AC-SCOREQ: AMPLITUDE-COUPLED SCALABLE COMPLEX 2-BIT RESIDUAL QUANTIZATION FOR EFFICIENT VECTOR RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Efficient vector retrieval requires compact representations that balance storage, ranking quality, and search time. Successive low-bit phase-code applications compensate quantization residuals, but their shared block scales leave local magnitude differences that further phase refinement cannot directly represent. We propose AC-Scoreq, a fixed-codebook complex 2-bit residual quantizer that combines phase encoding, local amplitude adaptation, and cross-stage scale fitting. It applies a fixed four-direction phase codebook to successive residuals; the final application additionally selects a relative-amplitude shape for each pair of adjacent complex coordinates. The encoder alternates discrete-code refinement with joint fitting of the block scales, and each database vector is encoded independently without corpus-level codebook training or prior knowledge of queries. Varying the number of phase-code applications enables trade-offs among retrieval quality, search time, and compression ratio. Experimental results across multiple datasets and vector dimensionalities show that AC-Scoreq maintains competitive retrieval quality and low search latency while substantially reducing vector storage overhead. Its multi-application design further provides flexible accuracy compensation across compression ratios, demonstrating the method's effectiveness, scalability, and practical deployment value.

1 INTRODUCTION

Semantic retrieval, recommendation, and vector databases represent items as high-dimensional embeddings and rank them by inner product, cosine similarity, or Euclidean distance (Reimers & Gurevych, 2019; Karpukhin et al., 2020). These similarity computations are also central to approximate nearest neighbor (ANN) search (Indyk & Motwani, 1998; Johnson et al., 2019; Malkov & Yashunin, 2018; Aumüller et al., 2020). At database scale, storing and scanning full-precision vectors creates substantial memory, bandwidth, and computation costs (Johnson et al., 2019; Jegou et al., 2010). Vector quantization reduces these costs, but a low-bit representation must balance storage, ranking fidelity, search efficiency, and ease of deployment.

Product quantization (PQ) and its variants learn compact subspace codebooks, often attaining strong accuracy at the cost of training and adaptation when the data distribution changes (Jegou et al., 2010; Ge et al., 2013; Kalantidis & Avrithis, 2014; Guo et al., 2020). More recent low-bit methods avoid or reduce corpus-dependent training. RaBitQ and its multi-bit extension provide compact estimators with theoretical guarantees (Gao & Long, 2024; Gao et al., 2025); RSLM combines fixed Lloyd–Max codes with structured rotations and norm correction (Lenhardt et al., 2026); and TurboQuant uses data-oblivious rotations and distribution-aware scalar quantization (Zandieh et al., 2026).

A key challenge in fixed low-bit quantization is how to allocate a limited coding budget between directional and magnitude information. In block-scaled phase quantization, each coordinate selects a fixed direction while multiple coordinates share one magnitude scale. Additional residual applications improve the aggregate approximation, but every application remains subject to this shared-scale constraint. Heterogeneous local magnitudes may therefore require further residual compensation even after the available phase directions provide a reasonable directional approximation. This motivates our central question: can a small amount of relative-amplitude information use a fixed

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.

phase codebook more effectively while preserving compact storage and structured similarity computation?

We adopt a complex representation to organize phase and magnitude within one algebra for encoding, residual updates, and scoring. AC-Scoreq first applies Randomized Hadamard Rebalancing (RHR), then repeatedly uses the fixed four-direction phase codebook $\{1, i, -1, -i\}$. Its representation has three explicit granularities: each complex coordinate retains an independent 2-bit phase code; each adjacent coordinate pair shares one 2-bit relative-amplitude shape in the final application; and each block has one scale per application. The shape redistributes a block scale within a pair, providing local magnitude adaptation without an independent magnitude parameter for every coordinate. These shapes are part of the additive reconstruction rather than a post-processing correction. The encoder jointly refits scales across applications and alternates this fit with phase and shape updates, producing a unified phase–amplitude–scale representation. All codebooks are fixed analytically, and database vectors are encoded independently without corpus-level codebook training or prior knowledge of queries.

The number of phase-code applications T is chosen when database vectors are encoded. Each application adds a prescribed amount of phase-code and scale metadata and a corresponding scoring contribution, while the final application also stores AC shapes. Thus, one fixed-codebook algorithm yields discrete, independently encoded operating points that trade storage and computation for retrieval quality; these are not separately trained models, continuously adjustable rates, or a progressively truncatable bitstream. We make comparisons fair and interpretable by evaluating methods at their native discrete operating points, reporting actual bytes per vector, and comparing configurations in nearby storage regimes. Accordingly, our claims concern storage–latency–quality trade-offs at comparable budgets, not strict equal-byte superiority. Equal-amplitude and same-budget extra-phase controls separately isolate the value of adaptive relative-amplitude coding and its use of the additional bits.

The phase codes support a structured similarity identity: a code selects a signed real or imaginary query component, blockwise sums are weighted by stored scales, and the application contributions are accumulated. Our empirical study focuses on the quantization component of vector retrieval through full-corpus semantic search, without an ANN index. We evaluate FIQA and DBPedia-Entity with ModernBERT, Qwen-1.5B, and Qwen-7B embeddings under inner-product, cosine, and L2-based rankings. Recall@100 uses relevance labels rather than FP32 exact neighbors. For the Qwen-7B timing experiment, we measure complete full-corpus search over the evaluation queries, including direct packed-code scoring and top-k selection, rather than the cost of an isolated scoring primitive. This scope lets us examine how compressed representations affect retrieval quality and search cost without adding approximation from an ANN index.

The methodological contribution is the phase–amplitude–scale construction under a fixed low-bit codebook: coordinate-level phases preserve directional choices, pair-level shapes introduce compact relative-amplitude freedom, and jointly fitted block scales coordinate all residual applications. Varying T is a useful framework property rather than a claim that residual staging itself is new; it exposes independently encoded operating points using the same construction. Table 3 directly tests the central bit-allocation question by comparing AC with equal-amplitude and same-budget extra-phase controls.

The main contributions are:

1. **Amplitude-coupled complex quantization.** We introduce a compact phase–amplitude representation that combines independent 2-bit complex phase codes with one shared 2-bit relative-amplitude shape per adjacent coordinate pair. It adapts local magnitudes without storing independent per-coordinate scales.

2. **Joint refinement and structured scoring.** We integrate this representation into fixed-codebook residual quantization and jointly fit block scales while alternating phase and shape updates. The phase directions support direct packed-code similarity computation through signed query-component selection, block accumulation, and scale weighting.

3. **Configurable operating points and evaluation.** Varying the number of residual applications provides independently encoded storage–quality–latency choices without corpus-level codebook training. We evaluate them across two datasets, three embedding models,

and three similarities, and use equal-amplitude and same-budget extra-phase controls to isolate relative-amplitude coding.

2 Related Work

Data-dependent vector quantization. PQ partitions a vector into subspaces and represents each subvector with a learned centroid (Jegou et al., 2010). OPQ learns an orthogonal rotation before PQ to distribute quantization difficulty more evenly across subspaces, while LOPQ uses coarse partitions and locally optimized transformations (Ge et al., 2013; Kalantidis & Avrithis, 2014). AVQ modifies the quantization objective to better preserve maximum-inner-product rankings (Guo et al., 2016; 2020). These approaches are effective, but their learned transformations or codebooks must be trained from representative database samples and may require adaptation after distribution shifts.

Residual and additive quantization. Residual vector quantization (RVQ) and additive quantization (AQ) represent vectors as sums of contributions from multiple codebooks; RVQ selects successive codes for the remaining residual, whereas AQ optimizes the combined representation. Residual coding is also used in neural codec pipelines. AC-Scoreq adopts this established additive reconstruction principle rather than presenting residual staging itself as a new contribution. Its distinction lies in the structure and granularity of each contribution: it repeatedly uses one fixed four-direction phase codebook, adds pairwise relative-amplitude shapes only in the final application, and jointly refits block scales across applications. The focus is therefore compact local magnitude adaptation within fixed-codebook residual quantization, not residual summation alone.

Low-bit and training-free quantization. Binary hashing methods such as SimHash and ITQ compress vectors aggressively but retain limited magnitude information (Charikar, 2002; Gong et al., 2012). RaBitQ represents rotated vectors with 1-bit codes and provides a bound for distance estimation; extended RaBitQ generalizes the estimator to multiple bits (Gao & Long, 2024; Gao et al., 2025). RSLM applies cascaded structured rotations, fixed Lloyd–Max quantizers, and norm correction, with RSLM4Lite embedding its scale metadata into the packed code (Lenhardt et al., 2026). TurboQuant combines data-oblivious random rotation with scalar quantizers designed for the resulting coordinate distribution and establishes near-optimal distortion rates (Zandieh et al., 2026). These methods motivate fixed or data-oblivious alternatives to learned PQ codebooks.

Relative to these approaches, AC-Scoreq is organized by representation granularity and query-time structure: coordinate-level phase codes select signed query components, pair-level shapes distribute relative amplitude, and block/application-level scales weight accumulated contributions. Alternating code refinement and cross-application scale fitting coordinate these three levels. This construction retains direct scoring from packed codes and requires neither corpus-level codebook training nor a full-precision reranking vector. We do not claim residual coding or training-free quantization itself as new; the contribution is their specific phase–amplitude–scale integration.

3 Method

3.1 Problem Setup

Let $X = \{x_i\}_{i=1}^{N} \subset \mathbb{R}^d$ be a database and $q \in \mathbb{R}^d$ a query. AC-Scoreq independently compresses database embeddings to approximately preserve rankings under inner product, cosine similarity, and Euclidean distance. Queries undergo the same coordinate transform but remain unquantized.

AC-Scoreq applies a fixed complex phase codebook T times to successive residuals. Each application has phase codes and block scales; the final application additionally assigns a relative-amplitude shape to each adjacent complex-coordinate pair. Amplitude coupling (AC) is part of application T, not an extra stage.

Before quantization, the default AC-Scoreq configuration applies Randomized Hadamard Rebalancing (RHR) to an input vector:

$$\tilde{x} = S\Pi H_B x, \tag{1}$$

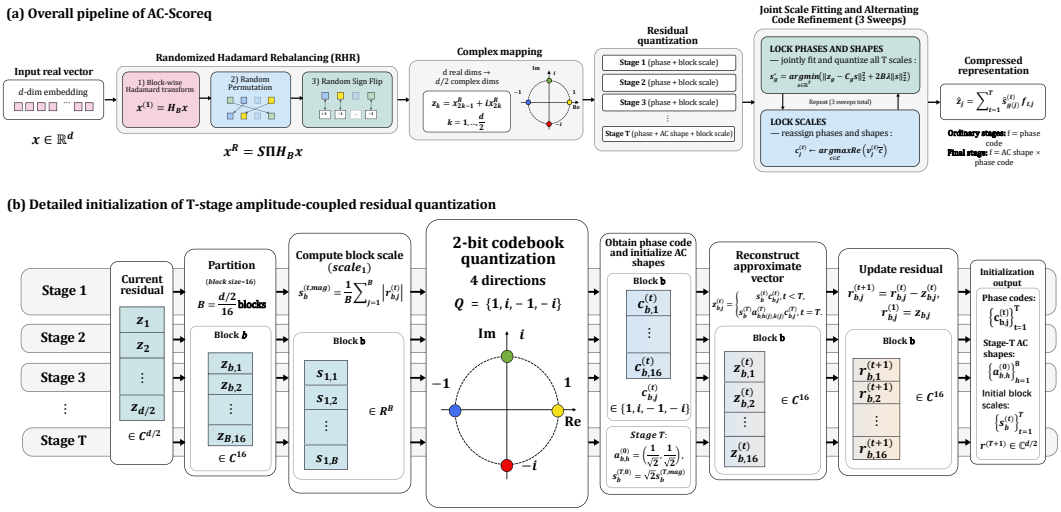


Figure 1: AC-Scoreq pipeline. (a) RHR and complex mapping precede T residual phase-code applications; the final application adds AC shapes. Three refinement sweeps alternate joint scale fitting and code reassignment. (b) Initialization selects phases and block scales from successive residuals and starts the final application's shapes at equal amplitude.

where Π is a fixed random permutation matrix, S is a fixed random sign-flipping matrix, and

$$H_B = \mathrm{diag}(\bar{H}_{128}, \ldots, \bar{H}_{128}), \qquad \bar{H}_{128} = \frac{H_{128}}{\sqrt{128}}. \tag{2}$$

The same orthogonal transform is applied to queries and database vectors, preserving their full-precision inner products and distances. RHR redistributes coordinate energy using a fast block Hadamard transform (Ailon & Chazelle, 2009); without preprocessing, $\tilde{x} = x$.

3.2 Scalable Complex Residual Quantization

3.2.1 Complex Phase Codes with Block-Level Scales

AC-Scoreq pairs adjacent real coordinates of $\tilde{x}$ into complex coordinates:

$$z_j = \tilde{x}_{2j-1} + i\tilde{x}_{2j}, \qquad j = 1, \ldots, d/2. \tag{3}$$

This form separates pairwise direction from magnitude and supports phase selection, residual updates, AC, and scoring in one algebraic representation. Every application uses the fixed four-direction codebook

$$\mathcal{C} = \{1, i, -1, -i\}. \tag{4}$$

Given a complex residual r_j, the nearest phase direction maximizes its real projection:

$$c_j = \arg\max_{c \in \mathcal{C}} \ \mathrm{Re}(r_j \bar{c}). \tag{5}$$

The orthogonal phase directions select signed real or imaginary components; no general complex multiplication is needed for a phase projection. Each coordinate uses 2 phase bits. Every $B = 16$ complex coordinates form a block B_g and share a real scale s_g, giving contribution $s_g c_j$ in an ordinary application.

For the first $T - 1$ applications, the encoder tries two block-scale initializations for residual r and selected phases c_j:

$$s_g^{\mathrm{mag}} = \frac{1}{B} \sum_{j \in B_g} |r_j|, \qquad s_g^{\mathrm{proj}} = \frac{1}{B} \sum_{j \in B_g} \mathrm{Re}(r_j \overline{c_j}). \tag{6}$$

The projection value minimizes single-application squared error when the phase codes are fixed. Both initializations undergo the same joint refinement, and the lower-error candidate is retained.

3.2.2 Scalable Multi-Stage Residual Compensation

Set $r^{(1)} = z$. For applications $t = 1, \ldots, T-1$, AC-Scoreq selects phase codes and block scales for the current residual, then passes the remaining error to the next application:

$$\Delta z_j^{(t)} = s_{g(j)}^{(t)} c_j^{(t)}, \qquad r_j^{(t+1)} = r_j^{(t)} - \Delta z_j^{(t)}, \qquad t < T. \tag{7}$$

Application T also selects a 2-bit phase code for each coordinate, while additionally using the amplitude shapes defined below. The reconstructed complex vector is

$$\hat{z}_j = \sum_{t=1}^{T} \Delta z_j^{(t)}. \tag{8}$$

Thus, $T = 1, 2, 3$ contain respectively zero, one, or two ordinary applications before the AC-augmented final application. Configurations are independently encoded, not nested bitstreams.

We evaluate $T \in \{1, 2, 3\}$; gains diminish by $T = 3$ in our experiments, where compression remains near $8\times$ (Section 3.4).

3.2.3 Amplitude Coupling in the Final Phase-Code Application

Repeated phase coding reduces directional error, but a shared block scale cannot represent unequal coordinate magnitudes. For residual u, nonnegative scale s, and unit phase c at angular difference θ,

$$|u - sc|^2 = (|u| - s)^2 + 2s|u|(1 - \cos\theta). \tag{9}$$

Even perfect phase alignment leaves the radial term when $|u| \neq s$. This motivates spending the final application's extra bits on relative amplitude rather than further phase precision.

As a single-application illustration, consider two phase-aligned residuals with magnitudes 2 and 1. Equal-amplitude coding with one shared scale has error $(2 - s)^2 + (1 - s)^2$, minimized at $s = 1.5$ with error 0.5. A relative-amplitude shape instead reconstructs them as $s(a_1, a_2)$ and can allocate unequal magnitudes while retaining one scale. The finite set in Eq. (10) does not represent every ratio exactly, and additional equal-amplitude residual applications can also reduce this error; the example isolates the representation freedom supplied by the pairwise shape.

AC is used only in application T for two reasons. First, the preceding applications repeatedly correct the dominant directional components, so the remaining error increasingly exposes the local magnitude mismatch caused by a shared block scale. Second, repeating shape fields in every application would increase the rate linearly and weaken the stage-count trade-off. Concentrating the shape bits in the final application therefore preserves inexpensive phase-only residual compensation in the earlier applications while assigning one compact correction to the error component they cannot represent directly. The same-budget comparison in Section 4 empirically tests this allocation against additional phase information.

In application T, two adjacent complex coordinates share one 2-bit shape code, retaining separate phase codes without the cost of a separate amplitude code per coordinate. The same-budget extra-phase ablation in Table 3 tests this bit allocation. Each pair h selects $a_h = (a_{h,1}, a_{h,2})$ from the analytic set

$$\mathcal{A} = \left\{ (0,0),\ \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right),\ \left(\cos\frac{\pi}{8}, \sin\frac{\pi}{8}\right),\ \left(\sin\frac{\pi}{8}, \cos\frac{\pi}{8}\right) \right\}. \tag{10}$$

The equal shape preserves equal magnitudes; two unit-norm asymmetric shapes place more amplitude on either coordinate; and $(0, 0)$ suppresses an already explained pair. The asymmetric directions, at $\pi/8$ and $3\pi/8$, symmetrically bracket the equal direction. This set is a compact analytic choice, not an optimality claim. For pair $h = (j_1, j_2)$ in block B_g,

$$\left(\Delta z_{j_1}^{(T)}, \Delta z_{j_2}^{(T)}\right) = s_g^{(T)} \left(a_{h,1} c_{j_1}^{(T)}, a_{h,2} c_{j_2}^{(T)}\right). \tag{11}$$

Each coordinate keeps its phase code while the pair's shape distributes the shared scale. For a target residual pair $u = (u_1, u_2)$, fixed phases c_1, c_2, scale $s \geq 0$, and projections $p_k = \mathrm{Re}(u_k \overline{c_k})$, its error is

$$E_h(a; s) = \sum_{k=1}^{2} |u_k - s a_k c_k|^2 = \sum_{k=1}^{2} |u_k|^2 + s^2 \|a\|_2^2 - 2s \sum_{k=1}^{2} p_k a_k. \tag{12}$$

The nonzero shapes have unit norm, so selecting among them maximizes $p^\top a$; the zero shape can suppress a weak pair. The three nonzero directions cover the nonnegative amplitude quadrant with worst-case angular mismatch $\pi/8$; with a pair-specific optimal scale, the corresponding excess error over a continuous direction is at most $\|p\|_2^2 \sin^2(\pi/8)$. Including the equal-amplitude shape ensures that, with the other variables fixed, shape selection cannot worsen that pair's error relative to equal amplitudes. Eight pairs per block select shapes independently, adding $d/2$ bits per vector only in application T. Appendix A gives the final application's scale initialization and the scope of this conditional geometric bound.

3.2.4 Joint Scale Fitting and Alternating Code Refinement

Residual coding initializes the representation. Let $h(j)$ denote the pair containing j and $k(j) \in \{1, 2\}$ its position. Define the application basis

$$f_{t,j} = \begin{cases} c_j^{(t)}, & t < T, \\ a_{h(j),k(j)} c_j^{(T)}, & t = T. \end{cases} \tag{13}$$

Before final scale quantization, the reconstruction is

$$\hat{z}_j = \sum_{t=1}^{T} s_{g(j)}^{(t)} f_{t,j} = \sum_{t=1}^{T-1} s_{g(j)}^{(t)} c_j^{(t)} + s_{g(j)}^{(T)} a_{h(j),k(j)} c_j^{(T)}. \tag{14}$$

Let $C_g \in \mathbb{C}^{B\times T}$ contain $f_{t,j}$ in column t. Holding phase and shape codes fixed, jointly fit the block's real scales by

$$s_g^\star = \arg\min_{s\in\mathbb{R}^T} \|z_g - C_g s\|_2^2 + 2B\lambda\|s\|_2^2, \qquad B = 16. \tag{15}$$

The ridge weight $2B$ matches the number of real coordinates per block. Since the scales are real, the joint fit solves

$$\left[\mathrm{Re}(C_g^H C_g) + 2B\lambda I\right] s_g^\star = \mathrm{Re}(C_g^H z_g), \tag{16}$$

where H denotes conjugate transpose. Negative scales are absorbed by rotating the corresponding block's phase codes by π. The fitted scales are then quantized and held fixed while codes are updated. For application t, subtracting the other applications gives its leave-one-out target:

$$v_j^{(t)} = z_j - \sum_{\substack{u=1 \\ u\neq t}}^{T} \hat{s}_{g(j)}^{(u)} f_{u,j}. \tag{17}$$

Ordinary phase codes are reselected by

$$c_j^{(t)} \leftarrow \arg\max_{c\in\mathcal{C}} \mathrm{Re}\left(v_j^{(t)} \overline{c}\right), \qquad t < T. \tag{18}$$

In the final application, phases are reselected from $v^{(T)}$, then each pair $h = (j_1, j_2)$ chooses its shape by

$$a_h \leftarrow \arg\min_{a\in\mathcal{A}} \sum_{k=1}^{2} \left| v_{j_k}^{(T)} - \hat{s}_{g(h)}^{(T)} a_k c_{j_k}^{(T)} \right|^2. \tag{19}$$

Thus, each sweep first fits all T block scales with codes fixed, then fixes the scales and updates the discrete codes. Three sweeps follow two preliminary ordinary-application sweeps and AC initialization, with a final joint scale fit. To avoid storing one floating-point scale per block, application t stores one FP16 base and quantizes its nonnegative block scales to uint8:

$$b_t = \mathrm{FP16}\left(\max_g s_g^{(t)}\right), \quad u_g^{(t)} = \mathrm{clip}_{[0,255]}\left[\mathrm{round}\left(\frac{255 s_g^{(t)}}{b_t}\right)\right], \quad \hat{s}_g^{(t)} = \frac{u_g^{(t)} b_t}{255}. \tag{20}$$

Algorithm 1 gives the complete procedure. All phase and shape codebooks are fixed analytically and are never trained.

3.3 Similarity Computation from Quantized Representations

Let $w_j = \tilde{q}_{2j-1} + i\tilde{q}_{2j}$ denote the transformed query. Its approximate real inner product with the coded database vector is

$$\widehat{\langle q, x \rangle} = \mathrm{Re}\left(\sum_{j=1}^{d/2} w_j \overline{\hat{z}_j} \right). \tag{21}$$

The database coordinate is conjugated, so $\mathrm{Re}((a+ib)(u-iv)) = au + bv$. Substitution of Eq. (14) gives

$$\begin{aligned} \widehat{\langle q, x \rangle} = & \sum_{g=1}^{G} \sum_{t=1}^{T-1} \hat{s}_g^{(t)} \sum_{j \in B_g} \mathrm{Re}\left(w_j \overline{c_j^{(t)}} \right) \\ & + \sum_{g=1}^{G} \hat{s}_g^{(T)} \sum_{\substack{h=(j_1,j_2) \\ h \subset B_g}} \left[a_{h,1} \,\mathrm{Re}\left(w_{j_1} \overline{c_{j_1}^{(T)}} \right) + a_{h,2} \,\mathrm{Re}\left(w_{j_2} \overline{c_{j_2}^{(T)}} \right) \right]. \end{aligned} \tag{22}$$

For $w = a + ib$, the four phase projections are $a, b, -a, -b$ (Appendix A). Equation (22) therefore uses signed-component selection, block accumulation, and scale weighting, with pairwise shape weights in application T.

Precomputed quantized-vector norms extend the inner product to cosine similarity and negative squared L2 distance:

$$\cos(q, \hat{x}) = \frac{\widehat{\langle q, x \rangle}}{\|q\|_2 \|\hat{x}\|_2}, \qquad s_{\mathrm{L2}}(q, \hat{x}) = -\left(\|q\|_2^2 + \|\hat{x}\|_2^2 - 2\widehat{\langle q, x \rangle} \right). \tag{23}$$

Maximizing s_{L2} is equivalent to minimizing L2 distance. All three measures use the same database codes; Appendix E details the timed retrieval pipeline.

3.4 Storage and Computational Complexity

With $G = d/32$ blocks, each ordinary application stores $d/8$ bytes of phase codes. The final application stores $3d/16$ bytes of phase and pairwise-shape codes. Each application also needs G uint8 scales and one FP16 base. For the evaluated dimensions, the packed size is

$$M_T = (T-1)\frac{d}{8} + \frac{3d}{16} + T(G+2), \qquad G = \frac{d}{32}. \tag{24}$$

Relative to the $4d$-byte FP32 vector,

$$\mathrm{CR}_T = \frac{4d}{M_T}. \tag{25}$$

At $d = 768$, $T = 1, 2, 3$ require 170, 292, and 414 bytes ($18.07\times$, $10.52\times$, and $7.42\times$). The default $T = 3$ payload exceeds PQ-8x's 384 bytes by 30 bytes (7.8%), so comparisons using it are explicitly nearby-rate rather than equal-byte claims. An optional cached FP32 norm adds 4 bytes per vector; Table 19 in Appendix D reports the complete storage accounting, and Appendix C examines storage-parameter sensitivity.

RHR costs $O(d \log 128)$; phase assignment and residual updates cost $O(d)$ per application. Each sweep evaluates four shapes per pair and solves a real $T \times T$ system per block in $O(BT^2 + T^3)$ time. Here $B = 16$ and $T \leq 3$. More applications increase encoding work and storage but, because configurations are independently encoded and scales quantized, do not guarantee monotonically lower distortion.

4 Experiments

4.1 Experimental Setup

Datasets and embeddings. We evaluate the official BEIR test splits of FIQA (57,638 passages and 648 queries) and DBPedia-Entity (4,635,922 entities and 400 queries) against their complete

Table 1: Test-set retrieval accuracy on DBPedia-Entity with Qwen-7B embeddings (IP and cosine). Values are means over 50 random seeds. The 95% confidence intervals across seeds for NDCG@10, Recall@100, and MRR@10 have half-width at most 10^{-4}. Best and second-best values are marked among compressed methods, excluding FP32.

Method	**IP**			**Cosine**		
	NDCG @10	**Recall @100**	**MRR @10**	**NDCG @10**	**Recall @100**	**MRR @10**
Qwen-7B (3584d)						
FP32	0.47270	0.59365	0.77925	0.47217	0.59367	0.78162
PQ-8x (Jegou et al., 2010)	<u>0.47158</u>	0.59163	0.77373	0.47028	0.59217	0.77317
PQ-16x (Jegou et al., 2010)	0.46739	0.58700	0.77383	0.47088	0.58766	<u>0.77634</u>
OPQ-8x (Ge et al., 2013)	0.47033	0.59150	0.77394	0.47039	0.59146	0.77203
LOPQ-8x (Kalantidis & Avrithis, 2014)	0.47142	0.58996	0.77500	0.47121	0.59108	0.77275
RSLM4Lite (Lenhardt et al., 2026)	0.47025	<u>0.59242</u>	0.77607	0.47002	<u>0.59275</u>	0.77604
TurboQuant-4bit (Zandieh et al., 2026)	0.46931	0.59195	0.77223	<u>0.47135</u>	0.59191	0.77461
RaBitQ (Gao & Long, 2024)	0.46756	0.58661	0.77477	0.46756	0.58661	0.77477
exRaBitQ-2bit (Gao et al., 2025)	0.46836	0.58707	0.77219	0.46824	0.58707	0.77219
AC-Scoreq (Ours), $T = 1$	0.46902	0.58600	0.77307	0.46960	0.58767	0.77212
AC-Scoreq (Ours), $T = 2$	0.47101	0.59206	<u>0.77825</u>	0.47078	0.59206	0.77307
AC-Scoreq (Ours), $T = 3$	**0.47232**	**0.59299**	**0.77922**	**0.47136**	**0.59284**	**0.77806**

corpora, using only the official test qrels (Thakur et al., 2021; Hasibi et al., 2017). We use gte-modernbert-base (768d), gte-Qwen2-1.5B-instruct (1536d), and gte-Qwen2-7B-instruct (3584d) (Li et al., 2023; Warner et al., 2025; Yang et al., 2024); all methods share the embeddings within each dataset–model setting.

Baselines and Variants. Baselines are FP32, PQ-8x/16x (Jegou et al., 2010), OPQ-8x (Ge et al., 2013), LOPQ-8x (Kalantidis & Avrithis, 2014), RSLM4Lite (Lenhardt et al., 2026), TurboQuant-4bit (Zandieh et al., 2026), RaBitQ (Gao & Long, 2024), and exRaBitQ-2bit (Gao et al., 2025). We test AC-Scoreq with $T = 1, 2, 3$ and RHR; no-RHR results appear in the appendix.

Evaluation Metrics and Protocol. We report NDCG@10, Recall@100, and MRR@10 after full-corpus top-1000 retrieval using only the official test-query IDs; Recall@100 uses relevance labels rather than FP32 neighbors. Results average 50 random seeds, with 95% confidence-interval half-widths at most 10^{-4}. The encoder and fixed codebooks access neither queries nor qrels. We report all predefined operating points $T = 1, 2, 3$ with the other settings fixed across datasets and models. Table 1 gives the DBPedia-Entity Qwen-7B IP and cosine results; Appendix B reports the remainder, and Appendix E specifies timing.

4.2 Main Results

Table 1 gives the main DBPedia-Entity results; Appendix B contains complete accuracy results, and Table 19 in Appendix D details packed storage and the nearby-budget comparison. Table 2 reports measured Qwen-7B search time.

Retrieval fidelity across operating points. Increasing T generally improves retrieval fidelity by compensating more residual error. The $T = 3$ configuration remains close to the FP32 reference and competitive with the compressed baselines across the evaluated models and similarities, while $T = 1$ and $T = 2$ provide lower-cost alternatives. These accuracy results are interpreted jointly with each configuration's actual storage and measured latency. Complete metrics and storage-parameter sensitivity results appear in Appendices B and C.

End-to-end full-corpus search time. Table 2 compares full-corpus search over the same 400 official DBPedia-Entity test queries used for accuracy evaluation. Each baseline uses its official GPU kernel implementation. AC-Scoreq $T = 3$ takes 1.683 s for IP and 1.732 s for cosine, versus 2.362 s and 2.322 s for PQ-8x; all three AC configurations are faster than the timed compressed

baselines. These end-to-end times include scoring and top-k selection but exclude offline encoding; Appendix E gives the complete protocol.

Table 2: DBPedia-Entity full-corpus top-1000 retrieval time (seconds) with Qwen-7B embeddings on one NVIDIA A800; lower is better. Bold values mark AC-Scoreq at $T = 3$.

Method	**IP**	**Cosine**
FP32	59.272	59.894
PQ-8x (Jegou et al., 2010)	2.362	2.322
PQ-16x (Jegou et al., 2010)	1.810	1.882
RaBitQ (Gao & Long, 2024)	1.733	1.758
exRaBitQ-2bit (Gao et al., 2025)	2.298	2.396
OPQ-8x (Ge et al., 2013)	1.720	1.798
LOPQ-8x (Kalantidis & Avrithis, 2014)	1.803	1.871
RSLM4Lite (Lenhardt et al., 2026)	1.747	1.799
TurboQuant-4bit (Zandieh et al., 2026)	1.793	1.881
AC-Scoreq (Ours), $T = 1$	1.080	1.171
AC-Scoreq (Ours), $T = 2$	1.331	1.372
AC-Scoreq (Ours), $T = 3$	**1.683**	**1.732**

4.3 Ablation Study

Table 3: FIQA test-set inner-product NDCG@10 at $T = 3$ from paired reruns; best and second-best values among these three variants are bold and underlined. The controls retain the same additional coding budget: equal-amplitude coding disables shape refinement and keeps the initialized equal shape fixed, while extra-phase coding reallocates the shape bits to phase information.

Variant	**MB**	**Q1.5B**	**Q7B**
AC-Scoreq (Ours), $T = 3$	**.50295**	**.53947**	**.62828**
Equal-amplitude (no shape refinement)	.50063	.53634	.62582
Same-budget extra phase	<u>.50147</u>	<u>.53645</u>	<u>.62593</u>

Table 3 tests two controls at the same additional bit budget. Disabling shape refinement keeps the initialized equal-amplitude shape fixed and raises corpus MSE by approximately 31–37% across the three FIQA embeddings; reallocating the shape bits to phase gives lower retrieval scores than AC in all 27 FIQA model–similarity–metric cells. For Qwen-1.5B, the nine AC improvements over extra phase range from 0.00109 to 0.00524; in particular, the IP NDCG@10 values are 0.53947 and 0.53645. The consistent direction supports the tested allocation. Complete ablations appear in Appendix C.

5 Conclusion

AC-Scoreq introduces a compact phase–amplitude–scale representation for fixed-codebook residual quantization. Independent phase codes preserve coordinate-level direction, pairwise shapes add local relative-amplitude freedom, and jointly fitted block scales coordinate residual applications while retaining structured packed-code scoring. Varying T exposes discrete trade-offs among compression, scoring work, and retrieval quality: smaller T favors storage and latency, while additional applications compensate more residual error. Our evaluation reports exact storage and makes fair comparisons at nearby native budgets, with claims limited to the resulting storage–latency–quality trade-offs rather than strict equal-byte dominance. Experiments across multiple datasets and embedding dimensions show that these configurations maintain competitive retrieval quality under different compression budgets while reducing measured search latency. Overall, AC-Scoreq provides a practical fixed-codebook approach for balancing vector storage, retrieval efficiency, and ranking fidelity.

AI USE STATEMENT

In this work, we used generative AI tools to assist with the implementation of selected baselines for reproducing their experiments, specifically exRaBitQ-2bit, OPQ-8x, LOPQ-8x, RSLM4Lite, and TurboQuant. We did not use generative AI tools to develop the core ideas of this work or to implement its core experiments. Additionally, we used generative AI tools to assist with writing and polishing the manuscript, as well as literature retrieval and discovery, including finding related work. We take responsibility for reviewing and verifying all AI-assisted work and for the final content of this paper, including text, claims, code, and other artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Sections 3 and 4 specify the quantization method, datasets, embedding settings, and evaluation protocol. Appendix A provides the initialization details and complete encoding algorithm; Appendices B and C report the full retrieval results and ablations. Appendices D and E document storage accounting, implementation settings, and the retrieval timing protocol.

REFERENCES

Nir Ailon and Bernard Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM Journal on computing*, 39(1):302–322, 2009.

Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374, 2020.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pp. 380–388, 2002.

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. *Proceedings of the ACM on Management of Data*, 3(3): 1–26, 2025. doi: 10.1145/3725413.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2946–2953, 2013.

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, 2012.

Ruiqi Guo, Sanjiv Kumar, Krzysztof Choromanski, and David Simcha. Quantization based fast inner product search. In *Artificial intelligence and statistics*, pp. 482–490. PMLR, 2016.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *International Conference on Machine Learning*, pp. 3887–3896. PMLR, 2020.

Faegheh Hasibi, Fedor Nikolaev, Chenyan Xiong, Krisztian Balog, Svein Erik Bratsberg, Alexander Kotov, and Jamie Callan. DBpedia-Entity v2: A test collection for entity search. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pp. 1265–1268, 2017.

Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pp. 604–613, 1998.

Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE transactions on big data*, 7(3):535–547, 2019.

Yannis Kalantidis and Yannis Avrithis. Locally optimized product quantization for approximate nearest neighbor search. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2321–2328, 2014.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Rastislav Lenhardt, Teodora Dobos, Thomas Vecchiato, Jiri Isa, and Igor Ginzburg. RSLM: Training-free vector quantization for approximate nearest neighbor search. *arXiv preprint arXiv:2608.30384*, 2026. URL https://arxiv.org/abs/2608.30384.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.

Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE transactions on pattern analysis and machine intelligence*, 42(4):824–836, 2018.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, et al. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 2526–2547, 2025. doi: 10.18653/v1/2025.acl-long.127. URL https://aclanthology.org/2025.acl-long.127/.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

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 Additional Method Details

Final-application initialization. With equal initial AC shapes, the final application's block scale is

$$s_g^{(T,0)} = \frac{\sqrt{2}}{B} \sum_{j \in B_g} |r_j^{(T)}|. \tag{26}$$

The factor $\sqrt{2}$ offsets the equal shape's coefficient $1/\sqrt{2}$.

Scope of the pairwise geometric bound. For $s > 0$, Eq. (12) selects the zero shape if $\max_{a \neq (0,0)} p^\top a \leq s/2$. The angular excess-error bound stated in Section 3.2.3 assumes nearest phase codes and a pair-specific optimal nonnegative scale; it does not cover shared or quantized scales or retrieval rankings.

Phase projections. The four phase projections of a complex query coordinate $w = a + ib$ are

$$\begin{array}{c|cccc} c & 1 & i & -1 & -i \\ \hline \mathrm{Re}(w\bar{c}) & a & b & -a & -b \end{array}. \tag{27}$$

A.1 Encoding Algorithm

Algorithm 1 AC-Scoreq Encoding

Require: Database vector $x \in \mathbb{R}^d$; phase-code applications T; blocks $\{B_g\}$ with $B = 16$; fixed sets $\mathcal{C}$ and $\mathcal{A}$; refinement sweeps $K = 3$
Ensure: Packed phase codes, AC shapes, uint8 block scales, and FP16 scale bases

Apply the selected preprocessing and form complex coordinates z_j
if $T > 1$ **then**
    $\mathcal{M} \leftarrow \{\text{magnitude}, \text{projection}\}$
else
    $\mathcal{M} \leftarrow \{\text{AC initialization}\}$
end if
for each initialization $m \in \mathcal{M}$ **do**
    $r \leftarrow z$
    for $t = 1, \ldots, T-1$ **do**
        Select phase codes $c_j^{(t)} \in \mathcal{C}$ from r_j
        Initialize block scales $s_g^{(t)}$ using mode m
        $r_j \leftarrow r_j - s_{g(j)}^{(t)} c_j^{(t)}$
    end for
    Refine the ordinary applications for two preliminary sweeps and recompute their residual r
    Initialize $c_j^{(T)}$ from r_j and all AC shapes to $(1/\sqrt{2}, 1/\sqrt{2})$
    Initialize the scales $s_g^{(T)}$ using Eq. (26)
    for $k = 1, \ldots, K$ **do**
        Hold all codes fixed; jointly fit block scales using Eq. (15)
        Absorb negative scale signs into phases and quantize scales
        for $t = 1, \ldots, T-1$ **do**
            Hold the other applications and current scales fixed
            Reselect $c_j^{(t)}$ from the leave-one-out target in Eq. (17)
        end for
        Reselect the phases of application T and each pair's AC shape
    end for
    Jointly fit and quantize the block scales once more
    Compute $E_m = \sum_j |z_j - \hat{z}_j|^2$
end for
Select the candidate with minimum E_m and pack its codes and scales
return packed phase codes, AC shapes, uint8 scales, and FP16 bases

Table 4: Test-set NDCG@10 results on FIQA across three embedding models and three ranking functions; higher is better. Best and second-best values are marked among compressed methods, excluding FP32.

Method	**IP**			**Cosine**			**Negative L2**		
	MB	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**
FP32	.50297	.54002	.62912	.52243	.53960	.62906	.52625	.53974	.62882
PQ-8x	.50135	.53806	.62792	.51887	.53794	.62685	.51907	.53861	**.62853**
PQ-16x	.48410	.53361	.61977	**.52057**	.53732	.62409	**.51990**	.53781	.62573
OPQ-8x	.50239	.53523	.62753	.51309	.53572	.62628	.50107	.53543	.62719
LOPQ-8x	.50223	.53714	.62504	.51185	.53720	.62727	.50867	.53730	.62672
RSLM4Lite	.50135	.53666	.62660	.51648	.53694	.62655	.50732	.53694	.62599
TurboQuant-4bit	.49964	.53707	.62446	.51872	.53857	.62584	.50092	.53722	.62590
RaBitQ	.45316	.52541	.61864	.47338	.52541	.61904	.47451	.52541	.61895
exRaBitQ-2bit	.45371	.52395	.62120	.47574	.52395	.62052	.45784	.52395	.62063
AC-Scoreq w/o RHR, $T = 1$	.35847	.52404	.61517	.41894	.52814	.62312	.43430	.52837	.62352
AC-Scoreq w/o RHR, $T = 2$	.48441	.53499	.62410	.51105	.53337	.62413	.51061	.53294	.62360
AC-Scoreq w/o RHR, $T = 3$	.49609	.53794	.62655	.51930	.53821	.62692	.51685	.53794	.62592
AC-Scoreq (Ours), $T = 1$	.45852	.53053	.62186	.48929	.53000	.61935	.49149	.53057	.61861
AC-Scoreq (Ours), $T = 2$	.49966	.53574	.62592	.51611	.53593	.62605	.51843	.53617	.62470
AC-Scoreq (Ours), $T = 3$	**.50295**	**.53947**	**.62828**	.52028	**.53947**	**.62814**	.51697	**.53893**	.62824

B Additional Retrieval Metrics

This appendix reports the complete test-set results available for AC-Scoreq and the evaluated baselines. Tables are grouped by metric, following the main evaluation: NDCG@10, Recall@100, and MRR@10. Each table includes ModernBERT (MB), Qwen-1.5B (Q1.5B), and Qwen-7B (Q7B) under inner-product (IP), cosine, and negative L2 rankings. FP32 is the uncompressed reference. AC-Scoreq is reported both with RHR, as in the main results, and without RHR. All values are higher-is-better; negative L2 ranks by the negated distance.

B.1 Additional NDCG@10 Results

Tables 4 and 5 report NDCG@10 for FIQA and DBPedia-Entity, respectively, across the three embedding models and all three ranking functions.

B.2 Additional Recall@100 Results

Tables 6 and 7 report Recall@100 for FIQA and DBPedia-Entity, respectively, across the three embedding models and all three ranking functions.

B.3 Additional MRR@10 Results

Tables 8 and 9 report MRR@10 for FIQA and DBPedia-Entity, respectively, across the three embedding models and all three ranking functions.

C Additional Ablation Results

This section reports the complete available ablation results. Tables 11–13 contain FIQA accuracy under inner-product, cosine, and negative-L2 ranking. Tables 15–17 report the corresponding DBPedia-Entity subset; configurations not evaluated there are omitted. The RHR and no-RHR $T = 3$ rows reuse the complete two-dataset comparison. The default and same-budget extra-phase accuracy rows were rerun under the same protocol. The merged equal-amplitude control disables shape refinement, so the final-stage shapes remain at their equal-amplitude initialization while retaining the 2-bit shape field and the same storage layout. Extra phase instead reallocates those bits to phase information.

Table 5: Test-set NDCG@10 results on DBPedia-Entity across three embedding models and three ranking functions; higher is better. Best and second-best values are marked among compressed methods, excluding FP32.

Method	**IP**			**Cosine**			**Negative L2**		
	MB	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**
FP32	.41525	.46287	.47270	.42277	.46322	.47217	.42299	.46302	.47201
PQ-8x	.41251	.45714	.47158	**.42209**	.45944	.47028	**.42101**	.46085	.47159
PQ-16x	.40204	.45038	.46739	.41652	.45695	.47088	.41711	.45796	**.47190**
OPQ-8x	.41412	.46079	.47033	.41805	.45987	.47039	.41899	.45915	.47068
LOPQ-8x	.41416	.45977	.47142	.41798	.45814	.47121	.41924	.45940	.47063
RSLM4Lite	.41351	.46030	.47025	.41955	.46063	.47002	.41972	.46052	.46964
TurboQuant-4bit	.41445	.46098	.46931	.41953	.46121	.47135	.41966	.46012	.47061
RaBitQ	.38374	.44277	.46756	.38332	.44277	.46756	.38162	.44278	.46756
exRaBitQ-2bit	.38465	.44611	.46836	.38694	.44607	.46824	.38623	.44607	.46836
AC-Scoreq w/o RHR, $T = 1$	.33359	.44536	.46736	.36206	.45007	.46653	.36518	.45122	.46589
AC-Scoreq w/o RHR, $T = 2$	.40830	.45769	.47014	.41584	.45558	.46993	.41585	.45706	.47019
AC-Scoreq w/o RHR, $T = 3$	.41018	.45797	.47078	.41858	.45914	.47087	.41704	.45928	.47045
AC-Scoreq (Ours), $T = 1$	.39857	.45623	.46902	.41326	.45629	.46960	.41075	.45444	.46861
AC-Scoreq (Ours), $T = 2$	.41393	.45957	.47101	.41951	.45972	.47078	.41896	.45971	.47062
AC-Scoreq (Ours), $T = 3$	**.41454**	**.46267**	**.47232**	.42035	**.46290**	**.47136**	.41975	**.46289**	.47078

Table 6: Test-set Recall@100 results on FIQA across three embedding models and three ranking functions; higher is better. Best and second-best values are marked among compressed methods, excluding FP32.

Method	**IP**			**Cosine**			**Negative L2**		
	MB	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**
FP32	.83131	.85567	.91828	.84642	.85567	.91880	.84881	.85567	.91830
PQ-8x	.83018	.85153	.91738	**.84525**	**.85196**	.91738	**.84592**	.85440	.91708
PQ-16x	.82275	.84797	.91342	.84284	.85122	.91498	.84091	**.85531**	**.91828**
OPQ-8x	.82864	.85024	.91600	.83407	.85097	.91610	.83114	.85007	.91659
LOPQ-8x	.82334	.85044	.91677	.83364	.85049	.91513	.83876	.85007	.91669
RSLM4Lite	.82380	.85045	.91690	.83908	.85064	.91567	.83572	.85009	.91667
TurboQuant-4bit	.82224	.85033	.91695	.83504	.85049	.91615	.83434	.85061	.91685
RaBitQ	.78811	.84884	.91633	.80475	.84884	.91633	.80767	.84884	.91633
exRaBitQ-2bit	.78525	.84936	.91633	.80191	.84936	.91633	.78895	.84936	.91633
AC-Scoreq w/o RHR, $T = 1$	.66155	.84310	.91650	.73617	.84203	.91396	.75629	.83879	.91497
AC-Scoreq w/o RHR, $T = 2$	.82186	.84788	.91582	.83542	.84773	.91624	.83633	.85013	.91427
AC-Scoreq w/o RHR, $T = 3$	.82885	.84828	.91688	.84008	.84881	.91738	.84156	.85081	.91687
AC-Scoreq (Ours), $T = 1$	.81040	.85136	.91534	.82311	.85023	.91559	.82392	.84811	.91559
AC-Scoreq (Ours), $T = 2$	.82933	.85150	.91727	.83546	.85147	.91727	.83736	.85022	.91727
AC-Scoreq (Ours), $T = 3$	**.83061**	**.85160**	**.91790**	.84255	.85183	**.91790**	.84503	.85153	.91759

"No phase refinement" disables phase updates in the main alternating loop while retaining the common initialization procedure.

Interpretation of the ablations. For FIQA/ModernBERT, adaptive AC lowers corpus MSE from 6.320×10^{-2} with the merged equal-amplitude/no-shape-refinement control to 4.818×10^{-2}; the wider-angle set gives 5.412×10^{-2}. The wider-angle variant improves some retrieval cells but has higher MSE than the default across all three FIQA embeddings. One refinement sweep raises MSE for all three embeddings. Five sweeps slightly lower MSE for ModernBERT and Qwen-1.5B but raise it from 6.650×10^{-6} to 6.900×10^{-6} for Qwen-7B while increasing encoding work, motivating the three-sweep default. Blocks of 32 complex coordinates and 4-bit block scales each lower the $d = 768$ payload from 414 to 378 bytes. On FIQA/ModernBERT/IP, their NDCG@10 values are 0.50008 and 0.50102, compared with 0.50295 for the default and 0.50135 for PQ-8x at

Table 7: Test-set Recall@100 results on DBPedia-Entity across three embedding models and three ranking functions; higher is better. Best and second-best values are marked among compressed methods, excluding FP32.

Method	**IP**			**Cosine**			**Negative L2**		
	MB	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**
FP32	.52501	.56752	.59365	.51487	.56869	.59367	.51519	.56800	.59381
PQ-8x	.51968	.56530	.59163	.51395	.56637	.59217	.50881	**.56753**	.59284
PQ-16x	.49786	.55643	.58700	.51372	.56107	.58766	**.51121**	.56378	.59277
OPQ-8x	.51979	.56403	.59150	.51420	.56458	.59146	.50852	.56608	.59241
LOPQ-8x	.51878	.56445	.58996	.51315	.56702	.59108	.50663	.56501	.59153
RSLM4Lite	.52031	.56455	.59242	.51180	.56439	.59275	.50765	.56444	.59226
TurboQuant-4bit	.51877	.56548	.59195	.51404	.56656	.59191	.50931	.56620	.59167
RaBitQ	.47503	.55152	.58661	.46819	.55152	.58661	.46237	.55164	.58661
exRaBitQ-2bit	.47227	.55124	.58707	.46531	.55124	.58707	.47295	.55124	.58707
AC-Scoreq w/o RHR, $T = 1$	.33905	.55575	.59019	.38386	.56318	.59081	.39518	.55775	.59183
AC-Scoreq w/o RHR, $T = 2$	.49959	.56278	.59167	.50150	.56234	.59208	.49404	.56176	.59188
AC-Scoreq w/o RHR, $T = 3$	.51638	.56435	.59212	.50668	.56551	.59283	.50283	.56544	.59252
AC-Scoreq (Ours), $T = 1$	.48094	.56139	.58600	.48101	.55725	.58767	.47820	.56292	.58644
AC-Scoreq (Ours), $T = 2$	.51256	.56373	.59206	.51031	.56241	.59206	.50169	.56323	.59206
AC-Scoreq (Ours), $T = 3$	**.52068**	**.56637**	**.59299**	**.51432**	**.56703**	**.59284**	.50947	.56636	**.59296**

Table 8: Test-set MRR@10 results on FIQA across three embedding models and three ranking functions; higher is better. Best and second-best values are marked among compressed methods, excluding FP32.

Method	**IP**			**Cosine**			**Negative L2**		
	MB	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**
FP32	.59099	.62177	.69657	.60563	.62286	.69723	.61274	.62255	.69711
PQ-8x	**.58744**	.61994	.69503	.60171	.61936	.69536	.60160	.62092	.69478
PQ-16x	.56291	.61490	.69007	**.60539**	.62029	.69319	**.60585**	.61993	.69471
OPQ-8x	.57772	.61474	.69427	.59677	.61666	.69469	.58438	.61758	.69413
LOPQ-8x	.57822	.61882	.69212	.59573	.61936	.69468	.58389	.61011	.69426
RSLM4Lite	.57828	.61810	.69482	.59123	.61932	.69472	.58831	.61931	.69359
TurboQuant-4bit	.57498	.61811	.69079	.59391	.61878	.69372	.58248	.61795	.69456
RaBitQ	.53616	.61121	.68630	.55823	.61121	.68710	.55778	.61121	.68697
exRaBitQ-2bit	.53936	.61122	.68969	.55983	.61122	.68884	.54338	.61122	.68892
AC-Scoreq w/o RHR, $T = 1$	.43681	.60600	.68990	.51031	.61089	.68509	.52734	.60994	.69422
AC-Scoreq w/o RHR, $T = 2$	.56640	.61785	.69313	.59657	.61532	.69216	.59410	.61558	.69152
AC-Scoreq w/o RHR, $T = 3$	.57780	.61969	.69481	.59932	.61969	.69464	.59800	.61969	.69451
AC-Scoreq (Ours), $T = 1$	.53681	.61425	.68171	.56972	.61600	.69268	.57417	.61371	.68462
AC-Scoreq (Ours), $T = 2$	.58174	.61853	.69442	.60085	.61885	.69471	.59830	.61957	.69298
AC-Scoreq (Ours), $T = 3$	.58468	**.62158**	**.69538**	.60473	**.62077**	**.69596**	.59801	**.62143**	**.69685**

384 bytes. These controls probe storage sensitivity; retrieval metrics need not change monotonically with MSE.

D Implementation Details

D.1 Packed Storage Comparison

Table 19 reports the theoretical packed storage of the evaluated representations at $d = 768$ under the accounting used throughout the paper. Because AC-Scoreq's payload is determined by the discrete number of phase-code applications together with fixed-width phase, shape, and scale fields, it cannot in general realize an arbitrary exact integer compression ratio; we therefore compare methods at the

Table 9: Test-set MRR@10 results on DBPedia-Entity across three embedding models and three ranking functions; higher is better. Best and second-best values are marked among compressed methods, excluding FP32.

Method	**IP**			**Cosine**			**Negative L2**		
	MB	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**	**MB**	**Q1.5B**	**Q7B**
FP32	.74864	.79632	.77925	.75959	.79290	.78162	.75820	.79272	.78044
PQ-8x	.74530	.78186	.77373	.75723	.78721	.77317	**.75767**	.78898	.77589
PQ-16x	.74196	.78356	.77383	.75609	.78186	.77634	.75667	.78448	.77833
OPQ-8x	.74474	.78069	.77394	**.76145**	.78282	.77203	.74386	.78223	.77374
LOPQ-8x	.74626	.78558	.77500	.75616	.78422	.77275	.74661	.78573	.77375
RSLM4Lite	.74335	.78495	.77607	.75659	.78574	.77604	.73442	.78566	.77473
TurboQuant-4bit	.74737	.78683	.77223	.75682	.78576	.77461	.73300	.78320	.77474
RaBitQ	.71166	.77254	.77477	.72798	.77254	.77477	.73262	.77254	.77477
exRaBitQ-2bit	.71375	.77065	.77219	.72743	.77065	.77219	.72181	.77065	.77219
AC-Scoreq w/o RHR, $T = 1$	.67971	.77887	.77134	.70864	.77863	.77153	.70785	.77992	.77166
AC-Scoreq w/o RHR, $T = 2$	.74198	.78032	.77208	.74987	.78121	.77259	.74739	.78029	.77340
AC-Scoreq w/o RHR, $T = 3$	.74803	.78763	.77419	.75136	.78752	.77701	.74964	.78186	.77449
AC-Scoreq (Ours), $T = 1$	.73419	.77442	.77307	.74311	.78109	.77212	.74385	.78258	.77020
AC-Scoreq (Ours), $T = 2$	.74203	.78225	.77825	.75154	.78348	.77307	.75074	.78461	.77307
AC-Scoreq (Ours), $T = 3$	**.74824**	**.79048**	**.77922**	.75743	**.78950**	**.77806**	.75151	**.79084**	**.77904**

Table 10: FIQA ablation results at $T = 3$, reported as test-set inner-product NDCG@10; higher is better. "Same-budget extra phase" spends the AC shape bits on phase information instead of relative amplitude. The default and extra-phase accuracy rows are paired reruns under the same protocol.

Variant	**MB**	**Q1.5B**	**Q7B**
AC-Scoreq w/o RHR, $T = 3$	.49609	.53794	.62655
AC-Scoreq (Ours), $T = 3$	.50295	.53947	.62828
Equal-amplitude (no shape refinement)	.50063	.53634	.62582
Wide amplitude shapes	.50082	.53780	.62801
Same-budget extra phase	.50147	.53645	.62593
No phase refinement	.50134	.53944	.62580
One refinement sweep	.50167	.53836	.62687
Five refinement sweeps	.50118	.53818	.62685
Block size 8	.50026	.53744	.62766
Block size 32	.50008	.53818	.62739
Ridge coefficient $\lambda = 0$	.50253	.53890	.62758
Ridge coefficient $\lambda = 10^{-3}$	.50118	.53647	.62577
4-bit block scales	.50102	.53590	.62529

nearest native storage operating points and report the actual bytes per vector rather than claim strictly matched-rate superiority.

Nearby-budget interpretation. At $d = 768$, independently encoded AC-Scoreq configurations $T = 1, 2, 3$ require 170, 292, and 414 bytes per vector, whereas PQ-8x requires 384 bytes. We treat the default $T = 3$ comparison as a nearby-budget comparison because both methods operate in the same storage regime and their exact payloads are disclosed; we do not claim strict equal-byte superiority or attribute every small accuracy difference independently of the 30-byte gap. The three configurations instead expose distinct operating points: $T = 1$ prioritizes compression and measured latency, $T = 2$ provides an intermediate point, and $T = 3$ spends additional storage and computation for higher retrieval fidelity.

D.2 Experimental Implementation

We provide implementation details to facilitate reproducibility. Quantization is applied to fixed corpus embeddings without fine-tuning the embedding models. The database vectors are compressed,

Table 11: Complete $T = 3$ ablation results on FIQA: test-set NDCG@10; higher is better. Dashes denote configurations not rerun for this dataset.

Variant	IP			Cosine			Negative L2		
	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B
AC-Scoreq w/o RHR, $T = 3$	.49609	.53794	.62655	.51930	.53821	.62692	.51685	.53794	.62592
AC-Scoreq (Ours), $T = 3$	.50295	.53947	.62828	.52028	.53947	.62814	.51697	.53893	.62824
Equal-amplitude (no shape refinement)	.50063	.53634	.62582	.51463	.53752	.62705	.51624	.53622	.62703
Wide amplitude shapes	.50082	.53780	.62801	.51466	.53859	.62706	.51676	.53615	.62720
Same-budget extra phase	.50147	.53645	.62593	.51739	.53591	.62684	.51682	.53784	.62715
No phase refinement	.50134	.53944	.62580	.51639	.53647	.62614	.51666	.53880	.62710
One refinement sweep	.50167	.53836	.62687	.51806	.53824	.62693	.51600	.53673	.62723
Five refinement sweeps	.50118	.53818	.62685	.51722	.53945	.62745	.51741	.53913	.62814
Block size 8	.50026	.53744	.62766	.51779	.53688	.62744	.51077	.53749	.62756
Block size 32	.50008	.53818	.62739	.51845	.53900	.62801	.51092	.53879	.62821
Ridge coefficient $\lambda = 0$	.50253	.53890	.62758	.51558	.53900	.62710	.51505	.53891	.62738
Ridge coefficient $\lambda = 10^{-3}$	.50118	.53647	.62577	.51425	.53924	.62741	.51652	.53792	.62780
4-bit block scales	.50102	.53590	.62529	.51832	.53779	.62797	.51609	.53650	.62708

Table 12: Complete $T = 3$ ablation results on FIQA: test-set Recall@100; higher is better. Dashes denote configurations not rerun for this dataset.

Variant	IP			Cosine			Negative L2		
	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B
AC-Scoreq w/o RHR, $T = 3$	.82885	.84828	.91688	.84008	.84881	.91738	.84156	.85081	.91687
AC-Scoreq (Ours), $T = 3$	.83061	.85160	.91790	.84255	.85183	.91790	.84503	.85153	.91759
Equal-amplitude (no shape refinement)	.83021	.84067	.91690	.83989	.85044	.91690	.84127	.85089	.91715
Wide amplitude shapes	.82018	.84120	.91760	.83875	.84124	.91751	.84160	.85050	.91751
Same-budget extra phase	.82989	.84636	.91636	.84027	.84678	.91637	.84235	.84985	.91637
No phase refinement	.82933	.84927	.91627	.84008	.84773	.91627	.84156	.85081	.91627
One refinement sweep	.82918	.84978	.91758	.84085	.85035	.91758	.84427	.84980	.91758
Five refinement sweeps	.82881	.85148	.91727	.84344	.85158	.91727	.84232	.85112	.91727
Block size 8	.82288	.83246	.91786	.83998	.85046	.91670	.84477	.85046	.91670
Block size 32	.82186	.84937	.91703	.84076	.85092	.91728	.84347	.84969	.91751
Ridge coefficient $\lambda = 0$	.82932	.84917	.91746	.84034	.85081	.91746	.84108	.85112	.91746
Ridge coefficient $\lambda = 10^{-3}$	.82883	.85081	.91727	.83944	.85066	.91727	.84130	.85081	.91727
4-bit block scales	.82880	.85048	.91607	.84155	.85102	.91727	.84198	.84852	.91581

while query embeddings remain numerical and are transformed by RHR only when that preprocessing is enabled. The same corpus and query embeddings are used across methods within each dataset–model setting.

Hardware and evaluation protocol. The Qwen-7B DBPedia-Entity retrieval times in Table 2 were measured sequentially on one NVIDIA A800 GPU. Other experimental runs used both A800 and RTX 5090 servers; timings from different GPUs are not combined. Both retrieval quality and latency use only the official test queries (648 for FIQA and 400 for DBPedia-Entity), with the top-1000 candidates retrieved from the complete corpus. Retrieval time excludes embedding generation and offline database encoding.

AC-Scoreq configuration. We evaluate $T = 1, 2, 3$ total applications of the fixed phase codebook $\mathcal{C} = \{1, i, -1, -i\}$. For each T, the first $T-1$ applications use ordinary block-scaled complex phase codes. The final application uses the same phase codebook together with one of four fixed amplitude shapes for each pair of complex coordinates. Thus, amplitude coupling is part of application T, including when $T = 1$. Each block contains 16 complex coordinates, or 32 real dimensions. When enabled, RHR applies a normalized Hadamard transform in blocks of 128 coordinates, followed by a fixed permutation and sign flips shared by database and query vectors. The no-RHR ablation omits all three operations.

Table 13: Complete $T = 3$ ablation results on FIQA: test-set MRR@10; higher is better. Dashes denote configurations not rerun for this dataset.

Variant	IP			Cosine			Negative L2		
	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B
AC-Scoreq w/o RHR, $T = 3$	.57780	.61969	.69481	.59932	.61969	.69464	.59800	.61969	.69451
AC-Scoreq (Ours), $T = 3$	.58468	.62158	.69538	.60473	.62077	.69596	.59801	.62143	.69685
Equal-amplitude (no shape refinement)	.58415	.61984	.69441	.59779	.61854	.69319	.59781	.61828	.69515
Wide amplitude shapes	.58414	.62074	.69459	.59832	.62020	.68819	.59295	.62124	.69560
Same-budget extra phase	.58273	.62013	.69392	.60215	.61893	.69351	.59685	.61975	.69575
No phase refinement	.58260	.62100	.69382	.60013	.61986	.69321	.59605	.62055	.69569
One refinement sweep	.58113	.62086	.69464	.60305	.61972	.69306	.59723	.61892	.69591
Five refinement sweeps	.58302	.62115	.69605	.60420	.62181	.69651	.59946	.62153	.69699
Block size 8	.58061	.62007	.69474	.59841	.61997	.69492	.59315	.62017	.69675
Block size 32	.58174	.62081	.69422	.59738	.61989	.69206	.59288	.62058	.69506
Ridge coefficient $\lambda = 0$	.58335	.62033	.69474	.59964	.61899	.69584	.59718	.62105	.69402
Ridge coefficient $\lambda = 10^{-3}$	.58342	.62112	.69522	.59709	.61984	.69570	.59723	.62107	.69608
4-bit block scales	.58232	.61856	.69265	.60018	.61877	.69295	.59110	.61842	.69462

Table 14: Packed storage and corpus reconstruction MSE for the FIQA ablations. "B" denotes packed bytes per database vector.

Variant	ModernBERT			Qwen-1.5B			Qwen-7B		
	B	Ratio	MSE	B	Ratio	MSE	B	Ratio	MSE
AC-Scoreq w/o RHR, $T = 3$	414	7.42×	$5.127e-2$	822	7.47×	$2.163e-05$	1910	7.50×	$9.850e-06$
AC-Scoreq (Ours), $T = 3$	414	7.42×	$4.818e-2$	822	7.47×	$1.623e-05$	1910	7.50×	$6.650e-06$
Equal-amplitude (no shape refinement)	414	7.42×	$6.320e-2$	822	7.47×	$2.128e-05$	1910	7.50×	$9.100e-06$
Wide amplitude shapes	414	7.42×	$5.412e-2$	822	7.47×	$1.822e-05$	1910	7.50×	$7.790e-06$
No phase refinement	414	7.42×	$4.818e-2$	822	7.47×	$1.623e-05$	1910	7.50×	$6.940e-06$
One refinement sweep	414	7.42×	$4.990e-2$	822	7.47×	$1.681e-05$	1910	7.50×	$7.190e-06$
Five refinement sweeps	414	7.42×	$4.787e-2$	822	7.47×	$1.613e-05$	1910	7.50×	$6.900e-06$
Block size 8	486	6.32×	$3.536e-2$	966	6.36×	$1.181e-05$	2246	6.38×	$5.040e-06$
Block size 32	378	8.12×	$5.540e-2$	750	8.19×	$1.882e-05$	1742	8.22×	$8.040e-06$
Ridge coefficient $\lambda = 0$	414	7.42×	$4.816e-2$	822	7.47×	$1.622e-05$	1910	7.50×	$6.940e-06$
Ridge coefficient $\lambda = 10^{-3}$	414	7.42×	$4.839e-2$	822	7.47×	$1.630e-05$	1910	7.50×	$6.970e-06$
4-bit block scales	378	8.12×	$4.987e-2$	750	8.19×	$1.688e-05$	1742	8.22×	$7.250e-06$

For $T > 1$, the encoder tries mean-magnitude and projection-based initializations for the ordinary applications and retains the candidate with lower final reconstruction error. It performs two preliminary ordinary-code refinement sweeps, initializes the final amplitude-coupled application, and then performs three alternating refinement sweeps. Each sweep jointly fits the T real block scales with the discrete codes fixed, quantizes the scales, and reselects the ordinary phase codes followed by the final phase and shape codes. A final joint scale fit and quantization follow the third sweep. The normal equations use the ridge coefficient $\lambda = 10^{-4}$ in Eq. (15). No corpus-wide codebook training or query-derived statistics are used.

Storage and retrieval implementation. The theoretical packed sizes in Table 19 include 2-bit phase codes in ordinary applications, 6 bits per pair for the two final phase codes and amplitude shape, one uint8 scale per block and application, and one FP16 scale base per vector and application. Cached vector norms, identifiers, and file-format overhead are excluded from those packed sizes. The timed pipelines include scoring and top-k selection. All compared implementations use the same 5,000-vector batch size and top-k setting, and each baseline uses the official GPU kernel implementation provided for that method.

E Retrieval Latency Analysis

Table 2 reports a full-corpus scan over 4,635,922 DBPedia-Entity vectors for the 400 official Qwen-7B test-query embeddings, returning the top 1,000 results per query. Inner product and cosine were

Table 15: Complete $T = 3$ ablation results on DBPedia-Entity: test-set NDCG@10; higher is better. Dashes denote configurations not rerun for this dataset.

Variant	IP			Cosine			Negative L2		
	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B
AC-Scoreq w/o RHR, $T = 3$	.41018	.45797	.47078	.41858	.45914	.47087	.41704	.45928	.47045
AC-Scoreq (Ours), $T = 3$	.41454	.46267	.47232	.42035	.46290	.47136	.41975	.46289	.47078
Equal-amplitude (no shape refinement)	.41341	.45642	.47192	.42021	.45824	.46982	.42128	.45819	.47056
One refinement sweep	.41317	.45726	.47144	.42002	.45765	.47072	.41816	.45856	.47057
Block size 8	.41367	.45747	.46998	.41869	.45846	.47032	.41704	.45852	.46995
4-bit block scales	.41228	.45597	.47170	.41926	.45626	.46872	.41784	.45673	.46918

Table 16: Complete $T = 3$ ablation results on DBPedia-Entity: test-set Recall@100; higher is better. Dashes denote configurations not rerun for this dataset.

Variant	IP			Cosine			Negative L2		
	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B
AC-Scoreq w/o RHR, $T = 3$	.51638	.56435	.59212	.50668	.56551	.59283	.50283	.56544	.59252
AC-Scoreq (Ours), $T = 3$	.52068	.56637	.59299	.51432	.56703	.59284	.50947	.56636	.59296
Equal-amplitude (no shape refinement)	.51039	.56375	.58235	.51031	.56389	.59152	.50789	.56358	.59114
One refinement sweep	.51971	.56420	.59160	.51356	.56520	.59267	.50963	.56528	.59239
Block size 8	.52044	.56240	.59132	.51325	.56458	.59097	.50860	.56527	.59111
4-bit block scales	.51909	.56339	.59137	.51369	.56592	.59184	.50924	.56457	.59143

timed separately on the same A800; after multiple warm-up passes, each method was measured in multiple complete retrieval runs, and each table entry is their arithmetic mean. Encoding and training time are excluded. All methods use batches of 5,000 database vectors. The FP32 reference uses a corpus stored in FP16 and converted to FP32 for scoring. Each baseline uses its official GPU decoding and scoring kernel implementation.

Scope of the latency comparison. The reported mean times measure complete retrieval pipelines, including scoring and top-k selection; pipelines that decode compressed vectors also include decoding time. All rows use the same A800, full corpus, 400-query official test workload, 5,000-vector batch size, and top-1000 task. The baselines use their official GPU kernel implementations rather than a common generic routine. The comparison therefore evaluates practical end-to-end search costs under identical retrieval conditions.

Under both measured similarities, AC-Scoreq at $T = 1$ has the lowest latency among the timed compressed methods. Its $T = 2$ and $T = 3$ configurations are also faster than the timed PQ, OPQ, LOPQ, RaBitQ, exRaBitQ, RSLM4Lite, and TurboQuant implementations. Additional phase-code applications increase scoring work, so $T = 3$ is slower than $T = 1$ while providing higher retrieval accuracy in most evaluated settings. Because each baseline uses its official kernel implementation and all rows share the same retrieval conditions, the table compares complete implementations rather than imposing one common scoring routine on different compressed representations.

Table 17: Complete $T = 3$ ablation results on DBPedia-Entity: test-set MRR@10; higher is better. Dashes denote configurations not rerun for this dataset.

Variant	IP			Cosine			Negative L2		
	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B	MB	Q1.5B	Q7B
AC-Scoreq w/o RHR, $T = 3$	.74803	.78763	.77419	.75136	.78752	.77701	.74964	.78186	.77449
AC-Scoreq (Ours), $T = 3$	.74824	.79048	.77922	.75743	.78950	.77806	.75151	.79084	.77904
Equal-amplitude (no shape refinement)	.74082	.77230	.77781	.74606	.78039	.77812	.75086	.77306	.77805
One refinement sweep	.74808	.78031	.77469	.75619	.77857	.76985	.74940	.77925	.77879
Block size 8	.74313	.77954	.77773	.75729	.78279	.76829	.75051	.78240	.77825
4-bit block scales	.74417	.77691	.77816	.75480	.78145	.77470	.75007	.78185	.77830

Table 18: Packed storage and corpus reconstruction MSE for the DBPedia-Entity ablations. "B" denotes packed bytes per database vector.

Variant	ModernBERT			Qwen-1.5B			Qwen-7B		
	B	Ratio	MSE	B	Ratio	MSE	B	Ratio	MSE
AC-Scoreq (Ours), $T = 3$	414	7.42×	$4.592e-2$	822	7.47×	$1.617e-05$	1910	7.50×	$6.920e-06$
Equal-amplitude (no shape refinement)	414	7.42×	$6.026e-2$	822	7.47×	$2.121e-05$	1910	7.50×	$9.070e-06$
One refinement sweep	414	7.42×	$4.756e-2$	822	7.47×	$1.675e-05$	1910	7.50×	$7.170e-06$
Block size 8	486	6.32×	$3.349e-2$	966	6.36×	$1.177e-05$	2246	6.38×	$5.030e-06$
4-bit block scales	378	8.12×	$4.754e-2$	750	8.19×	$1.682e-05$	1742	8.22×	$7.230e-06$

Table 19: Theoretical packed per-vector storage under $d = 768$, excluding a separately cached database-vector norm. AC-Scoreq includes packed phase and amplitude codes, uint8 block scales, and one FP16 scale base per stage.

Type	Method	Bytes / vector	Compression Ratio
Full precision	FP32	3072	1.00×
Baselines	PQ-8x (Jegou et al., 2010)	384	8.00×
	PQ-16x (Jegou et al., 2010)	192	16.00×
	OPQ-8x (Ge et al., 2013)	384	8.00×
	LOPQ-8x (Kalantidis & Avrithis, 2014)	386	7.96×
	RSLM4Lite (Lenhardt et al., 2026)	384	8.00×
	TurboQuant-4bit (Zandieh et al., 2026)	514	5.98×
	ExRaBitQ-2bit (Gao et al., 2025)	196	15.67×
	RaBitQ (coarse-only) (Gao & Long, 2024)	104	29.54×
AC-Scoreq (Ours)	$T = 1$	170	18.07×
	$T = 2$	292	10.52×
	$T = 3$	414	7.42×