hamming distance $H_B(x, y) = |\{j : b_j(x) \neq b_j(y)\}|$ counts the positions at which two codes differ. Both the storage and the comparison are extremely cheap: a code occupies B bits, compared with $32d$ bits for the original float vector.

Random-hyperplane binary codes. Let $A \in \mathbb{R}^{B \times d}$ have independent rows $a_j \sim \mathcal{N}(0, I_d)$. The standard B-bit code is $h(x) = \mathbb{I}[Ax \geq 0] \in \{0, 1\}^B$, with $h_j(x) = \mathbb{I}[a_j^\top x \geq 0]$. Thus nonnegative projections are encoded as 1 and negative projections as 0. For two nonzero fixed vectors x, y with angle $\theta(x, y) \in [0, \pi]$, a random hyperplane separates them with probability (Charikar, 2002)

$$\Pr[h_j(x) \neq h_j(y)] = \frac{\theta(x, y)}{\pi}. \tag{2}$$

Consequently, their unnormalized Hamming distance $H_B(x, y) = \sum_{j=1}^B \mathbb{I}[h_j(x) \neq h_j(y)]$ satisfies

$$H_B(x, y) \sim \mathrm{Binomial}\left(B, \frac{\theta(x, y)}{\pi}\right), \quad \widehat{\cos\theta}(x, y) = \cos\left(\pi \frac{H_B(x, y)}{B}\right). \tag{3}$$

Although H_B/B is unbiased for θ/π, the nonlinear cosine estimate is not strictly unbiased at finite B. Many data-dependent hashing schemes have been proposed since, but a systematic re-evaluation under fair search procedures found random-projection hashing to be among the strongest methods (Cai, 2021); we therefore adopt it as the default encoder throughout this paper. As a reference baseline, we combines the classical random-hyperplane angle estimate (Charikar, 2002) with the original vector norms in the squared L2 distance identity:

$$\widehat{\delta}_0^2(q,x) = \|q\|_2^2 + \|x\|_2^2 - 2\|q\|_2\|x\|_2 \cos\left(\pi \frac{H_B(q,x)}{B}\right). \tag{4}$$

SimHash with stored vector norms has also been used for inner-product estimation (Pham, 2020).

Candidate generation and exact reranking. A common ANN strategy uses lightweight computations to select $L \geq K$ candidates $\mathcal{C}_L(q)$, then ranks them by exact distances to the query using full-precision vectors and returns the closest K points.

3 Related Work

Binary hashing for nearest-neighbor search. We focus on hashing methods that encode high-dimensional vectors as binary codes and use Hamming distance to select candidates for exact reranking. These methods include data-independent constructions, such as random-hyperplane LSH (Charikar, 2002), and data-dependent methods that learn encodings from the database; we refer readers to Wang et al. (2018) for a broader overview. A representative learned method is iterative quantization (ITQ), which learns an orthogonal rotation of zero-centered embeddings to reduce their quantization error (Gong et al., 2013). Use Hamming distance of generated binary codes as an estimate of the (L2) distance between original vectors for sorting. Despite the development of numerous learned hashing methods, Cai (2021) found random-projection LSH to perform particularly well among the methods tested in a systematic re-evaluation. This evidence, together with its simplicity and explicit relationship between Hamming distance and angle, motivates our choice of random-hyperplane encoding. Our focus is on improving the resulting distance estimates through centering and residual norms, rather than learning a new projection.

Quantization-based methods. Quantization-based methods compress vectors and estimate distances from their compressed representations. Product quantization (PQ), a classical method widely supported by vector databases, divides vectors into subvectors and encodes each using a learned codebook, enabling efficient distance estimation through lookup tables (Jégou et al., 2011); For a broader overview of vector quantization methods such as Optimized Product Quantization (OPQ) (Ge et al., 2014) and Additive Quantization (AQ) (Babenko & Lempitsky, 2014), we refer readers to Matsui et al. (2018). More recent methods include RaBitQ, which provides randomized quantization with theoretical distance-estimation guarantees (Gao & Long, 2024), SAQ(Li et al., 2025)and TurboQuant, which uses random rotations and scalar quantization, with residual correction for inner-product estimation (Zandieh et al., 2026). Inspired by RaBitQ's residual formulation, we combine residual norms with Hamming-based angle estimates. Norm-aware transformations also have a long history in hashing for maximum inner product search, from asymmetric LSH constructions (Shrivastava & Li, 2014; Neyshabur & Srebro, 2015) to norm-partitioned schemes that handle long-tailed norms (Yan et al., 2018); our focus is on Euclidean search under directional concentration instead. Our goal is to understand and improve this simple binary-code approach, rather than to establish a new state of the art in vector quantization.

4 Why Direct Binary Angles Fail

When vectors concentrate around a shared direction, it can limit the ability of random-hyperplane codes to distinguish true neighbors from other points. We illustrate this limitation through a toy example and measurements on real datasets.

4.1 A toy failure case

Suppose the data live in the first quadrant of the plane and two random projections coincide as shown in Figure 1(a). Every point receives the identical 2-bit code $(1, 1)$. The Hamming distance is zero

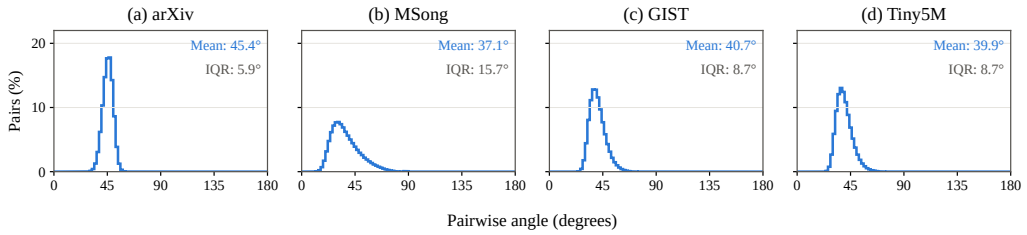

Figure 2: Original-vector angle distributions on (a) arXiv, (b) MSong, (c) GIST, and (d) Tiny5M. Annotations report the mean angle and interquartile-range (IQR) width.

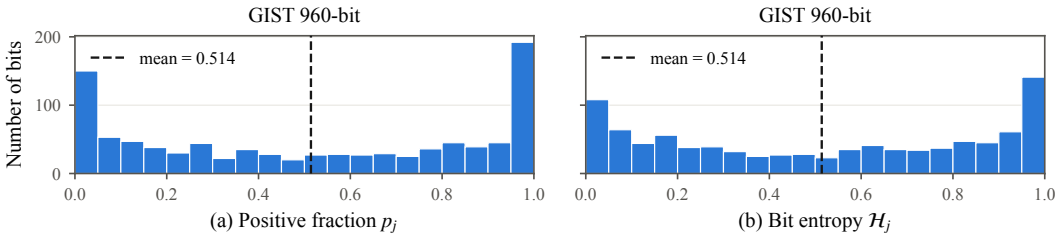

Figure 3: Positive-fraction and per-bit entropy distributions for random-projection codes on GIST.

for every pair: the code carries no angular information at all. Two hyperplanes nominally partition the plane into four angular cells, but when the data occupy only one cell, the effective resolution collapses, and near neighbors become indistinguishable from far-away points.

4.2 Angular concentration in real datasets

We examine angular concentration in four real-world datasets by uniformly sampling one million pairs of distinct, nonzero database vectors per dataset. Figure 2 shows their angles before centering, using the same pairs as the before-and-after comparison in Section 7.4. Detailed statistics appear in Appendix D, Table 3.

Two observations stand out. First, pairwise angles occupy relatively narrow bands: the middle half spans only $5.9°$ on arXiv and $8.7°$ on GIST. Tiny5M shows a similar spread, while MSong has a broader distribution. Second, the angles themselves are small: mean angles across the four datasets range from $37.1°$ to $45.4°$, well below $90°$. Thus, vectors are close in direction on average, rather than merely having similar pairwise angles. This shared directional bias can cause random hyperplanes to assign the same sign to most points, limiting the codes' ability to resolve angular differences. Section 4.3 examines the resulting bit imbalance.

4.3 Bit imbalance and low entropy

We examine codes generated by applying Gaussian random projections directly to the original vectors, with $B = d$. For each bit, the positive fraction p_j is the proportion of database points assigned a value of one; its entropy $\mathcal{H}_j$ ranges from zero for a constant bit to one for a balanced bit. Definitions and further discussion are provided in Appendix D.2.

Figure 3 shows clear bit imbalance on GIST: many positive fractions lie near zero or one, indicating that almost all points receive the same value on those bits. The entropy distribution confirms this imbalance: 461 of the 960 bits have $\mathcal{H}_j < 0.5$, including 172 with $\mathcal{H}_j < 0.1$, and mean per-bit entropy is only 0.5141. These nearly constant bits contribute little to pairwise Hamming differences, limiting the codes' ability to distinguish points. The arXiv, MSong, and Tiny5M results show similar trends (Appendix D.2).

4.4 Bit balance alone is insufficient for L2 ranking

The preceding section identifies bit imbalance and low entropy in direct random-projection codes. Does obtaining more balanced codes resolve the retrieval problem? We compare these codes with mean-centered PCA-ITQ codes (Gong et al., 2013) on GIST. With $B = 960$, the search encoders have

mean per-bit entropies of 0.5132 and 0.9980, respectively. Nevertheless, when both rank candidates by Hamming distance and use $L = 1000$, ITQ achieves only 0.60090 Recall@20, compared with 0.66465 for direct random projection. Both scan the full database, use the same 1,000 queries, and perform exact L2 reranking. Thus, substantially higher bit entropy does not necessarily translate into better L2 candidate ranking. Appendices F.5 and F.6 provide further experimental details and results, including a comparison with median-threshold codes that produce nearly balanced bits.

Bit entropy measures how evenly each bit divides the database, not how well the resulting scores preserve L2 distance. Moreover, even an exact angle does not determine L2 distance without vector lengths. Vectors with the same direction but different lengths receive identical random-hyperplane codes, yet their Euclidean distances to a query can be very different. Both angular representation and distance scoring therefore matter. These observations suggest that improving bit balance alone is insufficient; candidate scoring must also account for vector lengths when estimating L2 distances.

5 Norm-Aware Residual Hashing

5.1 Translation-aware distance estimation

To address the directional concentration and bit imbalance identified above, we shift every database vector and query by the same center $c \in \mathbb{R}^d$. In this translated space, the vectors $x - c$ and $q - c$ describe their directions relative to c instead of the original origin. Translation can therefore expose angular differences that were obscured by a shared component. However, Hamming distance between translated codes describes residual directions, not their lengths. Following Section 4.4, we retain both when estimating the original L2 distance.

The new angle cannot, however, be used directly in the original distance expansion

$$\delta^2(q, x) = \|q\|_2^2 + \|x\|_2^2 - 2\|q\|_2\|x\|_2 \cos \angle(q, x). \tag{5}$$

In general, $\theta_c = \angle(q - c, x - c)$ differs from $\angle(q, x)$, so replacing only the cosine term above by an estimate of $\cos\theta_c$ would not estimate the original distance. Instead, we translate the entire distance expansion. Because $q - x = (q - c) - (x - c)$, the following identity holds *for any* $c \in \mathbb{R}^d$:

$$\delta^2(q, x) = \|q - c\|_2^2 + \|x - c\|_2^2 - 2\|q - c\|_2\|x - c\|_2 \cos\theta_c. \tag{6}$$

Thus, estimating the translated angle together with the translated norms yields an estimate of the original squared Euclidean distance without changing the retrieval metric. This follows the norm-aware residual viewpoint used by RaBitQ (Gao & Long, 2024).

For encoding, define the residuals and their norms as $x_c = x - c,\ q_c = q - c,\ \rho_x = \|x_c\|_2,\ \rho_q = \|q_c\|_2$. We encode a database vector and a query using the same Gaussian projection $A \in \mathbb{R}^{B\times d}$: $h_c(x) = \mathbb{I}[Ax_c \geq 0],\ h_c(q) = \mathbb{I}[Aq_c \geq 0]$, both in $\{0, 1\}^B$.

Let $H_c(q, x)$ denote the Hamming distance between the two centered codes. Random-hyperplane hashing gives $H_c(q, x)/B \approx \theta_c/\pi$, leading to the squared-distance estimator

$$\widehat{\delta}_c^2(q, x) = \rho_q^2 + \rho_x^2 - 2\rho_q\rho_x \cos\left(\pi \frac{H_c(q, x)}{B}\right). \tag{7}$$

Thus the sketch changes the coordinate system used to estimate an angle, but does not change the target distance. For each vector x in the database, we store one B-bit code and one floating-point residual norm. The projection A is shared by the index, while the centers are stored as index metadata. At query time, a table of the $B+1$ possible cosine values reduces Equation 7 to XOR, popcount, table lookup, and a few scalar operations. We focus on L2 distance; Appendix E.3 discusses adaptations to cosine similarity and inner-product search.

5.2 Choosing the translation center

Equation 6 is valid for every choice of the point c. We use the database arithmetic mean as a simple, parameter-free translation center: $c_{\text{mean}} = \mu = N^{-1}\sum_{i=1}^N x_i$.

The motivation for using this center is that it removes the common offset among all database vectors, which can improve the informativeness of the resulting binary codes. For the j-th projection vector

a_j, $a_j^\top x = a_j^\top \mu + a_j^\top (x - \mu)$. The term $a_j^\top \mu$ is a shared offset across all database vectors. When its magnitude dominates the variation in $a_j^\top (x - \mu)$, the corresponding binary bit takes the same value for most points and provides little discrimination. Subtracting the mean removes this shared offset exactly, so the projection sign reflects variation among the data points more directly. This can produce more balanced and informative bits, although the mean does not explicitly optimize bit entropy or retrieval recall.

The mean also minimizes the average squared residual length, since

$$\frac{1}{N}\sum_i \|x_i - c\|^2 = \frac{1}{N}\sum_i \|x_i - \mu\|^2 + \|c - \mu\|^2.$$

Residual lengths also enter the estimation-error bound in Proposition 1. These properties motivate the mean as a simple default. We also compare alternative center choices in Appendix B.

5.3 Integration with inverted-file search

We integrate the centered estimator into an inverted-file (IVF) index. Rather than using one global center, we assign each posting list $\mathcal{P}_j$ its own list-local center $c_j = |\mathcal{P}_j|^{-1} \sum_{x \in \mathcal{P}_j} x$. Points in that list are encoded relative to c_j, and the index stores their residual norms. Each probed list therefore requires a different query code $h_{c_j}(q)$.

A naive implementation would compute the dense projection $A(q - c_j)$ independently for every probed list, costing $O(n_{\text{probe}} Bd)$. We avoid these repeated matrix–vector multiplications by exploiting projection linearity:

$$A(q - c_j) = Aq - Ac_j. \tag{8}$$

During index construction, we compute and store Ac_j once for each list. A query computes Aq only once and obtains each list-specific code by subtracting Ac_j and taking signs. This reduces projection and code-generation cost to $O(Bd + n_{\text{probe}} B)$. We then use Equation 7 to select a global Top-L candidate set from the probed lists and rerank those candidates using exact distances. Appendix E.2 gives the complete query and storage costs.

6 Analysis

We now analyze how closely our estimator approximates the true squared Euclidean distance. Proposition 1 bounds this error in terms of the code length and residual norms. Our analysis builds on random-hyperplane collision probabilities and the concentration of normalized Hamming distance around normalized angle (Charikar, 2002; Yi et al., 2015). Combining these results with the residual distance identity and the Hoeffding bound (Hoeffding, 1963).

Proposition 1 (Finite-bit distance error)**.** *For fixed nonzero residuals $q - c$ and $x - c$, let A contain B independent isotropic Gaussian rows. For any $\alpha \in (0, 1)$, with probability at least $1 - \alpha$,*

$$|\widehat{\delta}_c^2(q, x) - \delta^2(q, x)| \le 2\pi \rho_q \rho_x \sqrt{\frac{\log(2/\alpha)}{2B}}. \tag{9}$$

Appendix A provides a detailed proof of proposition 1. The bound shows that the estimation error can be made small by using sufficiently many bits. For fixed residual norms and confidence level, the error bound decreases as $1/\sqrt{B}$: quadrupling the code length halves the bound.

Since nearest-neighbor search depends on candidate ranking, we next ask when estimation errors can reverse the order of two candidates. Corollary 1 gives a sufficient condition for preserving their order.

Corollary 1 (Pairwise order preservation)**.** *For two candidates x^+ and x^- with $\delta^2(q, x^+) < \delta^2(q, x^-)$, assign failure probability $\alpha/2$ to each application of Proposition 1, giving bounds ϵ_+ and ϵ_-. If $\delta^2(q, x^-) - \delta^2(q, x^+) > \epsilon_+ + \epsilon_-$, then $\widehat{\delta}_c^2(q, x^+) < \widehat{\delta}_c^2(q, x^-)$ with probability at least $1 - \alpha$.*

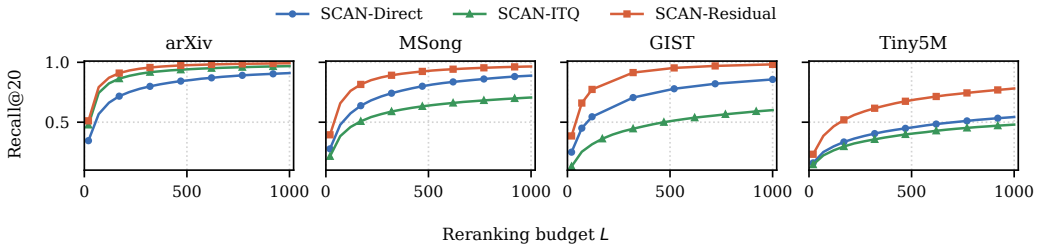

Figure 4: Recall@20 versus reranking budget L with $B = d$.

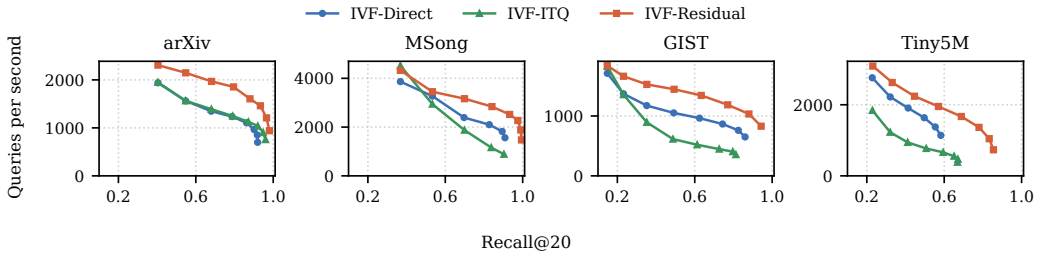

Figure 5: Recall–QPS trade-off for IVF-Direct, IVF-ITQ, and IVF-Residual with $B = d$, $K = 20$, and list assignments ($n_{\text{list}} = 4096$). Reranking budgets are listed in Appendix E.4, Table 5.

7 Experiments

7.1 Setup

Compared methods. *Residual* (ours) uses the distance estimator proposed in Section 5. It has two search variants: SCAN-Residual uses a global mean and scans all database codes, whereas IVF-Residual uses list-local means and scans only selected posting lists. *Direct* randomly projects the original vectors into binary codes and estimates squared L2 distances using Equation 4; its two search variants are SCAN-Direct and IVF-Direct. *ITQ* learns binary codes using PCA-ITQ (Gong et al., 2013) and ranks candidates by Hamming distance. Its SCAN-ITQ and IVF-ITQ variants share one global mean and learned transform. All variants retain L candidates, rerank them using full-precision distances, and return the closest K points.

We report experiments on four million-scale datasets: arXiv, MSong, GIST (Jégou et al., 2011), and Tiny5M. Their basic properties are summarized in Appendix D.1, Table 2. All experiments use the original, unnormalized vectors and squared Euclidean distance. This section contains four main experiments, with more experiments provided in the appendix. Experiment configurations like hardware are detailed in Appendix E.4.

7.2 Experiment 1: effect of centering in SCAN

We vary the reranking budget L to evaluate candidate quality. Figure 4 shows that SCAN-Residual achieves higher recall than SCAN-Direct and SCAN-ITQ across the tested budgets on all four datasets. For example, on GIST with $L = 1000$, SCAN-Residual achieves Recall@20 of 0.98260, compared with 0.85700 for SCAN-Direct and 0.60090 for SCAN-ITQ. These gains reflect better candidate ranking rather than a larger reranking budget, allowing SCAN-Residual to reach a target recall with fewer full-precision distance computations in reranking stage.

7.3 Experiment 2: search performance in IVF

We evaluate the recall–QPS trade-off in IVF search. Figure 5 shows that IVF-Residual achieves better trade-offs than IVF-Direct and IVF-ITQ across the four datasets. For example, on GIST with $n_{\text{probe}} = 128$, IVF-Residual achieves Recall@20 of 0.94325 at 829 QPS, compared with 0.86025 at 649 QPS for IVF-Direct and 0.81265 at 355 QPS for IVF-ITQ. IVF-Residual retains more true neighbors with a smaller budget, improving recall with fewer exact distance computations.

7.4 Experiment 3: Geometry and bit statistics before and after centering

We examine changes in angular distributions, positive fractions, and per-bit entropy before and after centering on all four datasets. Figure 6(a) shows that centering broadens the angular distributions and shifts their means toward $90°$. In panels (b) and (c), positive fractions move closer to 0.5 and per-bit entropies approach one. These observations show reduced directional concentration and bit imbalance; Appendix D.3 provides detailed statistics.

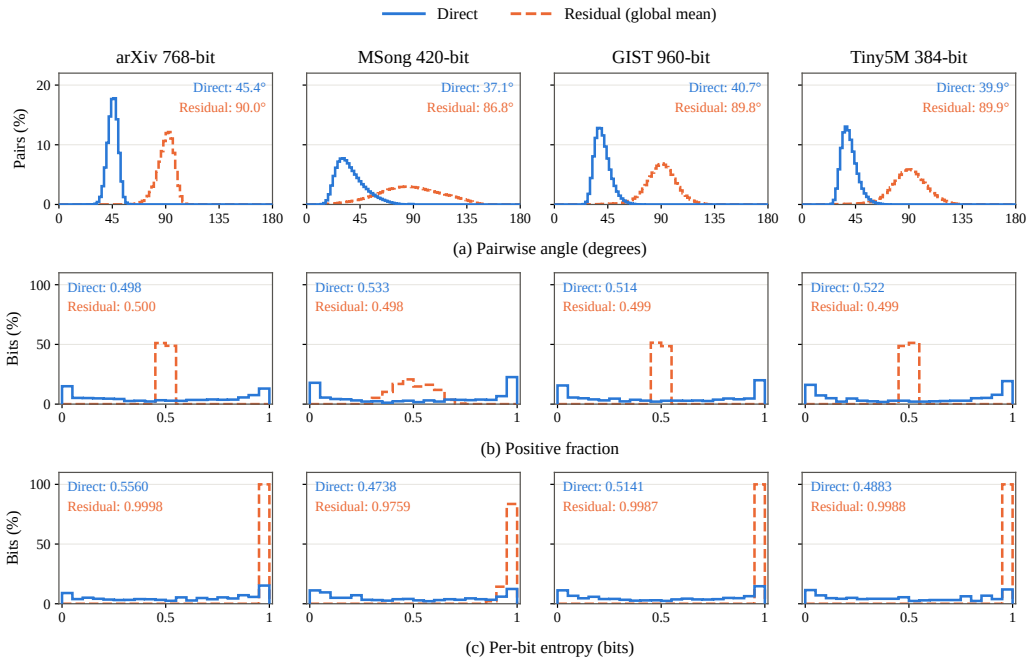

Figure 6: Direct versus Residual: angle, positive-fraction, and per-bit entropy distributions on four datasets ($B = d$). Annotations show the mean of each distribution.

7.5 Experiment 4: Hamming ranking versus norm-aware ranking

With fixed binary codes, we compare Hamming-distance ranking with our estimator (Equation 7), which also uses each vector's distance to the center. On GIST with $B = 960$, mean-centered random-projection codes have mean bit entropy 0.9987. At $L = 1000$, however, Hamming ranking achieves only 0.65335 Recall@20, whereas our norm-aware score achieves 0.98260. Both use identical database and query codes, the same candidate budget, and exact L2 reranking; only candidate scoring changes. With the same ITQ codes examined in Section 4.4, switching to residual-norm scoring likewise raises recall from 0.60090 to 0.92865 without changing the encoder. Thus, high bit entropy alone is insufficient for accurate L2 ranking, and retaining residual lengths is important for both tested encoders. Appendices F.6 and F.5 give the full comparisons.

8 Conclusion

We examined angular concentration and bit imbalance as limitations of binary hashing for ANN search. We proposed NARH, a mean-centered binary distance estimator that uses centered binary codes and distances to the center to estimate original squared Euclidean distances. We derived a finite-bit error bound and a sufficient condition for preserving pairwise rankings. We also extended the estimator to IVF with list-local centers, using precomputed center projections to reduce query computation. Experiments on four real-world datasets demonstrate improved search performance over random-projection and ITQ baselines. Our results highlight that improving binary-code retrieval requires attention not only to the codes themselves, but also to how they are used to rank candidates. We hope this work offers useful insights and motivates further research on classical hashing-based methods for efficient ANN search.

AI USE STATEMENT

Generative AI tools were used to assist with research ideation, source-code development, language editing, and preparation of an initial manuscript draft. The authors manually checked the mathematical derivations, executed and verified the reported experiments, inspected the cited literature, and take responsibility for all claims, code, and text in the submission. No generative AI tool was used to fabricate experimental observations or bibliographic records.

ETHICS STATEMENT

This work studies algorithms for nearest neighbor retrieval over existing vector datasets and does not involve human subjects or collection of personal data. As with other retrieval accelerators, downstream behavior inherits the properties and possible biases of the embedding model and indexed corpus.

REFERENCES

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=hSyW5go0v8.

Artem Babenko and Victor S. Lempitsky. Additive quantization for extreme vector compression. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pp. 931–938. IEEE Computer Society, 2014. doi: 10.1109/CVPR.2014.124. URL https://doi.org/10.1109/CVPR.2014.124.

Deng Cai. A revisit of hashing algorithms for approximate nearest neighbor search. *IEEE Transactions on Knowledge and Data Engineering*, 33(6):2337–2348, 2021. doi: 10.1109/TKDE.2019.2953897.

Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the 34th Annual ACM Symposium on Theory of Computing*, pp. 380–388, 2002.

Jianyang Gao and Cheng Long. High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations. *Proc. ACM Manag. Data*, 1(2), June 2023. doi: 10.1145/3589282. URL https://doi.org/10.1145/3589282.

Jianyang Gao and Cheng Long. RaBitQ: Quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data*, 2(3):1–27, 2024. doi: 10.1145/3654970.

Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. Practical and asymptotically optimal quantization of high-dimensional vectors in euclidean space for approximate nearest neighbor search. *Proc. ACM Manag. Data*, 3(3), June 2025. doi: 10.1145/3725413. URL https://doi.org/10.1145/3725413.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(4):744–755, 2014. doi: 10.1109/TPAMI.2013.240. URL https://doi.org/10.1109/TPAMI.2013.240.

Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(12):2916–2929, 2013.

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the 30th Annual ACM Symposium on Theory of Computing*, pp. 604–613, 1998.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

Leonardo Kuffo, Elena Krippner, and Peter Boncz. Pdx: A data layout for vector similarity search. *Proc. ACM Manag. Data*, 3(3), June 2025. doi: 10.1145/3725333. URL https://doi.org/10.1145/3725333.

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. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://papers.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Hui Li, Shiyuan Deng, Xiao Yan, Xiangyu Zhi, and James Cheng. Saq: Pushing the limits of vector quantization through code adjustment and dimension segmentation. *Proc. ACM Manag. Data*, 3 (6), December 2025. doi: 10.1145/3769824. URL https://doi.org/10.1145/3769824.

Qi Liu, Kai Zheng, Rui Huang, Wuchao Li, Kuo Cai, Yuan Chai, Yanan Niu, Yiqun Hui, Bing Han, Na Mou, Hongning Wang, Wentian Bao, Enyun Yu, Guorui Zhou, Han Li, Yang Song, Defu Lian, and Kun Gai. Recflow: An industrial full flow recommendation dataset. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=vVHc8bGRns.

Yusuke Matsui, Yusuke Uchida, Hervé Jégou, and Shin'ichi Satoh. A survey of product quantization. *ITE Transactions on Media Technology and Applications*, 6(1):2–10, 2018. doi: 10.3169/mta.6.2.

Behnam Neyshabur and Nathan Srebro. On symmetric and asymmetric LSHs for inner product search. In *International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1926–1934, 2015.

Mohammad Omama, Po han Li, and Sandeep P. Chinchali. Exploiting distribution constraints for scalable and efficient image retrieval. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=d0tlL0ZWlu.

Ninh Pham. Sublinear maximum inner product search using concomitants of extreme order statistics. *arXiv preprint arXiv:2012.11098*, 2020. URL https://arxiv.org/abs/2012.11098.

Dong Hoon Shin et al. Adaptive spatial hashing with dual-domain memristive hardware. *Nature Communications*, 17:5951, 2026. doi: 10.1038/s41467-026-72743-8. URL https://www.nature.com/articles/s41467-026-72743-8.

Anshumali Shrivastava and Ping Li. Asymmetric LSH (ALSH) for sublinear time maximum inner product search (MIPS). In *Advances in Neural Information Processing Systems*, volume 27, pp. 2321–2329, 2014.

Ziwen Song, Bin Wang, and Xiaochun Yang. Accelerating high-dimensional ann search via skipping redundant distance computations. *Proc. ACM Manag. Data*, 3(6), December 2025. doi: 10.1145/3769753. URL https://doi.org/10.1145/3769753.

The Faiss Developers. Faiss: A library for efficient similarity search and clustering of dense vectors. https://github.com/facebookresearch/faiss, 2026. GitHub repository. Accessed September 26, 2026.

Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018. doi: 10.1109/TPAMI.2017.2699960.

Xiao Yan, Jinfeng Li, Xinyan Dai, Hongzhi Chen, and James Cheng. Norm-ranging LSH for maximum inner product search. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Xinyang Yi, Constantine Caramanis, and Eric Price. Binary embedding: Fundamental limits and fast algorithm. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 2162–2170, 2015.

Yiwen Yuan, Zecheng Zhang, Xinwei He, Akihiro Nitta, Weihua Hu, Manan Shah, Blaž Stojanovič, Shenyang Huang, Jan Eric Lenssen, Jure Leskovec, and Matthias Fey. ContextGNN: Beyond two-tower recommendation systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=nzOD1we8Z4.

Amir Zandieh, Majid Daliri, Majid Hadian, and Vahab Mirrokni. Turboquant: Online vector quantization with near-optimal distortion rate. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=tO3ASKZlok.

A PROOFS

A.1 PROOF OF PROPOSITION 1

Fix q, x, c independently of the projection matrix A, with nonzero residuals $q - c$ and $x - c$. Let θ_c be their angle and $p = \theta_c/\pi$. For the j-th row a_j of A, define the bit-disagreement indicator

$$Z_j = \mathbb{I}[(h_c(q))_j \neq (h_c(x))_j], \qquad H_c = \sum_{j=1}^{B} Z_j.$$

By the random-hyperplane identity in Equation 2, $\Pr(Z_j = 1) = p$. Since the rows of A are independent, the Z_j are independent Bernoulli variables with mean p.

Thus, H_c/B is the average of B independent variables in $[0, 1]$. Hoeffding's inequality gives, for any $t > 0$,

$$\Pr\left[\left|\frac{H_c}{B} - p\right| \geq t\right] \leq 2\exp(-2Bt^2). \tag{10}$$

Choose $t = \sqrt{\log(2/\alpha)/(2B)}$, so that $2\exp(-2Bt^2) = \alpha$. Consequently, the event $\mathcal{E} = \{|H_c/B - p| \leq t\}$ has probability at least $1 - \alpha$.

To transfer this bound to the cosine estimate, let $f(z) = \cos(\pi z)$. Its derivative satisfies $|f'(z)| = \pi|\sin(\pi z)| \leq \pi$ on $[0, 1]$. The mean value theorem therefore gives $|f(u) - f(v)| \leq \pi|u - v|$ for all $u, v \in [0, 1]$. On $\mathcal{E}$, using $\cos(\pi p) = \cos\theta_c$, we obtain

$$\left|\cos\left(\pi\frac{H_c}{B}\right) - \cos(\theta_c)\right| \leq \pi\left|\frac{H_c}{B} - p\right| \leq \pi t. \tag{11}$$

Finally, subtract the exact residual-distance identity in Equation 6 from the estimator in Equation 7. Both use the same residual norms, so the terms $\rho_q^2 + \rho_x^2$ cancel. On the same event $\mathcal{E}$,

$$\begin{aligned}\left|\widehat{\delta}_c^2(q, x) - \delta^2(q, x)\right| &= 2\rho_q\rho_x\left|\cos\left(\pi\frac{H_c}{B}\right) - \cos\theta_c\right| \\ &\leq 2\pi\rho_q\rho_x t = 2\pi\rho_q\rho_x\sqrt{\frac{\log(2/\alpha)}{2B}}.\end{aligned}$$

Since all subsequent bounds hold deterministically on $\mathcal{E}$, the final distance bound retains probability at least $1 - \alpha$. This proves Equation 9. □

A.2 PROOF OF COROLLARY 1

Apply Proposition 1 to x^+ and x^- with failure probability $\alpha/2$ each. The error bounds for x^+ and x^-, respectively, are

$$\epsilon_+ = 2\pi\rho_q\rho_{x^+}\sqrt{\frac{\log(4/\alpha)}{2B}}, \qquad \epsilon_- = 2\pi\rho_q\rho_{x^-}\sqrt{\frac{\log(4/\alpha)}{2B}}. \tag{12}$$

Let $\mathcal{E}_+$ and $\mathcal{E}_-$ denote the events that the absolute estimation errors for x^+ and x^- do not exceed ϵ_+ and ϵ_-, respectively. Each event fails with probability at most $\alpha/2$. By the union bound, $\Pr(\mathcal{E}_+ \cap \mathcal{E}_-) \geq 1 - \alpha/2 - \alpha/2 = 1 - \alpha$. This step does not require the two errors to be independent, even though the candidates share the same random projection matrix.

On this joint event, the nearer candidate can be overestimated by at most ϵ_+, while the farther candidate can be underestimated by at most ϵ_-. Consequently,

$$\begin{aligned}\widehat{\delta}_c^2(q, x^-) - \widehat{\delta}_c^2(q, x^+) &\geq \big(\delta^2(q, x^-) - \epsilon_-\big) - \big(\delta^2(q, x^+) + \epsilon_+\big) \\ &= \delta^2(q, x^-) - \delta^2(q, x^+) - \epsilon_- - \epsilon_+ > 0.\end{aligned}$$

The last inequality follows from the assumed distance gap $\delta^2(q, x^-) - \delta^2(q, x^+) > \epsilon_+ + \epsilon_-$. Thus the estimated distances preserve the true pairwise order with probability at least $1 - \alpha$. □

B CENTER-CHOICE ABLATIONS

Besides the arithmetic mean in Section 5.2, we consider two distinct median centers and a randomly selected database point. Although the mean minimizes average squared residual energy, this does not guarantee the lowest finite-bit Top-K ranking error. To see one limitation, define $u_i(c) = (x_i - c)/\|x_i - c\|_2$ and $\rho_i(c) = \|x_i - c\|_2$. The mean condition $\sum_i (x_i - \mu) = 0$ can be written as

$$\sum_i \rho_i(\mu) u_i(\mu) = 0, \qquad \sum_i u_i(\mu) \neq 0 \quad \text{in general.} \tag{13}$$

Thus, the mean balances residual directions after weighting them by residual length; points farther from the center have greater influence. This distinction motivates testing centers with different geometric objectives, without assuming that any such objective directly optimizes retrieval recall.

The coordinate-wise median minimizes the sum of coordinate-wise absolute deviations:

$$(c_{\text{coord}})_j \in \arg\min_{t \in \mathbb{R}} \sum_{i=1}^{N} |(x_i)_j - t|, \qquad j = 1, \ldots, d. \tag{14}$$

For an even number of points, we average the two middle values in each coordinate. In contrast, the geometric median minimizes Euclidean residual lengths:

$$c_{\text{geo}} \in \arg\min_{c \in \mathbb{R}^d} \sum_{i=1}^{N} \|x_i - c\|_2. \tag{15}$$

When it does not coincide with a data point, its first-order condition is

$$\sum_{i=1}^{N} \frac{x_i - c_{\text{geo}}}{\|x_i - c_{\text{geo}}\|_2} = 0. \tag{16}$$

This balances normalized residual directions rather than residual vectors weighted by length. This optimality condition does not guarantee uniform binary codes or improved Top-K ranking. We compute geometric medians offline with mean-initialized Weiszfeld iterations.

B.1 DIFFERENT CENTERS IN SCAN

We compare four global center choices on arXiv, MSong, GIST, and Tiny5M: the arithmetic mean mentioned in Section 5.2, coordinate-wise median, geometric median, and a uniformly sampled database point. We use exhaustive binary-code scan followed by exact reranking. The first three centers are computed from the full dataset; the sampled point is selected for full dataset and uses it as the shared center for every database vector and query. We repeat this selection with five center seeds (1–5), independently of the queries and projection, and report mean recall with a one-standard-deviation band across the five runs. The code lengths for each dataset are 768, 420, 960, and 384 bits, respectively. Each variant uses its own residual lengths in Equation 7 to estimate distances, and reranks the estimated Top-L using original-vector distances. The geometric-median solver uses at most 100 Weiszfeld iterations and stops when the center displacement divided by $\max(1, \|c\|_2)$ is at most 10^{-6}.

Figure 7 shows that the effect of center choice depends on the dataset. We can observe that the three center selection methods perform close to each other. For example, at $L = 1000$ on arXiv, the recalls for the mean, coordinate-wise median, and geometric median are 0.99215, 0.99155, and 0.99170,

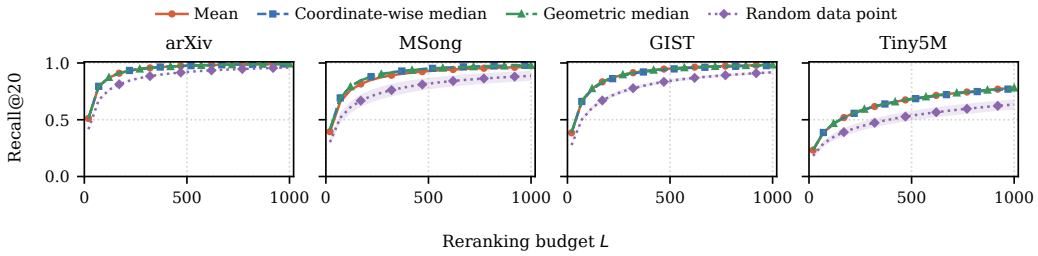

Figure 7: Global center choice in SCAN: Recall@20 versus reranking budget L for the arithmetic mean, coordinate-wise median, geometric median, and a random point in the dataset. The random-data curve and shaded band show the mean and $\pm$ one sample standard deviation over five center seeds. All variants share the same Gaussian projection within each dataset, with $B = d$, and use exact L2 reranking.

respectively, with very small differences among them. The random-data baseline is consistently weaker: all five sampled centers yield lower recall than the mean at every tested budget on all four datasets. At $L = 1000$, its mean recalls are 0.95968, 0.88690, 0.91772, and 0.63442 on arXiv, MSong, GIST, and Tiny5M, respectively; the sample standard deviations are 0.00093, 0.04372, 0.01025, and 0.04537. Thus, in these runs, translating to an arbitrary database point does not reproduce the retrieval quality of mean centering. This reflects the effectiveness of our center choice method.

B.2 Code length at a fixed SCAN reranking budget

We perform the corresponding code-length comparison under exhaustive SCAN, using a single global mean, coordinate-wise median, geometric median, or randomly selected database point as the center. For the random-data baseline, we use the same five center seeds as in Appendix B.1 and keep each sampled center fixed across code lengths. We fix $L = 300$ on arXiv, MSong, and GIST and $L = 500$ on Tiny5M, and vary $B \in \{d/4, d/2, d, 2d\}$.

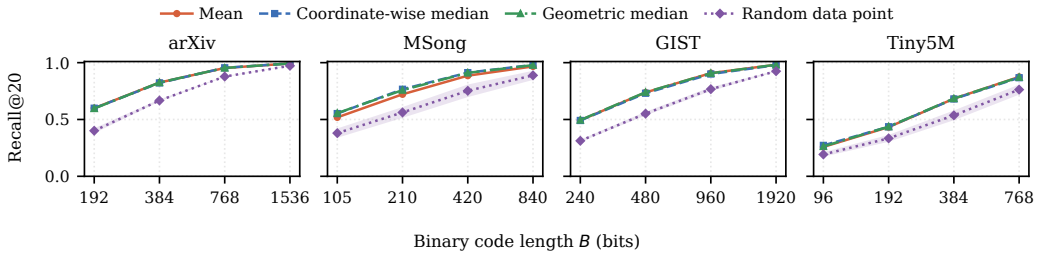

Figure 8: SCAN Recall@20 versus binary code length for four global centers, with $L = 300$ on arXiv, MSong, and GIST and $L = 500$ on Tiny5M. The random-data curve and shaded band show the mean and one standard deviation over five sampled centers. The horizontal axes use a logarithmic scale.

From the results presented in Figure 8, we can observe that the first three centers have a relatively small effect on recall, while the random-data baseline performs noticeably worse. For example, in GIST with 960 bits, the recall for the mean, coordinate-wise median, geometric median and random-data baseline are 0.90770, 0.90000, 0.90685 and 0.76676. Increasing the binary code length significantly improves recall for all centering schemes, because longer binary codes provide more precise distance estimation, thereby improving retrieval accuracy. Similar trends can be observed in the IVF experiments in the next section.

B.3 Code length at a fixed IVF reranking budget of different centers

We compared three different centers—the K-means centroid, coordinate-wise median, and geometric median—across code lengths at a fixed IVF reranking budget on all four datasets. We fix $L = 300$ on arXiv, MSong, and GIST and $L = 500$ on Tiny5M, with $n_{\text{probe}} = 128$ and $K = 20$, and vary $B \in \{d/4, d/2, d, 2d\}$. All variants use the same 4,096 post lists, and Gaussian projection. The mean baseline uses the existing K-means centroids; the two medians are computed from the points assigned to each list. Coarse list selection is identical across centers and code lengths. Recall is measured over all 1,000 queries after reranking with exact distance of the Top-L returned candidates.

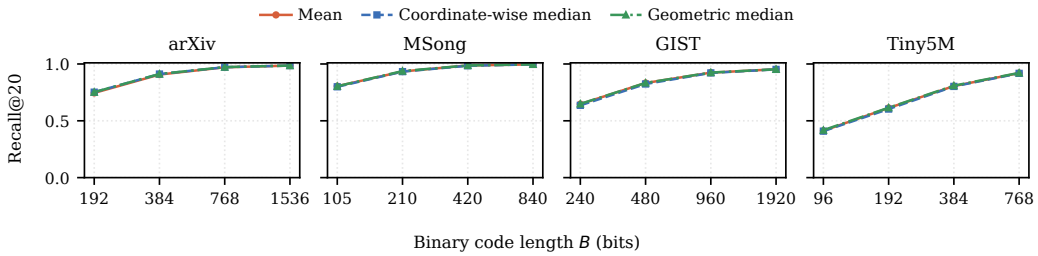

Figure 9: IVF Recall@20 versus binary code length for three list-local centers. The reranking budget is fixed at $L = 300$ on arXiv, MSong, and GIST and $L = 500$ on Tiny5M; all curves use $n_{\text{probe}} = 128$. The horizontal axes use a logarithmic scale.

Figure 9 shows that at different binary code lengths, the choice of center has a relatively small effect on recall. For example, on arXiv with 768 bits, the recalls for the mean, coordinate-wise median, and geometric median are 0.97090, 0.97170, and 0.97250, respectively. Meanwhile, increasing the code length significantly improves recall for all three center methods. This reflects that mean centering is a suitable choice in practice.

C Comparison with RaBitQ and PQ

In this section, we compare IVF-Residual with $B \in \{d, 2d\}$ against native IVF-RaBitQ 1-bit (Gao & Long, 2024) and 2-bit (Gao et al., 2025). Search is single-threaded over all 1,000 queries. QPS includes query processing and reranking. All methods use the same 4,096 post lists, $K = 20$, and exact reranking. Residual uses list-local mean centers as described in Section 5.3. Table 1 lists the fixed L for each dataset and bit setting. We use grid search to find L which can achieve the high recall and QPS trade-off.

All three PQ baselines evaluated below, IVF-PQ8, IVF-PQ4 FastScan, and IVF-PQ4 (float ADC), use the CPU implementations in the Faiss library (The Faiss Developers, 2026). We first evaluate IVF-PQ (Jégou et al., 2011) with exact reranking on the original vectors. Each four-dimensional subvector uses an 8-bit codebook, giving $d/4$ subquantizers and $2d$ code bits per vector. PQ uses the same coarse centroids and list assignments. We select the fastest measured PQ configuration reaching 98% Recall@20 over $L \in \{100, 300, 500, 1000, 2000\}$ and ten probe counts from 1 to 512, then fix its L and plot the full probe sweep. This selects $L = 100$ on all four datasets; the existing Residual and RaBitQ curves remain unchanged. PQ QPS is measured over three full query passes after warmup and includes exact reranking; these measurements were collected separately from the existing curves.

We also add IVF-PQ4 FastScan with exact reranking, using four-dimensional subvectors and 4-bit codebooks (d code bits per vector before padding). It uses the same IVF partitions and timing protocol. Applying the same 98% recall selection rule gives $L = 300$, 500, and 2000 for arXiv, GIST, and Tiny5M, respectively. MSong does not reach this target: we show $L = 2000$, which attains the highest observed recall, 79.175%, at 16 probes. For clarity, the plot omits its five points beyond 16 probes, where recall decreases. A diagnostic on 100 MSong queries with identical PQ4 codes at 128 probes and $L = 500$ obtains 98.5% candidate recall with floating-point ADC, versus 59.5% with the

tested default FastScan path. The discrepancy therefore concerns the FastScan distance path rather than PQ4 encoding alone; its precise cause remains unresolved.

We also evaluate IVF-PQ4 (float ADC), which uses the same PQ4 codes as FastScan but computes asymmetric distances with floating-point lookup tables, followed by exact reranking. The same 98% selection rule gives $L = 300$, 500, 500, and 2000 for arXiv, MSong, GIST, and Tiny5M; we plot all ten probe counts at each fixed budget. On all 1,000 MSong queries, it reaches 98.190% recall at 302.34 QPS with 64 probes and $L = 500$, showing that PQ4 encoding itself does not impose the observed FastScan recall limitation. At the 98% target, however, floating-point ADC is slower than FastScan on the other three datasets. The timing protocol is unchanged, but measurements come from separate runs.

As an unquantized reference, we add IVF-Flat, which computes exact L2 distances to every original vector in the selected lists and returns Top-K directly, without a reranking budget L. It uses the same 4096 list assignments, all 1,000 queries, $K = 20$, and single-threaded search. We plot all ten probe counts from 1 to 512; QPS includes list selection, distance computation, and Top-K maintenance, averaged over three passes after warmup. These timings were collected separately from the other curves.

Table 1: Fixed reranking budgets L in Figure 10.

	IVF-Residual		IVF-RaBitQ		IVF-PQ8	IVF-PQ4	
Dataset	$B = d$	$B = 2d$	1-bit	2-bit	float ADC	FastScan	float ADC
arXiv	200	100	100	50	100	300	300
MSong	200	100	100	50	100	2000	500
GIST	300	200	300	100	100	500	500
Tiny5M	400	500	500	100	100	2000	2000

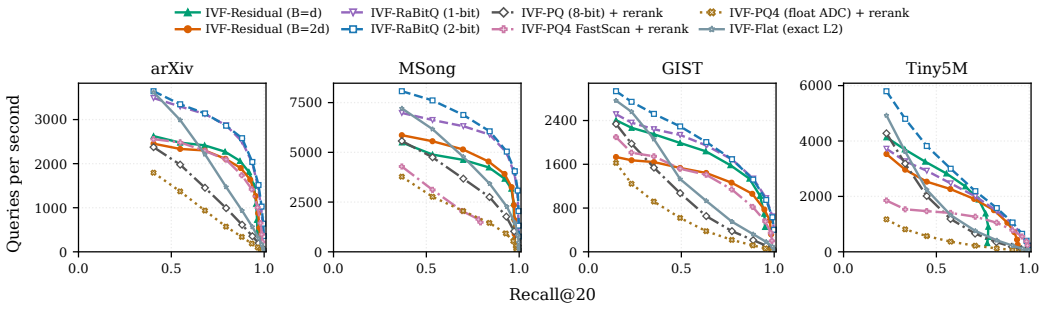

Figure 10: Extended Recall–QPS comparison of IVF-Residual, IVF-RaBitQ, IVF-PQ (8-bit), IVF-PQ4 FastScan, IVF-PQ4 (float ADC), and IVF-Flat. Quantized methods use exact reranking with fixed L from Table 1; IVF-Flat uses original-vector distances without quantization or reranking.

We can see that at low recall levels, IVF-RaBitQ 1-bit and 2-bit achieve notably higher QPS than IVF-Residual. At high recall levels, our method IVF-Residual becomes more competitive, providing fast query speed while maintaining high recall, with query performance close to that of RaBitQ. On the arXiv dataset, when we further tune the reranking budget to push recall to a higher level, IVF-Residual with $B = d$ achieves 99.245% Recall@20 at 562 QPS, compared with 99.250% at 602 QPS for IVF-RaBitQ 1-bit, retaining approximately 93% of its QPS at nearly identical recall. The goal of this paper is not to propose a method that outperforms RaBitQ, but to show that the classical approach of projecting data into binary code space followed by Hamming distance computation can still achieve efficient approximate nearest neighbor search with appropriate residual encoding, remaining competitive.

D ADDITIONAL DATASET DIAGNOSTICS

D.1 EXPERIMENTAL DATASETS

Table 2 summarizes the four datasets used in our experiments. Dataset sources are described in Kuffo et al. (2025); Gao & Long (2023). The binary-code length B equals the original vector dimension in the main experiments.

Table 2: Datasets used in our experiments.

Dataset	Modality	Base vectors	d	Queries	Bits B
arXiv	text	2,253,000	768	1,000	768
MSong	audio	994,185	420	1,000	420
GIST	image	1,000,000	960	1,000	960
Tiny5M	image	5,000,000	384	1,000	384

D.2 ANGULAR CONCENTRATION AND BIT IMBALANCE

Definitions and interpretation. For projection a_j, let $y_{ij} = a_j^\top x_i$ and $h_j(x_i) = \mathbb{I}[y_{ij} \geq 0]$. The empirical positive fraction is $p_j = N^{-1} \sum_{i=1}^{N} h_j(x_i)$, and the corresponding binary entropy is

$$\mathcal{H}_j = -p_j \log_2 p_j - (1 - p_j) \log_2 (1 - p_j), \tag{17}$$

with the convention $0 \log 0 = 0$. This is the marginal entropy of the observed bit: $\mathcal{H}_j = 0$ for a constant bit and $\mathcal{H}_j = 1$ for a perfectly balanced bit. The mean per-bit entropy is $\overline{\mathcal{H}} = B^{-1} \sum_{j=1}^{B} \mathcal{H}_j$. A positive fraction near zero or one means that almost all points receive the same sign. Such a bit occupies one code position but rarely contributes to the Hamming distance between database points. We report the full distributions of p_j and $\mathcal{H}_j$: averaging positive fractions across bits biased toward opposite signs can give a value near 0.5, even when many individual bits are nearly constant. These marginal statistics describe bit balance, not the accuracy of L2 candidate ranking.

Additional dataset results. Figure 11 gives the arXiv counterpart to the GIST bit-saturation results in Figure 3. Many bits have positive fractions close to 0 or 1, meaning that they assign the same binary value to almost all data points. Of the 768 bits, 330 have $\mathcal{H}_j < 0.5$, including 99 with $\mathcal{H}_j < 0.1$, and the mean per-bit entropy is 0.5560. Thus, although saturation is less extreme than on GIST, arXiv exhibits the same qualitative loss of useful Hamming-space resolution.

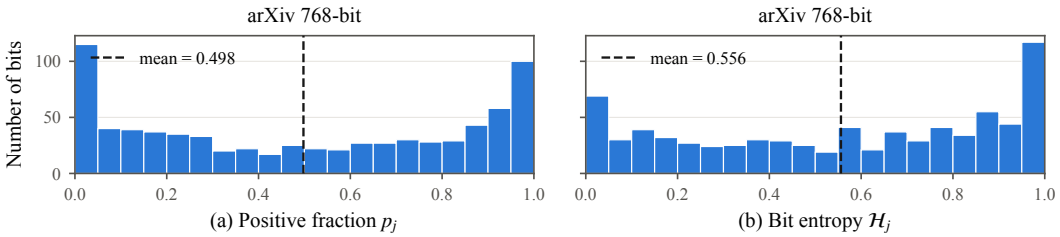

Figure 11: Positive-fraction and per-bit entropy distributions for 768-bit direct random-projection codes on arXiv.

Figures 12 and 13 show the same diagnostics for MSong and Tiny5M with $B = d$. On MSong, 230 of 420 bits have $\mathcal{H}_j < 0.5$, including 87 below 0.1, and the mean entropy is 0.4738. On Tiny5M, the corresponding counts are 208 and 71 out of 384 bits, with mean entropy 0.4883. Both datasets exhibit imbalanced positive fractions and many low-entropy bits. For each dataset, we compute these statistics over the all valid data points. Table 3 provides the detailed angular statistics underlying the distributions in Figure 2. All four datasets have small mean pairwise angles.

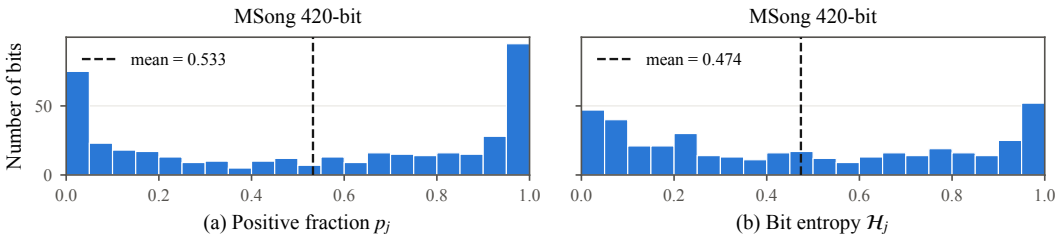

Figure 12: Positive-fraction and per-bit entropy distributions for 420-bit direct random-projection codes on MSong.

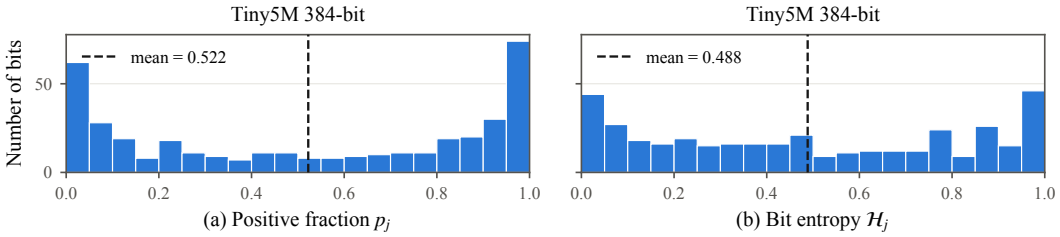

Figure 13: Positive-fraction and per-bit entropy distributions for 384-bit direct random-projection codes on Tiny5M.

D.3 Paired Direct and Residual diagnostics on four datasets

Table 4 supplements Section 7.4 and Figure 6. Each global mean of dataset is accumulated in float64 over all finite raw base vectors, including zero vectors; inputs are not normalized before centering. For directional and bit comparisons, both methods use the same rows with nonzero original and residual norms. This leaves 2,253,000 arXiv, 994,185 MSong, 999,990 GIST, and 4,999,556 Tiny5M rows. The excluded rows are 10 original zero vectors in GIST and 444 in Tiny5M; no nonfinite rows or zero residuals were found.

For each dataset we sample 1,000,000 ordered pairs uniformly from these rows, excluding self-pairs, with replacement across pairs, using seed 54321. The identical pairs are measured before and after translation. Reported mean angles average the individual arccosine values, rather than applying arccosine to a mean cosine. The original-vector angles are also used in Figure 2 and Table 3. In particular, these pairs are neither query neighbors nor a measurement of Top-K margins.

Bit statistics use a full scan, not the pair sample. We generate one float64 Gaussian matrix per dataset using NumPy's default generator, and use that identical matrix for both methods. The sign convention is nonnegative projection $\mapsto 1$. We reuse $A(x-\mu) = Ax - A\mu$ and cross-check this identity numerically against direct residual projection, as well as the translation-invariant squared-L2 identity. This is an independent diagnostic implementation, not a claim of bit-for-bit agreement with the search implementation's random-number generator. The reported bit distributions describe one projection realization per dataset, without multi-seed error bars. The direct-code diagnostics in Section 4.3 and this paired study both use $B = d$ and the same saved per-bit statistics.

E Implementation Details

The mean is accumulated in double precision and stored in float32. Each code uses $\lceil B/64 \rceil$ 64-bit words; unused final bits are masked. A data record stores one float32 residual norm. During an exhaustive query, a max-heap keeps the L smallest estimated distances. A second max-heap keeps the K smallest exact squared L2 distances among those candidates. The search timer starts before query centering and encoding and stops after final Top-K materialization. Code construction and ground-truth comparison are excluded.

Table 3: Angle statistics before centering, computed from one million randomly sampled pairs of distinct database vectors per dataset. Mean angles average individual pair angles; bands are their empirical percentiles. See Appendix D.3 for the shared sampling protocol.

Dataset	Mean cos-dist	Mean angle	IQR band	p5–p95 band
arXiv	0.2998	45.4°	$[42.5^\circ, 48.4^\circ]$	$[38.0^\circ, 52.2^\circ]$
MSong	0.2192	37.1°	$[28.3^\circ, 44.0^\circ]$	$[21.5^\circ, 60.3^\circ]$
GIST	0.2465	40.7°	$[35.9^\circ, 44.5^\circ]$	$[31.4^\circ, 53.1^\circ]$
Tiny5M	0.2378	39.9°	$[35.1^\circ, 43.7^\circ]$	$[30.6^\circ, 52.4^\circ]$

Table 4: Paired centering diagnostics with $B = d$. Angle means and IQR widths are in degrees. $\bar{p}$ is the mean positive fraction, $\overline{\mathcal{H}}$ the mean per-bit entropy, and $\Delta_p = B^{-1} \sum_j |p_j - 0.5|$ the mean absolute imbalance. The last column counts bits with entropy below 0.1.

Dataset	Method	Mean angle	IQR	$\bar{p}$	$\overline{\mathcal{H}}$	Δ_p	$\mathcal{H}_j < 0.1$
arXiv	Direct	45.40	5.88	0.4979	0.5560	0.3220	99/768
arXiv	Residual	89.98	9.16	0.5000	0.9998	0.0063	0/768
MSong	Direct	37.14	15.74	0.5326	0.4738	0.3530	87/420
MSong	Residual	86.83	36.29	0.4980	0.9759	0.0768	0/420
GIST	Direct	40.67	8.68	0.5142	0.5141	0.3360	172/960
GIST	Residual	89.83	15.94	0.4994	0.9987	0.0195	0/960
Tiny5M	Direct	39.89	8.67	0.5224	0.4883	0.3484	71/384
Tiny5M	Residual	89.87	18.54	0.4993	0.9988	0.0183	0/384

E.1 Search pseudocode

Algorithms 1 and 2 summarize SCAN-Residual and IVF-Residual. The index stores original vectors, centered codes $h_c(x)$, and residual norms $\rho_x = \|x - c\|_2$; IVF additionally stores each projected center Ac_j. Both algorithms use the same distance estimate

$$\mathrm{Estimate}(b, r, b_x, r_x) = r^2 + r_x^2 - 2rr_x\, T[\mathrm{popcount}(b \oplus b_x)], \qquad T[t] = \cos(\pi t/B),$$

where T is precomputed. KeepL maintains a bounded max-heap of the L smallest estimated distances, and $\mathrm{ExactTopK}(q, C, K)$ evaluates original-vector L2 distances for all candidates in C and returns the nearest K. IVF maintains one shared heap across all probed lists, not L candidates per list.

Algorithm 1 SCAN-Residual search

Require: Query q, database X, global center c, projection A, $L \geq K$
$b \leftarrow \mathbb{I}[A(q - c) \geq 0]$; $r \leftarrow \|q - c\|_2$; $C \leftarrow \varnothing$
for each $x \in X$ **do**
  $s \leftarrow \mathrm{Estimate}(b, r, h_c(x), \rho_x)$
  $C \leftarrow \mathrm{KeepL}(C, (x, s), L)$
end for
return $\mathrm{ExactTopK}(q, C, K)$

Algorithm 2 IVF-Residual search with a shared query projection

Require: Query q, IVF index $\{\mathcal{P}_j, c_j, Ac_j\}$, projection A, n_{probe}, $L \geq K$
$J \leftarrow$ indices of the n_{probe} nearest coarse centroids to q
$z \leftarrow Aq$; $C \leftarrow \varnothing$ ▷ one dense query projection
for each $j \in J$ **do**
  $b \leftarrow \mathbb{I}[z - Ac_j \geq 0]$; $r \leftarrow \|q - c_j\|_2$
  for each $x \in \mathcal{P}_j$ **do**
    $C \leftarrow \mathrm{KeepL}(C, (x, \mathrm{Estimate}(b, r, h_{c_j}(x), \rho_x)), L)$
  end for
end for
return $\mathrm{ExactTopK}(q, C, K)$

E.2 IVF QUERY COMPLEXITY AND STORAGE

Let S be the total number of vectors in the n_{probe} selected posting lists. After the coarse centroids select these lists, the query computes Aq once in $O(Bd)$ time. For each selected list j, it subtracts the stored Ac_j and takes signs in $O(B)$ time to obtain $h_{c_j}(q)$. Computing the corresponding query residual norm $\|q - c_j\|_2$ costs $O(d)$ per list; when c_j is also the coarse centroid, this distance can be reused from list selection.

Scanning the selected binary codes costs $O(S\lceil B/64\rceil)$ word operations. Maintaining the estimated Top-L set costs $O(S \log L)$ in the worst case, and exact reranking with Top-K maintenance costs $O(L(d + \log K))$. Excluding coarse-list selection, the complete query complexity is therefore

$$O\left(Bd + n_{\text{probe}}(B + d) + S\left\lceil \frac{B}{64} \right\rceil + S \log L + L(d + \log K)\right). \tag{18}$$

The central optimization is that only the first term is a dense projection; without the reuse in Equation 8, query projection alone would cost $O(n_{\text{probe}}Bd)$.

The additional per-vector storage is one B-bit code and one residual norm. Each posting list stores its center c_j and projected center Ac_j, requiring $d + B$ floating-point values per list. If the translation center is the IVF coarse centroid, its d values are already part of the coarse quantizer and need not be duplicated.

E.3 COSINE SIMILARITY AND INNER-PRODUCT SEARCH

Although our experiments use L2 distance, the residual estimator can also be adapted to cosine similarity and maximum inner-product search. The two cases require different treatment of vector norms.

Cosine similarity. For nonzero vectors, first normalize the database and query as $\bar{x} = x/\|x\|_2$ and $\bar{q} = q/\|q\|_2$. Their cosine similarity $s_{\cos}(q, x) = \bar{q}^\top \bar{x}$ satisfies

$$\|\bar{q} - \bar{x}\|_2^2 = 2\big(1 - s_{\cos}(q, x)\big). \tag{19}$$

Thus, maximizing cosine similarity is equivalent to minimizing squared L2 distance in the normalized space. Compute the global mean, or the IVF partitions and list-local means, from these normalized database vectors. Then encode $\bar{x} - c$ and $\bar{q} - c$ and apply Equation 7 to estimate $\|\bar{q} - \bar{x}\|_2^2$. Retain the L candidates with the smallest estimates and rerank them using exact cosine similarities, equivalently exact L2 distances between the full-precision normalized vectors. Normalization must precede center computation: the angle between centered residuals is not the original cosine similarity. Zero vectors require an explicit application-specific convention, since their cosine similarity is undefined.

Inner-product similarity. Converting inner-product search into distance-based search through norm-dependent transformations has a long history in hashing, including asymmetric LSH for maximum inner product search (Shrivastava & Li, 2014), the parameter-free symmetric scheme of Neyshabur & Srebro (2015), and norm-partitioned variants robust to long-tailed norms (Yan et al., 2018). Our estimator shares this spirit but requires no separate hash tables, vector norm bounds, or normalized queries, operating instead within a single shared sketch. For $s_{\text{IP}}(q, x) = q^\top x$, database norms affect the ranking, so normalizing the database would generally change the search objective. Instead, retain the original vectors and use the identity $q^\top x = (\|q\|_2^2 + \|x\|_2^2 - \delta^2(q, x))/2$ to define

$$\widehat{s}_{\text{IP},c}(q, x) = \frac{\|q\|_2^2 + \|x\|_2^2 - \widehat{\delta}_c^2(q, x)}{2}. \tag{20}$$

This requires storing the original squared norm $\|x\|_2^2$ alongside each code and residual norm, adding one floating-point value per vector. For a fixed query, maximizing this score is equivalent to minimizing $\widehat{\delta}_c^2(q, x) - \|x\|_2^2$, since $\|q\|_2^2$ is constant. Keep the best L candidates under this corrected score, then rerank them by exact inner products on the original vectors. The absolute inner-product estimation error is exactly half the squared-distance estimation error, so Proposition 1 also gives a bound for this score under the same assumptions.

IVF considerations. For cosine search, both routing and scoring operate in the normalized space. For inner-product search, use a coarse routing rule targeting maximum inner product, for example ranking coarse centroids by $q^\top c_j$, rather than assuming that L2-nearest centroids give the same list order. This routing is approximate and its probe budget must be tuned for the target objective. Within each selected list, the same query-projection reuse $A(q - c_j) = Aq - Ac_j$ still applies. Scores must retain all list-dependent terms, including $\|q - c_j\|_2^2$, so candidates from different lists remain comparable. These adaptations preserve the target scoring objective; their empirical recall–qps trade-offs are not evaluated in this paper.

E.4 Experimental configuration

Direct and Residual use the same random projection matrix, whose entries are independently sampled from a standard Gaussian distribution. PCA-ITQ is trained on 20,000 randomly sampled database residuals for 50 iterations, using the full database arithmetic mean without additional sample-mean subtraction or whitening. Unless otherwise stated, the code length is $B = d$, where d is the original dimensionality; at this length, PCA retains all dimensions.

Choosing the reranking budget L. The budget L balances recall against the cost of full-precision distance computations. A practical selection procedure uses representative sampled queries with exact Top-K ground truth, while fixing the binary code encoder, code length, and K. First, increase L in SCAN until a desired high recall is reached, and use this budget as the initial value for IVF. Next, run IVF on the original vectors, computing exact distances for all points in the probed lists, to identify a probe count with sufficient recall. This checks list coverage independently of binary-distance estimation. Finally, fix that probe count and tune L around the SCAN-derived initial value, seeking the smallest tested budget that still meets the recall target after reranking. The initial value of L is a starting point rather than an upper bound. Increasing L cannot recover neighbors in unprobed lists, so adequate list coverage must be established before tuning the reranking budget.

Table 5 lists the fixed budgets used in Figure 5. IVF-ITQ uses a larger budget on MSong, GIST, and Tiny5M to achieve higher recall.

Table 5: Reranking budgets L used in Figure 5.

Dataset	IVF-Direct	IVF-ITQ	IVF-Residual
arXiv	1000	1000	500
MSong	1000	3000	500
GIST	1000	3000	500
Tiny5M	1000	3000	800

Hardware and timing. All experiments run on an AMD Ryzen 7 8845HS processor. AVX-512 is enabled for every evaluated method. Search is single-threaded, and queries are processed sequentially, one at a time. We measure end-to-end online search time, including query encoding and, where applicable, IVF-list selection, binary-distance estimation, candidate-heap maintenance, exact reranking, and final result generation. Offline index construction and ground-truth comparison are excluded from the timed region.

F Additional experiments

F.1 Effect of binary code length in SCAN

We evaluate SCAN-Residual under different binary code lengths to examine their effect on candidate quality. In general, longer code lengths produce more accurate estimates. Figure 14 shows that longer binary codes improve SCAN-Residual's recall across the tested reranking budgets on all four datasets. For example, on GIST with $L = 500$, Recall@20 increases from 0.58270 with $B = d/4$ to 0.81790 with $B = d/2$, 0.95055 with $B = d$, and 0.99350 with $B = 2d$, where d is the original dimensionality of the data. Similar trends are observed on arXiv, MSong, and Tiny5M. These gains show that increasing the binary code length leads to more accurate candidates, thereby improving

recall without changing the reranking budget L. But it comes at the cost of increased code storage and Hamming-distance computation.

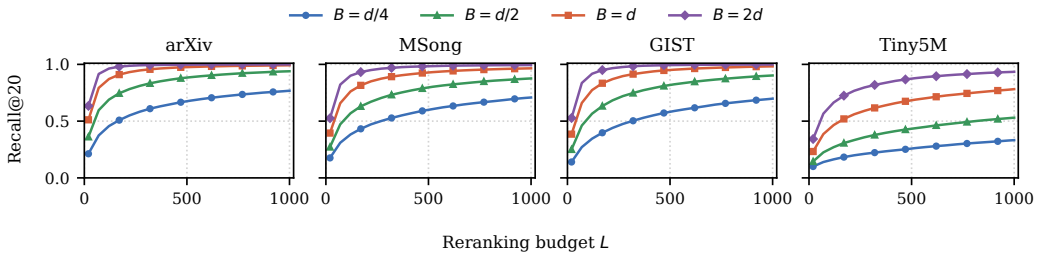

Figure 14: SCAN-Residual Recall@20 versus reranking budget L for four binary code lengths. Each panel uses the same dataset and global mean; d denotes the original dimensionality. Curves use one fixed projection seed.

F.2 Code length in IVF search

We then evaluate IVF-Residual under different binary code lengths to measure their effect on Recall-QPS trade-offs.Figure 15 shows the trade-off between code length, recall, and query speed. Shorter codes require less computations during search, but provide less accurate distance estimates. With a fixed reranking budget L, estimation errors can place true nearest neighbors outside the estimated Top-L, even when they are in the probed lists. These neighbors never reach exact reranking and cannot be recovered, limiting the achievable recall. Longer codes improve candidate ranking and allow more true neighbors to enter the reranking stage, but their additional computation can reduce QPS. For example, on Tiny5M with $n_{\text{probe}} = 128$ and $L = 800$, increasing B from d to $2d$ raises Recall@20 from 0.85440 to 0.94160, while QPS decreases from 744 to 610. Therefore, code length should be selected to balance the required recall against query speed: shorter codes favor QPS, whereas longer codes can support higher recall at greater computational cost.

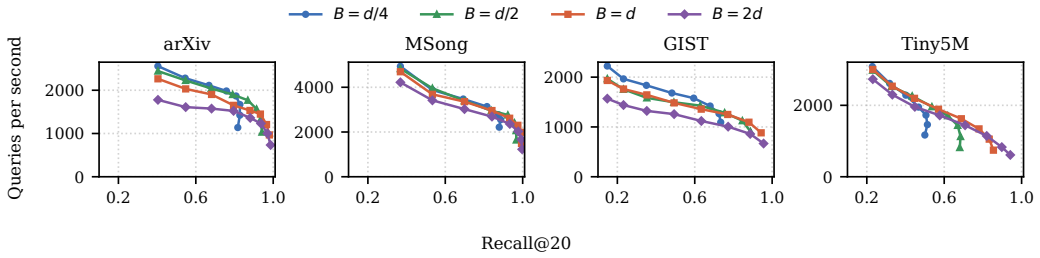

Figure 15: IVF-Residual Recall–QPS curves for four code lengths. Each curve varies the number of probed lists while keeping the dataset's reranking budget fixed. d is the original dimensionality.

F.3 Reranking budget in IVF search

We evaluate the performance of IVF-Residual under different reranking budgets L. With a smaller L, fewer points need to be reranked, so the search becomes faster, but the recall may drop; with a larger L, the recall improves, but the query speed decreases. Figure 16 shows this trade-off: increasing L improves recall but reduces QPS. For example, on the arXiv dataset, a smaller L increases QPS but reduces recall; when $L = 100$, high recall cannot be achieved. At $L = 300$ or 500, high query speed is achieved while maintaining high recall, and further increasing L reduces query speed. Therefore, a moderate reranking budget is sufficient when the candidate ranking already captures most true neighbors to achieve a good balance between recall and query speed.

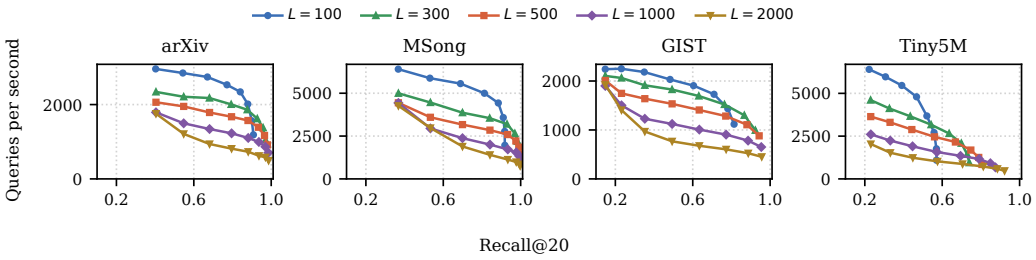

Figure 16: IVF-Residual Recall–QPS curves for different reranking budgets at fixed code length $B = d$. Each curve fixes L and varies the number of probed lists. The index and binary codes are shared across budgets within a dataset.

F.4 Index construction time

We compare offline construction costs on the same four unnormalized datasets used for search with L2 distance. We show the construction times from SCAN and IVF separately. Unlike the single-threaded query experiments, all builders have a budget of eight physical cores on the Ryzen 7 8845HS: we pin them to one hardware thread per core and run methods sequentially. OpenMP construction uses eight threads with single-threaded BLAS and no nested parallel teams; PCA-ITQ training instead uses eight BLAS threads. We report median wall-clock seconds over three fresh builds, including method-specific training, encoding, memory allocation, index layout, and required exact-reranking vector copies. Inputs are already resident in memory; file I/O and index destruction are excluded. Learned transforms and PQ codebooks are retrained in every repetition.

SCAN. Table 6 shows detailed construction times of Direct, ITQ, and Residual with $B = d$. Residual includes computing the global mean and residual norms. ITQ includes global-mean computation, PCA, and 50 ITQ iterations on 20,000 sampled vectors, followed by data points encoding. Each SCAN method uses the resident original vectors for reranking.

Table 6: SCAN in-memory construction time in seconds (median of three builds; eight-core budget).

Method	arXiv	MSong	GIST	Tiny5M
SCAN-Direct ($B = d$)	9.27	1.06	6.19	3.83
SCAN-ITQ ($B = d$)	24.79	7.23	29.05	10.29
SCAN-Residual ($B = d$)	10.03	1.20	6.29	4.26

IVF. All methods use identical, precomputed 4,096 coarse centroids and database assignments. Table 7 therefore reports construction *given fixed coarse partitions*, not end-to-end IVF construction: the common k-means training and database assignment are excluded for every method. Residual includes projecting the list-local centers and generating the posting-list codes. ITQ's full training cost is charged here as well. RaBitQ uses the same 1- and 2-bit builders as the search comparison. PQ uses Faiss 1.14.3 (The Faiss Developers, 2026) with four dimensions per subquantizer, 25 training iterations, and 4- or 8-bit codes. With Faiss's default training-sample cap, PQ4 and PQ8 use 4,096 and 65,536 training residuals per subquantizer, respectively. PQ4-FastScan also includes conversion to its block-packed layout; sharing a training pass with PQ4-ADC in the benchmark does not remove that training cost from either reported total.

For the SCAN part, SCAN-Residual has a construction time similar to that of SCAN-Direct, and offers a clear performance advantage over SCAN-ITQ. This is because our SCAN-Residual avoids iterative transform training, resulting in faster construction. For the IVF part, Residual has a slower construction speed than RaBitQ, but still holds an advantage over PQ, achieving lower construction time than PQ on some datasets. Although Residual avoids the training of a learned quantizer, it still requires dense random projection, which accounts for a large overhead.

Table 7: IVF in-memory construction time in seconds, excluding common coarse clustering and assignment (median of three builds; eight-core budget). PQ4 and PQ8 have nominal code lengths d and $2d$ bits, respectively; auxiliary storage and method-specific padding are additional.

Method	arXiv	MSong	GIST	Tiny5M
IVF-Direct ($B = d$)	13.00	1.78	8.85	7.46
IVF-ITQ ($B = d$)	28.28	8.05	30.32	13.46
IVF-Residual ($B = d$)	12.99	1.75	7.74	6.45
IVF-Residual ($B = 2d$)	23.59	3.11	16.35	13.07
IVF-RaBitQ 1-bit	3.78	0.96	2.04	4.36
IVF-RaBitQ 2-bit	17.52	4.29	9.84	18.02
IVF-PQ4 (float ADC)	8.01	1.96	4.52	7.57
IVF-PQ4 (FastScan)	7.98	1.98	4.64	7.63
IVF-PQ8 (float ADC)	16.49	6.19	13.51	14.30

F.5 Residual scoring with ITQ codes

We also test our distance estimator with binary codes generated by ITQ. Keeping the codes fixed, we compare selecting candidates by Hamming distance alone with using our estimator, which also accounts for each vector's distance to the center. We compare SCAN-Direct, SCAN-ITQ, SCAN-Residual, and an additional variant, *SCAN-Residual-ITQ*. The two ITQ variants use exactly the same global mean and trained PCA-ITQ transform (Gong et al., 2013), and therefore generate the same binary codes. SCAN-ITQ ranks candidates by Hamming distance, whereas SCAN-Residual-ITQ uses Equation 7, with H_c computed between the ITQ codes and ρ_q, ρ_x measured from the original vectors to the global mean. Thus, the comparison changes only the candidate-scoring rule, not the encoder or the center. The norm-aware score requires one additional stored floating-point residual norm per vector relative to a codes-only ITQ representation.

We use all 1,000 queries from each of the four datasets, $K = 20$, and $B = d$, with $L \in \{20, 70, \ldots, 970, 1000\}$. The PCA-ITQ models use the training configuration in Appendix E.4; at $B = d$, the full-dimensional orthogonal transform introduces no PCA truncation or whitening. Both ITQ variants are re-evaluated using these same saved models, while Direct and Residual reuse the measurements from Experiment 1. All methods scan the entire database and rerank candidates by exact L2 distances on the original vectors. This experiment measures recall at fixed reranking budgets, not QPS.

For Section 4.4, we also measure bit statistics over the full database using the saved ITQ models and the C++ search encoder. On GIST, ITQ positive fractions range from 0.4598 to 0.5448 and mean bit entropy is 0.998024. The direct random-projection control in Appendix F.6 has mean entropy 0.513160. These are search-encoder statistics; the independently generated NumPy projections used for the distribution plots give slightly different values.

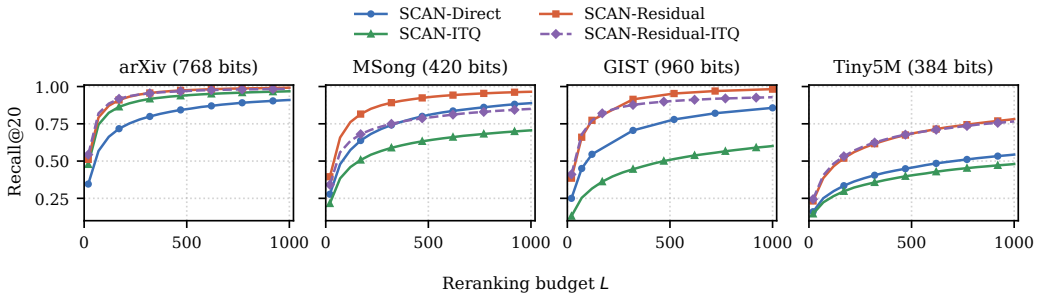

Figure 17: Recall@20 versus reranking budget L with $B = d$. SCAN-ITQ and SCAN-Residual-ITQ share identical mean-centered ITQ codes but rank candidates by Hamming distance and norm-aware residual scoring, respectively. All methods use exact L2 reranking.

Figure 17 shows that SCAN-Residual-ITQ improves over SCAN-ITQ at every tested budget on all four datasets. At $L = 1000$, Recall@20 increases from 0.96830 to 0.98435 on arXiv, from 0.70625 to 0.85000 on MSong, from 0.60090 to 0.92865 on GIST, and from 0.48000 to 0.76590 on Tiny5M. Since ITQ already uses mean centering, these gains demonstrate the benefit of incorporating residual lengths into candidate scoring, rather than adding centering or increasing the reranking budget. At this budget, the random-projection SCAN-Residual variant still achieves higher recall on all four datasets. The results therefore support the usefulness of residual scoring with this learned encoder, but do not show that replacing random projection with ITQ is uniformly beneficial.

For learned ITQ codes, the mapping $\cos(\pi H_c/B)$ is an empirical angle approximation: the independent Gaussian-row assumption of Proposition 1 does not hold. This experiment establishes empirical compatibility with ITQ, not a transfer of the random-hyperplane error guarantee to arbitrary learned hashing methods.

F.6 Separating bit balance from norm-aware scoring

We test whether correcting bit imbalance alone can explain the retrieval gains of NARH. All variants share the same Gaussian projection matrix A with $B = d$, $K = 20$, and $L \in \{20, 70, \ldots, 970, 1000\}$. We use all 1,000 queries from each dataset, scan the entire database, and rerank candidates using exact L2 distances on the original, unnormalized vectors. This experiment compares recall at fixed budgets L.

Paired controls. We evaluate three encoders, each with two candidate-scoring rules. *Direct + norm* is SCAN-Direct: it encodes the original vectors with zero-threshold random hyperplanes and ranks candidates using Equation 4. *Direct + Hamming* uses exactly the same codes but selects candidates solely by Hamming distance. Likewise, *Residual (NARH)* and *Mean-centered + Hamming* share the same mean-centered codes; only the former uses residual norms in Equation 7.

For the third encoder, we fit a separate threshold $\tau_j = \mathrm{median}_{x \in X}\, a_j^\top x$ for each bit and encode both database and query vectors as $b_j(x) = \mathbb{I}[a_j^\top x \geq \tau_j]$. Median-based hyperplane adaptation is also used by Shin et al. (2026); here we isolate this thresholding rule in a software SCAN experiment. Thresholds are computed from the full database only, without queries or ground truth. Even-sized samples use the average of the two middle values, and threshold ties use the same $\geq$ rule for database and query encoding. We compare *Median-threshold + Hamming* with *Median-threshold + raw norm*, which substitutes these codes into the original-norm score in Equation 4. The latter is a heuristic: shifted thresholds do not retain the original-angle collision law. All six variants perform exact reranking after candidate selection.

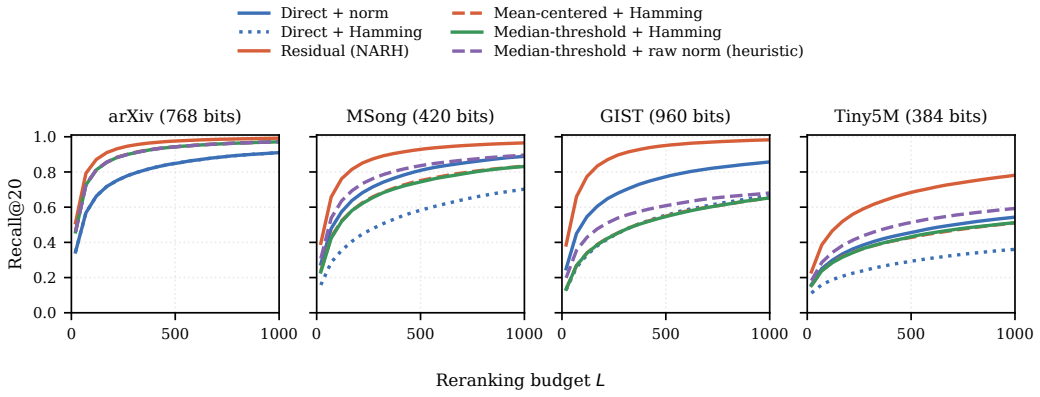

Figure 18: Bit-balance and scoring ablation: Recall@20 versus reranking budget L with $B = d$. Each pair of scoring rules shares identical binary codes. All measured budgets are retained.

Results. Table 8 confirms that median thresholds raise marginal bit entropy to nearly one. Nevertheless, Figure 18 shows that NARH outperforms both median-threshold variants and mean-centered

Table 8: Mean per-bit entropy over the full database. Median thresholds produce nearly balanced bits on all four datasets; values are rounded to six decimal places. Repeated projection values can prevent exact 50/50 balance.

Dataset	Direct	Mean-centered	Median-threshold
arXiv	0.552207	0.999818	1.000000
MSong	0.470434	0.972310	1.000000
GIST	0.513160	0.998702	1.000000
Tiny5M	0.494500	0.998894	1.000000

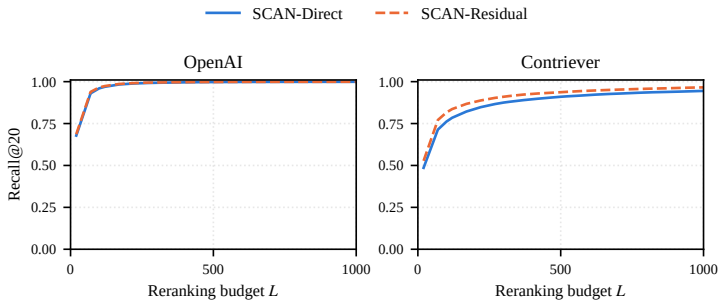

Figure 19: SCAN Recall@20 versus reranking budget L on OpenAI and Contriever with $B = d$.

Hamming ranking at every tested budget on all four datasets. At $L = 1000$, median-threshold Hamming ranking achieves Recall@20 of 0.97145, 0.83110, 0.65250, and 0.51185 on arXiv, MSong, GIST, and Tiny5M, respectively, compared with 0.99215, 0.96500, 0.98260, and 0.78160 for NARH. Adding the original-norm heuristic improves the median-threshold baseline on some datasets, but does not close this gap.

The identical-code comparison more directly isolates the role of scoring. On GIST at $L = 1000$, mean-centered Hamming ranking achieves only 0.65335 despite a mean bit entropy of 0.998702; using residual norms raises recall to 0.98260 without changing any code. These results show that balancing bits alone is insufficient for accurate L2 candidate ranking. Even with identical mean-centered codes, using residual lengths improves recall over Hamming ranking.

F.7 Boundary cases: OpenAI and Contriever

We additionally evaluate OpenAI and Contriever to examine the effect of centering on datasets that do not exhibit strong directional concentration. The purpose is to test the limits of the benefit from centering, rather than to assume that translation cannot help on these datasets.

Setup. OpenAI contains 999,000 database vectors with $d = 1536$ and 1,000 queries. Contriever contains 990,000 database vectors with $d = 768$; we use the first 1,000 of its 10,000 queries. Both experiments use the original vectors with no additional normalization and evaluate exact L2 neighbors.

Figure 19 shows the recall curves for OpenAI and Contriever. SCAN-Residual improves Recall@20 over SCAN-Direct by only 0.020–0.870 percentage points on OpenAI across the 24 tested rerank budgets L. At $L = 1000$, recall is 0.99910 for Direct and 0.99935 for Residual. For Contriever dataset, recall improves from 0.76160 to 0.81605 at $L = 100$, and from 0.94485 to 0.96585 at $L = 1000$. Across its tested budgets, gains range from 2.100 to 5.735 percentage points.

Paired geometry and bit statistics. Follow the setup of Experiment 3 and Table 4, we show the statistics difference of before and after centering for OpenAI and Contriever datasets. Figure 20 presents the paired distributions, and Table 9 reports the same statistics as Table 4.

Mean angles are 85.06° on OpenAI and 80.02° on Contriever, versus 37.14°–45.40° on the four main datasets. OpenAI vector pairs are thus already nearly orthogonal on average, rather than strongly

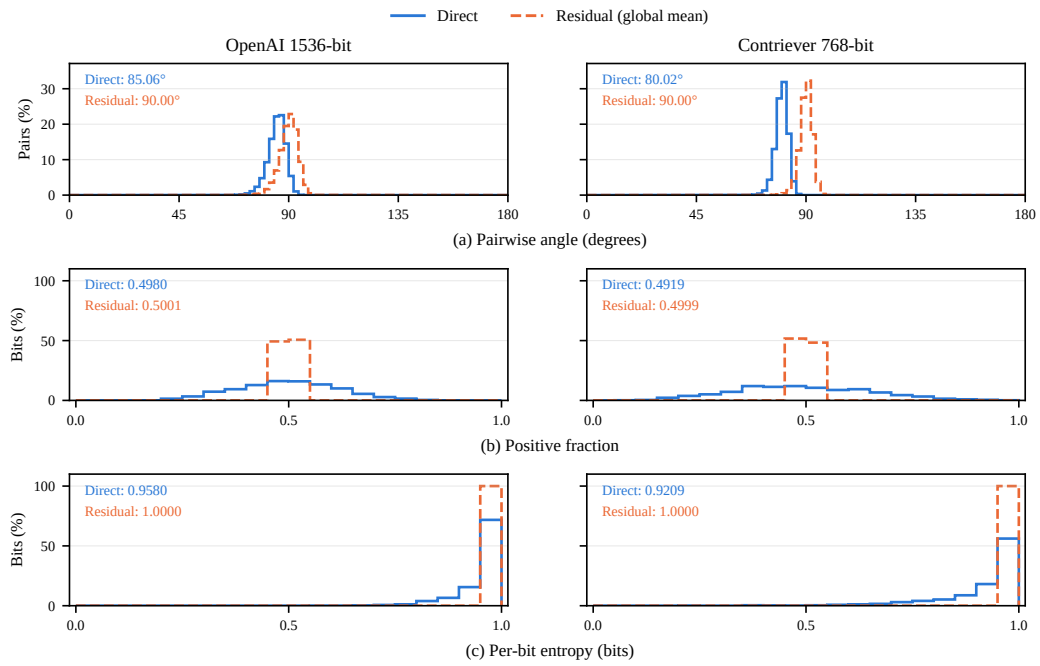

Figure 20: Direct versus Residual: angle, positive-fraction, and per-bit entropy distributions on on OpenAI and Contriever datasets ($B = d$). Annotations show the mean of each distribution.

Table 9: Paired boundary-case diagnostics using the definitions in Table 4. Mean angles and IQR widths are in degrees; the last column counts bits with entropy below 0.1.

Dataset	Method	Mean angle	IQR	$\bar{p}$	$\overline{\mathcal{H}}$	Δ_p	$\mathcal{H}_j < 0.1$
OpenAI	Direct	85.06	4.66	0.4980	0.9580	0.0959	0/1536
OpenAI	Residual	90.00	4.76	0.5001	1.0000	0.0019	0/1536
Contriever	Direct	80.02	3.23	0.4919	0.9209	0.1318	0/768
Contriever	Residual	90.00	3.18	0.4999	1.0000	0.0018	0/768

aligned in a shared direction. Direct codes have mean bit entropies of 0.9580 and 0.9209, respectively, with no bits below 0.1 under the tested projection. Centering moves both means to approximately 90° and mean bit entropy to approximately one. Angular IQRs barely change: 4.66° to 4.76° on OpenAI and 3.23° to 3.18° on Contriever. Thus, centering need not widen the angle distribution, and a narrow angle band alone does not imply severe bit imbalance.

Our method is motivated by datasets with strong directional concentration. The smaller recall gains on OpenAI and Contriever are therefore consistent with this motivation, since neither dataset exhibits this property as strongly as the four main datasets. On OpenAI, centering provides little additional recall at large reranking budgets, whereas Contriever shows a moderate improvement. These results illustrate that the benefit of centering depends on the dataset's directional geometry.