AC-FDE: TOWARDS ACCURATE AND COMPACT FIXED-DIMENSIONAL ENCODINGS FOR MULTI-VECTOR RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-Vector Retrieval (MVR) improves retrieval accuracy but incurs high computational cost due to MaxSim. Fixed-Dimensional Encoding (FDE) enables single-vector search by approximating MaxSim through repeated random bucketing and aggregation. Increasing the number of repetitions can improve retrieval accuracy, but produces high-dimensional vectors that reduce efficiency. We propose AC-FDE to improve this efficiency–accuracy trade-off through query-independent repetition selection and sampled blockwise PCA. A distortion bound guides repetition selection, while blockwise PCA compresses the resulting vectors with projections learned from a random document subset. Our analysis characterizes stable rank, blockwise reconstruction error, and sampled-subspace stability under stated assumptions. AC-FDE-hybrid further combines compressed FDE vectors with high-dimensional single-vector embeddings for complementary retrieval signals. Experiments across four retrieval benchmarks demonstrate improved accuracy–efficiency trade-offs, with small additional offline costs relative to MUVERA.

1 INTRODUCTION

Single-Vector Retrieval (SVR) encodes queries and documents into one embedding. Despite its efficiency, it forces complex semantics into a fixed point, losing nuance. Multi-Vector Retrieval (MVR, e.g., ColBERT [25]) improves accuracy by representing each token as a separate vector. For a query with m vectors and a document with n vectors, MVR computes a **MaxSim score**: for each query vector, take its maximum similarity with any document vector, then sum over all query vectors. This captures fine-grained token matches and outperforms SVR on key benchmarks, but incurring high computational overhead. To reduce MVR's costly pairwise comparisons and storage, recent methods use indexing [3; 39; 37; 15] or token pruning (SLIM [29], CITADEL [28]), but MaxSim computation persists.

MUVERA [19] instead transforms multi-vector representations into single vectors: it randomly partitions the vector space into buckets, aggregates vectors within each bucket (e.g., by summation), and concatenates the results. This process repeats over multiple independent **repetitions**, forming a fixed-dimensional embedding (FDE) vector whose similarity asymptotically approximates MaxSim. Alternative solutions like LEMUR [22] and ConstBert [31] incur higher online overhead and lack MUVERA's theoretical guarantees. However, MUVERA faces an **efficiency-accuracy dilemma**: as Figure 1 shows, the bucket aggregation systematically underestimates MaxSim, so closing the recall gap between MUVERA and vanilla MVR requires many repetitions. Yet many repetitions yield extremely high-dimensional FDE vectors, ultimately exceeding MVR's cost. Conversely, low-dimensional FDE vectors leave a substantial recall gap relative to MVR.

We propose AC-FDE to improve MUVERA's efficiency–accuracy trade-off. First, we introduce a **repetition selection mechanism**: a query-independent score bounds the expected gap between a bucket-local MaxSim proxy and the FDE score, guiding efficient offline repetition selection. Second, our stable-rank analysis motivates **blockwise PCA**, which minimizes reconstruction error among blockwise projections at lower cost than full-matrix PCA. Third, we fit PCA on a **small random document subset**, which only slightly deviates from full-corpus principal subspaces even under small sampling rate. The learned projections are reused for efficient online query compression. Finally,

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.

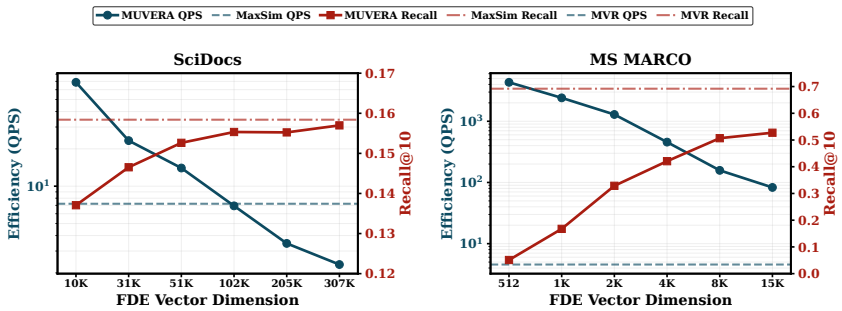


Figure 1: Recall and query speed (QPS) of MUVERA on SciDocs and MS MARCO [9] across varying dimension. With FDE vectors [33], increasing their dimensionalities by adding more repetitions improves recall but reduces retrieval efficiency.

AC-FDE-hybrid fuses compressed FDE vectors with high-dimensional single-vector embeddings to combine complementary retrieval signals.

Extensive experiments across multiple dense retrieval benchmarks demonstrate that AC-FDE effectively **resolves the efficiency-accuracy trade-off** inherent in MUVERA. First, AC-FDE achieves up to a *4.3*× speedup over baseline methods, accelerating MVR while maintaining equivalent recall levels. Second, at comparable dimensionalities, query processing with AC-FDE-hybrid is up to *two orders of magnitude* faster than SVR and vector fusion strategies at matched recall. Third, relative to MUVERA, AC-FDE introduces negligible overhead during the offline transformation of multi-vector representations into single-vector formats.

Contribution We introduce AC-FDE, a document-adaptive approach to fixed-dimensional encoding for multi-vector retrieval. Our main contribution is query-independent repetition selection using a document-side occupancy score motivated by a query-independent distortion bound. We combine this selection mechanism with sampled, repetition-wise compression to obtain compact representations without hurting online query efficiency, and extensively evaluate the resulting accuracy–efficiency trade-offs.

2 Preliminaries

2.1 Multi-vector Retrieval

Although Single-Vector Retrieval (SVR) is the predominant dense retrieval method, recent studies [30; 25; 19; 39] demonstrate that Multi-Vector Retrieval (MVR) yields superior performance under comparable embedding dimensions. Unlike SVR, which encodes each query Q and document P into a single vector, MVR represents them as variable-sized sets $\{q_i | i = 1, 2 \dots\}$ and $\{p_j | j = 1, 2, \dots\}$, respectively. The query-document similarity is then quantified via the following MaxSim score:

$$\text{MaxSim}(Q, P) = \sum_i \max_j \text{SIM}(q_i, p_j), \tag{1}$$

Throughout this paper, we follow Colbert [25], the pioneering MVR work, by normalizing each q_i and p_j and configuring the similarity score $\text{SIM}(\cdot, \cdot)$ as cosine similarity. As discussed in §1, while MaxSim delivers better performance for multi-vector retrieval, it also incurs significantly higher overhead than SVR.

2.2 MUVERA: Fixed-Dimensional Encoding for MVR

The **Fixed-Dimensional Encoding (FDE)** [19] maps a variable-sized set $X \subset \mathbb{R}^d$ to a fixed-dimensional encoding (FDE) vector $F(X) \in \mathbb{R}^{d_{\text{FDE}}}$ such that inner products approximate the MaxSim score:

$$\langle F(Q), F(P) \rangle \approx \text{MaxSim}(Q, P).$$

The encoding is constructed from R independent repetitions. For each repetition i:

1. **Random Partitioning:** A function $\phi_i : \mathbb{R}^d \to [B]$ (e.g., SimHash) maps vectors to $B = 2^k$ buckets based on k-bit signatures induced by ϕ_i.
2. **In-Bucket Aggregation:** Vectors from Q and P falling into bucket b are aggregated:
   - For queries: $\tilde{q}_{i,b}(Q) = \sum_{q \in Q : \phi_i(q) = b} q$,
   - For documents: $\tilde{p}_{i,b}(P)$ is the average of vectors in the bucket. When a document has no vectors in a bucket, a vector from the nearest non-empty bucket (nearest in terms of the L_0 distance between k-bit bucket signatures) is used as a surrogate (which is referred to as "surrogate filling"), ensuring that every bucket contributes a representative vector.
3. **Projection & Concatenation:** A JL random projection ψ_i reduces each aggregate to dimension d_{proj}. The projected vectors are concatenated across buckets:

$$F_i(Q) = \big(\psi_i(\tilde{q}_{i,1})^\top \,\|\, \cdots \,\|\, \psi_i(\tilde{q}_{i,B})^\top\big), \quad F_i(P) = \big(\psi_i(\tilde{p}_{i,1})^\top \,\|\, \cdots \,\|\, \psi_i(\tilde{p}_{i,B})^\top\big).$$

The final representation is the concatenation of all repetitions: $F(Q) = (F_1(Q) \,\|\, \cdots \,\|\, F_R(Q))$ (similarly for P), yielding dimension $d_{\text{FDE}} = R \cdot B \cdot d_{\text{proj}}$. The MaxSim approximation is then

$$\text{MaxSim}(Q, P) \approx \langle F(Q), F(P) \rangle = \frac{1}{R} \sum_{i=1}^{R} \langle F_i(Q), F_i(P) \rangle.$$

Matrix Notation. For a corpus of N documents, we define the *document FDE matrix* $\mathscr{P} \in \mathbb{R}^{N \times d_{\text{FDE}}}$ by stacking all document vectors $F(P_j)^\top$ as rows:

$$\mathscr{P} = \big[F(P_1)^\top \,\|\, \cdots \,\|\, F(P_N)^\top\big]^\top.$$

Similarly, for each repetition i, the *single-repetition sub-matrix* $\mathscr{P}^{(i)} \in \mathbb{R}^{N \times B \cdot d_{\text{proj}}}$ is formed from the $F_i(P_j)$ rows. These matrices facilitate compact notation in our subsequent derivations. Under mild conditions on k, d_{proj}, and R, the gap between $\langle F(Q), F(P) \rangle$ and $\text{MaxSim}(Q, P)$, converges to zero with high probability [19] (see Theorem A.1 in Appendix A), which relies on the following critical property:

Theorem 2.1 (Within-Bucket Near-Collinearity (from MUVERA [19], Claim A.4)). *With the same k as Theorem A.1. For any repetition i and any bucket b, with high probability, for any two document vectors p, p' assigned to bucket b, we have $\langle p, p' \rangle \geq 1 - O(\epsilon)$.*

The proof of Theorem 2.1 is provided in [19], and more details of MUVERA are provided in Appendix A.

3 Method Overview

Algorithm 1: Offline Construction of Compressed Document FDE Matrix

Input : Document set: $\mathcal{P}$, number of candidate repetitions: $R^{(\text{cand})}$, number of repetitions to retain: R (where $R < R^{(\text{cand})}$), number of buckets B, number of dimensions per repetition after compression: E

Output : Compressed matrix $\mathscr{U} \in \mathbb{R}^{|\mathcal{P}| \times RE}$, index $\mathcal{I}$, projection $\mathscr{V}$, functions $\{(\phi_i, \psi_i)\}_{i=1}^{R}$

Stage 1: Sample a subset $\widehat{\mathcal{P}} \subset \mathcal{P}$;

Stage 2: Select R repetitions on $\widehat{\mathcal{P}}$ with the lowest repetition-specific distortion scores (Equation equation 3);

Stage 3: Generate FDE matrices $\mathscr{P}$, $\widehat{\mathscr{P}}$ for $\mathcal{P}, \widehat{\mathcal{P}}$ using Equation equation 6 with the selected repetitions; denote their functions $\{(\phi_i, \psi_i)\}_{i=1}^{R}$;

Stage 4: Apply blockwise PCA to obtain compressed matrix $\mathscr{U}$ and eigenvector matrix $\mathscr{V}$;

Stage 5: Build index $\mathcal{I}$ on $\mathscr{U}$;

return $\mathcal{I}, \mathscr{U}, \mathscr{V}, \{(\phi_i, \psi_i)\}_{i=1}^{R}$

AC-FDE consists of offline construction and online retrieval. Offline, it augments MUVERA with three components: (1) **Adaptive Repetition Selection** retains R candidates using a query-independent distortion score; (2) **Blockwise PCA** compresses each repetition independently; and (3) **Sampling** reduces the cost of selection and PCA fitting using a random document subset.

Algorithm 1 builds an index over compressed document vectors and caches the selected encoding functions and PCA projections. AC-FDE-hybrid (§7) additionally fuses compressed FDE vectors with single-vector embeddings for complementary retrieval signals.

4 ADAPTIVE REPETITION SELECTION

We propose to select high-quality repetitions using a repetition-wise proxy for the ground-truth MaxSim score (§4.1). Comparing this proxy against the FDE score yields a query-independent distortion score (§4.2), enabling efficient offline selection before online query evaluation.

4.1 A REPETITION-WISE PROXY SCORE

We define the *Repetition-wise MaxSim Score*, $\overline{\mathrm{MaxSim}}_i$, using bucket-local maxima of token similarities under the same JL projection ψ_i as FDE. Unlike FDE, this proxy retains individual document token vectors rather than averaging them. For $b = \phi_i(q)$, let

$$m_i(q, P) = \begin{cases} \max_{p \in P : \phi_i(p) = b} \langle \psi_i(q), \psi_i(p) \rangle, & n_{i,b} > 0, \\ \max_{p \in P : \phi_i(p) = b'} \langle \psi_i(q), \psi_i(p) \rangle, & n_{i,b} = 0, \end{cases} \tag{2}$$

where $b' \in \operatorname{argmin}_{b'' :\, n_{i,b''} > 0} \| \operatorname{Bit}(b) - \operatorname{Bit}(b'') \|_0$, using the same tie-breaking rule as FDE. Finally, the repetition-wise MaxSim score is then $\overline{\mathrm{MaxSim}}_i(Q, P) := \sum_{q \in Q} m_i(q, P)$. Theorem 4.1 shows that the normalized gap between the proxy and the original MaxSim score can be made arbitrarily small by increasing the hash resolution and projection dimension.

Theorem 4.1. *Fix nonempty sets of unit vectors Q, P and a repetition i using SimHash and a Rademacher JL projection. For $\epsilon, \delta \in (0, 1)$, if $k \geq \frac{C}{\epsilon^2} \log \frac{4|P|}{\delta}$,$d_{\mathrm{proj}} \geq \frac{C}{\epsilon^2} \log \frac{8|Q||P|}{\delta}$, for a sufficiently large universal constant C, then, with probability at least $1 - \delta$,*

$$\frac{\left| \mathrm{MaxSim}(Q, P) - \overline{\mathrm{MaxSim}}_i(Q, P) \right|}{|Q|} \leq \epsilon.$$

The proof is deferred to Appendix B.

4.2 QUERY-INDEPENDENT DISTORTION BOUND

To rank repetitions, we analyze the gap between $\overline{\mathrm{MaxSim}}_i$ and the similarity of FDE vectors at the i_{th} repetition, i.e., $\langle F_i(Q), F_i(P) \rangle / |Q|$, which is denoted by $\mathrm{FDE}_i(Q, P)$. Theorem 4.2 provides an upper bound that is *query-independent*, depending only on document-side bucket occupancy.

Theorem 4.2. *Fix nonempty sets of unit vectors Q, P and a repetition i, with uniformly selected surrogates for empty buckets. Let $d_i^{\max}(P) = \max\{1, \max_b n_{i,b}\}$, for $\epsilon_{\mathrm{JL}}, \delta \in (0, 1)$, if $d_{\mathrm{proj}} \geq C \epsilon_{\mathrm{JL}}^{-2} \log \frac{2(|Q|+|P|)}{\delta}$, for a sufficiently large universal constant C, then, with probability at least $1 - \delta$ over ψ_i,*

$$0 \leq \frac{\overline{\mathrm{MaxSim}}_i(Q, P)}{|Q|} - \mathbb{E}_{\mathrm{surr}}[\mathrm{FDE}_i(Q, P)] \leq 2(1 + \epsilon_{\mathrm{JL}}) \left(1 - \frac{1}{d_i^{\max}(P)}\right).$$

The expectation is over surrogate selection, conditional on the hash and projection functions.

The proof of this theorem is given in Appendix D. Averaging this bound over a document set $\mathcal{P}$ yields the *repetition-specific distortion score*:

$$\mathrm{Score}_i := \frac{2}{|\mathcal{P}|} \sum\nolimits_{P \in \mathcal{P}} \left(1 - \frac{1}{d_i^{\max}(P)}\right). \tag{3}$$

Repetitions with lower Score_i are of higher quality. We generate a large candidate pool of $R^{(\mathrm{cand})}$ repetitions and retain the top-R with minimal scores. We observe that setting $R^{(\mathrm{cand})} = 10 \times R$ is sufficient to increase retrieval accuracy while incurring negligible overhead. The selection process and its computational overhead are detailed in Appendix C, where we also intuitively justify this score with an example.

Remark 1. *The score in equation 3 is intentionally query-independent, since repetition selection is performed offline before any query is observed. It is not a heuristic, but an average of the provable upper bound in Theorem 4.2. The primary use of this score is to* rank *repetitions. We also empirically demonstrate its positive correlation with the MaxSim approximation error in Appendix M.3. And we empirically demonstrate this error is positively correlated with the actual MaxSim error in Appendix M.4.*

5 DIMENSIONALITY REDUCTION FOR THE FDE MATRIX

In this section, we reveal the low stable rank property of the FDE matrix (§5.1), motivating efficient blockwise PCA (§5.2), and further speed up the process via document subsampling with theoretical guarantees (§5.3).

5.1 LOW STABLE RANK AND DIMENSIONALITY REDUCTION POTENTIAL OF $\mathscr{P}$

Theorem 5.1. *Consider the document FDE matrix $\mathscr{P}_0 \in \mathbb{R}^{N \times RBd_{\text{proj}}}$ with empty buckets filled with zeros rather than surrogates. Suppose that, within each bucket block, any two nonzero projected rows have cosine similarity at least $1-\epsilon$, where $0 \le \epsilon < 1$. Then*

$$\operatorname{srank}(\mathscr{P}_0) \le \sum_{i=1}^{R}\sum_{b=1}^{B} \operatorname{srank}(\mathscr{P}_0^{(i,b)}) \le \frac{RB}{1-\epsilon},$$

where zero blocks are omitted.

Here $\operatorname{srank}(\cdot) = \|\cdot\|_F^2/\|\cdot\|_2^2$ denotes stable rank. Appendix E further extends the bound to surrogate-filled FDE matrices. When $(1-\epsilon)d_{\text{proj}} \gg 1$, the bound is much smaller than $d_{\text{FDE}} = RBd_{\text{proj}}$, motivating dimensionality reduction. Although low stable rank alone does not guarantee that a few principal components capture most of the matrix's energy, this is empirically validated in §K.2.

5.2 BLOCKWISE PCA FOR FDE

Directly performing PCA on the FDE matrix $\mathscr{P}$ for dimensionality reduction is time-consuming, as it incurs a prohibitive cost of $O(d_{\text{FDE}}^3)$, where d_{FDE} is large. Since $\mathscr{P}$ is constructed by concatenating sub-matrices across all repetitions (see §2.2), a plausible alternative is to perform PCA locally at each repetition, i.e., blockwise PCA, and then concatenate the resulting eigenvectors across repetitions to form the full eigenvectors for $\mathscr{P}$.

However, blockwise PCA is not generally equivalent to full-matrix PCA because the former ignores the cross-block covariance captured by the latter [1; 14]. We provide a concrete counterexample in Appendix F.1. The following theorem suggests that performing blockwise PCA can still effectively achieve small reconstruction errors.

Theorem 5.2. *For each repetition i, let $\mathscr{V}^{(i)} \in \mathbb{R}^{(Bd_{\text{proj}}) \times E}$ contain the top-E eigenvectors of $\mathscr{P}^{(i)\top}\mathscr{P}^{(i)}$, and let $\mathscr{U}^{(i)} = \mathscr{P}^{(i)}\mathscr{V}^{(i)}$ and $\mathscr{U} = [\mathscr{U}^{(1)} \mid \cdots \mid \mathscr{U}^{(R)}]$. Then*

$$\left\| \mathscr{P} - \mathscr{U} \operatorname{diag}\big(\mathscr{V}^{(1)\top}, \ldots, \mathscr{V}^{(R)\top}\big) \right\|_F^2 = \sum_{i=1}^{R} \sum_{\ell > E} \sigma_\ell\big(\mathscr{P}^{(i)}\big)^2,$$

where $\sigma_\ell(\cdot)$ denotes the ℓth largest singular value. This reconstruction error is minimal among all blockwise orthogonal projections retaining E dimensions per repetition.

The proof of this theorem is provided in Appendix F.3. Based on the above theorem, $\mathscr{P}$ is approximated by the block-structured decomposition:

$$\mathscr{P} \approx \mathscr{U} \ \operatorname{diag}\Big(\mathscr{V}^{(1)\top}, \ldots, \mathscr{V}^{(R)\top}\Big), \qquad \text{where } \mathscr{U} = \begin{bmatrix}\mathscr{U}^{(1)} & \cdots & \mathscr{U}^{(R)}\end{bmatrix}.$$

The projection matrices $\{\mathscr{V}^{(i)}\}_{i=1}^{R}$ are reused for online query compression (§6). The algorithmic procedure and detailed complexity analysis are deferred to Appendix F.2.

Remark 2. *Although performing PCA separately on each repetition may not yield the same result as that by performing PCA on the full FDE matrix, it substantially reduces computation while preserving accuracy when the per-repetition spectra decay rapidly. Our experiments in §K.2 confirm that this error is small in practice and close to that of full-matrix PCA at the same total dimension.*

5.3 SAMPLING FOR BLOCKWISE PCA

Recall that the full document FDE matrix $\mathcal{P}$ is an $N \times d_{\text{FDE}}$ matrix, where N denotes the number of documents in the corpus and can be large. As analyzed in Appendix 5.3, even performing blockwise PCA on this matrix—in particular, deriving the projection matrices for each block—incurs

an overhead that is linear in the document count. To further reduce the cost of blockwise PCA for huge corpora, we therefore perform it on a sampled document subset $\widehat{\mathcal{P}}$ rather than on the full matrix $\mathcal{P}$, yielding projection matrices $\widehat{\mathcal{V}}^{(i)}$ for repetition i. The full FDE matrix $\mathcal{P}$ is then projected onto the span of $\widehat{\mathcal{V}}^{(i)}$. Theorem 5.3 bounds the deviation between the sampled and full-corpus principal subspaces. This guarantee concerns the stability of the learned compression subspaces.

Theorem 5.3. *Given a fixed corpus and candidate pool of size $R^{(\mathrm{cand})}$, draw N_s documents uniformly with replacement for both repetition selection and PCA. For each candidate i, let $\gamma_i > 0$, L_i, and ν_i^2 denote its full-corpus top-E eigengap, maximum FDE-vector norm, and outer-product variance, respectively (Appendix G.1). Set*

$$t = \log \frac{2R^{(\mathrm{cand})} B d_{\mathrm{proj}}}{\delta}, \qquad \epsilon_{\mathrm{samp}} = \max_{i \le R^{(\mathrm{cand})}} \frac{1}{\gamma_i} \left(\sqrt{\frac{2\nu_i^2 t}{N_s}} + \frac{2L_i^2 t}{3N_s} \right).$$

Given a sufficiently large N_s, with probability at least $1-\delta$, the largest principal angle between the sampled and full-corpus top-E PCA subspaces is $O(\epsilon_{\mathrm{samp}})$ simultaneously for all selected repetitions.

The proof of this theorem and the empirical validation of the assumption used in this theorem are given in Appendix G.2. Intuitively, the theorem shows that sufficiently many sampled documents recover principal subspaces close to those obtained from the full corpus when the covariance estimation error is small relative to the eigengap. This supports sampling as an approximation to full-corpus blockwise PCA. The effect of sampling on retrieval accuracy is assessed separately in §8.2 and Appendix M.5.

6 Online Query Processing with AC-FDE

Algorithm 2: Query processing algorithm

Input : a projection matrix: $\mathscr{V}$, a compressed document FDE matrix: $\mathscr{U}$, an index constructed on $\mathscr{U}$: $\mathcal{I}$, a sequence of R repetitions: $\{(\phi_i, \psi_i)\}_{i=1}^{R}$, a multi-vector query: Q.

Output : Retrieval results of executing Q

1 Transform Q into $F(Q)$ with the repetitions, $\{(\phi_i, \psi_i)\}_{i=1}^{R}$;

2 Project $F(Q)$ to $\mathscr{U}^{(Q)}$ using $\mathscr{V}$ with equation 4;

3 **return** the retrieval results of querying with $\mathscr{U}^{(Q)}$ on $\mathcal{I}$.

Algorithm 2 outlines online query execution. The multi-vector query Q is first converted into an FDE vector $F(Q)$ using the R repetitions $\{(\phi_i, \psi_i)\}_{i=1}^{R}$ selected offline. The cached projection matrix

$$\mathscr{V} = \mathrm{diag}\big(\mathscr{V}^{(1)}, \ldots, \mathscr{V}^{(R)}\big),$$

computed in Stage 4 of Algorithm 1, then compresses the query:

$$\mathscr{U}^{(Q)} = F(Q)\mathscr{V} = [F_1(Q)\mathscr{V}^{(1)} \,\|\, \cdots \,\|\, F_R(Q)\mathscr{V}^{(R)}]. \tag{4}$$

The resulting RE-dimensional vector is used to query the index $\mathcal{I}$ built over the compressed document matrix $\mathscr{U}$. *All projection matrices and FDE encoding functions are reused from offline construction; no PCA fitting or repetition selection is performed online.*

7 Implementing AC-FDE in Practice

Default Configuration AC-FDE uses a default sampling rate of 0.1 for repetition selection and blockwise PCA. HNSW indexes are built on the resulting single-vector representations. These choices keep the offline overhead small while preserving retrieval accuracy.

Optional Fusion with Standalone SVR AC-FDE is designed to accelerate MVR, but many deployments already have a strong single-vector retriever. Rather than replacing it, AC-FDE can be fused with it. A standalone SVR embedding captures global semantic similarity, while the AC-FDE vector preserves token-level MaxSim signals that a single global embedding often misses. This

requires only one index and one ANN query, unlike separate retrieval followed by rank fusion, which needs two searches and score calibration. Prior work has shown that compressed concatenation of embeddings can improve retrieval [35], and learned weighted concatenation is also established [4]. We therefore use direct concatenation by default and consider learned per-dimension weights as a variant. We outline the full procedure in Algorithm 5 in Appendix I. This fusing strategy is referred to as *AC-FDE-hybrid* in what follows.

8 EXPERIMENTS

8.1 EXPERIMENTAL SETUP

We evaluate on four BEIR benchmark datasets: MS MARCO [5], SciDocs [9], FiQA [32], and NQ [27]. Detailed description of these datasets is included in Appendix J.1.

Configurations of AC-FDE and AC-FDE-hybrid AC-FDE-hybrid fuses the FDE vector with a 1024-dimensional SVR vector from Qwen3-0.6B [44]. We truncate this SVR embedding to 768 dims and concatenate a 256-dimensional FDE vector from AC-FDE. This truncation is viable because Qwen3 employs Matryoshka Representation Learning [26], allowing embedding prefixes to retain semantic fidelity. We also consider another variant of AC-FDE-hybrid (referred to as AC-FDE-hybrid-2), which fuses 256-dimensional AC-FDE vectors and the raw SVR vectors without truncations, yielding 1280-dimensional fused vectors. HNSW indexes [33] are built on the fused vectors. Detailed setup is in Appendix J.3.

Baseline We compare AC-FDE against two types of baselines for accelerating MVR. First, we compare against **MUVERA** [19] and **LEMUR** [22], which transforms multi-vector representations into single vectors. Throughout the experiments, we follow [19] to quantize the FDE vectors produced by MUVERA. To ensure fair comparison against AC-FDE, we also perform PCA on the full document FDE matrix produced by MUVERA (referred to as **MUVERA-PCA**), which performs dimensionality reduction on the full query FDE vectors online. The other type of baselines accelerates MVR while keeping the multi-vector representations, including **CITADEL** [28], **SLIM** [29], **WARP** [40], and multi-vector indexing methods (**GEM** [43], **IGP** [3], **PLAID** [39], **DESSERT** [15], **EMVB** [37]). Other baselines such as ConstBERT [31] are excluded due to code unavailability. All methods employ the pre-trained ColBERT [25] model to produce 128-dimensional multi-vectors. Details are included in Appendix J.2.

In addition, we compare AC-FDE-hybrid against the strategy of fusing the same 768-dim SVR embedding with a 256-dimensional FDE vector produced by MUVERA, LEMUR, or random noise. We also fuse the same 768-dim SVR embedding and another SVR embedding (denoted by SVR-2) produced by the pre-trained Bert model [12] as another baseline. This experiment is repeated under two broadly adopted fusion strategies, including direct concatenation (denoted by *Fusing-1*) and learned weighted concatenation [4] (denoted by *Fusing-2*). For Fusing-2, we follow [4] to learn the weights by maximizing similarity for relevant query-document pairs while minimizing it for negatives (Details are included in Appendix J.3).

Unless explicitly stated, we retrieve 100 candidates with each method and then perform re-ranking among these candidates using the MaxSim score throughout experiments.

Metrics We report Recall@10 and Queries Per Second (QPS) throughout the experiments. Runtime for computing QPS consists of the candidate retrieval latency, re-ranking time, and the overhead of transforming multi-vector queries into single-vector representations for all methods.

8.2 EXPERIMENTAL RESULTS

Performance of AC-FDE Figure 2 shows the recall-efficiency trade-offs of AC-FDE and baseline methods for accelerating MVR. AC-FDE consistently outperforms all baselines, achieving up to $4.3\times$ speedup at equivalent recall. The runtime breakdown shown in Table 2 confirms that AC-FDE drastically reduces transformation time through block-wise PCA while keeping the online query transformation lightweight via linear operations, unlike LEMUR's expensive transformation via a

D	Method	ColBERT (1-Recall@N) 1	10	50	100	200	500	LateOn (1-Recall@N) 1	10	50	100	200	500
1280	AC-FDE	**27.72**	**68.90**	**85.01**	**88.01**	**89.94**	**91.30**	**24.07**	**61.96**	**80.59**	**85.59**	**89.20**	**92.13**
	LEMUR	22.78	58.52	77.58	82.65	85.69	87.65	18.31	54.44	76.22	83.11	87.99	92.09
2560	AC-FDE	**30.30**	**74.77**	**88.41**	**91.02**	**92.31**	**93.08**	**27.01**	**68.41**	**85.21**	**89.33**	**92.08**	**94.40**
	LEMUR	27.79	69.46	85.77	88.94	90.52	91.25	21.79	61.85	82.58	87.97	91.93	93.80
5120	AC-FDE	**34.76**	**79.15**	**91.05**	**92.91**	**93.78**	**94.33**	**28.58**	**70.13**	**87.05**	**90.67**	**92.85**	**94.58**
	LEMUR	28.94	74.53	89.28	91.56	92.38	92.77	22.91	62.85	83.64	88.93	92.15	94.38

Table 1: 1-Recall@N (%) of AC-FDE and LEMUR on MS MARCO (efSearch$=$ 500, without re-ranking).

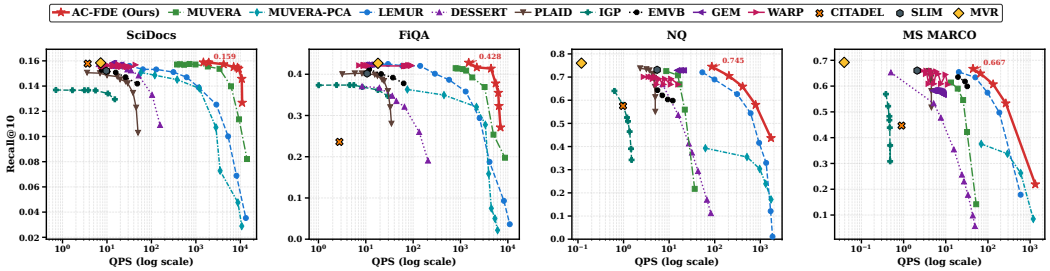


Figure 2: QPS-Recall trade-offs of AC-FDE and baseline approaches.

MLP layer. Multi-vector indexing methods, CITADEL, and SLIM underperform due to unavoidable document-level MaxSim aggregation.

To isolate the effect of re-ranking, we further compare the quality of the candidates retrieved by AC-FDE and LEMUR with varied number of candidates N before re-ranking. We follow prior studies [19; 22] report 1-Recall@N, i.e., the fraction of queries whose ground-truth top-1 document have been included in the top-N candidates before re-ranking. As Table 1 shows, AC-FDE outperforms LEMUR on MS MARCO, especially for small N (by up to 10.4% at $N = 10$).

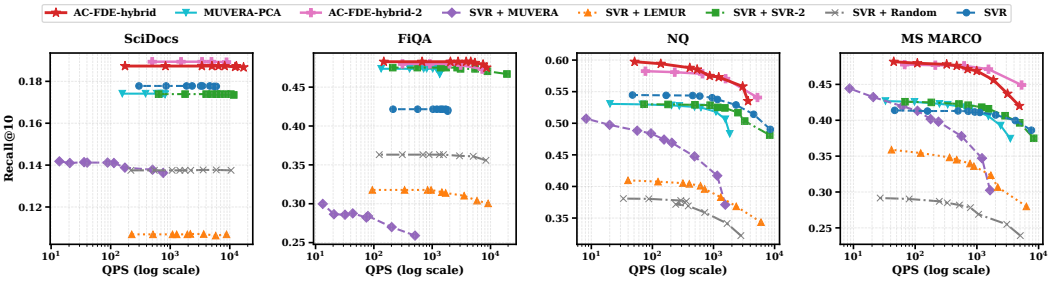


Figure 3: QPS-Recall trade-offs of AC-FDE-hybrid and baseline approaches (Fusing-1 strategy).

Performance of AC-FDE-hybrid Figures 3 and 4 show the recall-efficiency trade-offs of AC-FDE-hybrid, AC-FDE-hybrid-2 and baselines. Notably, AC-FDE-hybrid and AC-FDE-hybrid-2 perform nearly identically and significantly surpass all baselines, regardless of fusion strategy. On NQ, AC-FDE-hybrid achieves two orders of magnitude speedup at recall 0.55. Additionally, we report the query transformation time, i.e., overall overhead of transforming a multi-vector query into single-vector representations, in addition to retrieval time and memory usage of AC-FDE-hybrid and vanilla SVR in Table 14 (see Appendix N.1). This table suggests that AC-FDE-hybrid adds negligible query transformation time and increases memory usage by less than 1 GB compared to vanilla SVR. Overall, AC-FDE-hybrid substantially improves peak recall over baselines, indicating that FDE vectors produced by AC-FDE capture complementary similarity signals not present in standard SVR embeddings.

	MS MARCO				SciDocs				FiQA				NQ			
	MUVERA	MUVERA +PCA	LEMUR	AC-FDE	MUVERA	MUVERA +PCA	LEMUR	AC-FDE	MUVERA	MUVERA +PCA	LEMUR	AC-FDE	MUVERA	MUVERA +PCA	LEMUR	AC-FDE
Transformation	1.0	48.5	28.6	1.6	3.3	75.4	35.1	7.9	3.6	62.5	43.6	6.2	2.5	69.6	36.4	7.5
Retrieval	1087.4	303.2	317.4	274.4	119.3	98.7	124.6	104.3	853.9	1680.2	167.5	158.1	595.0	302.4	371.2	288.8
Rerank	21.1	22.1	23.5	22.7	56.9	46.4	64.8	53.0	40.4	27.7	61.3	38.7	22.0	20.0	31.1	25.4
Total	1109.5	373.8	369.5	**298.8**	179.5	220.4	224.5	**165.2**	897.8†	1770.4†	272.3	**203.0**	619.5	392.1	438.7	**321.8**

Table 2: Amortized per-query time (μs) of each online stage with ColBERT ($D = 2{,}560$), at the Recall@10 that both LEMUR and AC-FDE reach at efSearch$= 100$ (MS MARCO 0.522, SciDocs 0.134, FiQA 0.331, NQ 0.481). Bold: lowest total. †: target not reached; the highest-recall efSearch is shown.

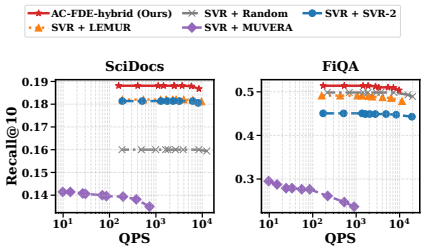


Figure 4: QPS-Recall trade-offs of AC-FDE-hybrid and baseline approaches (Fusing-2 strategy)

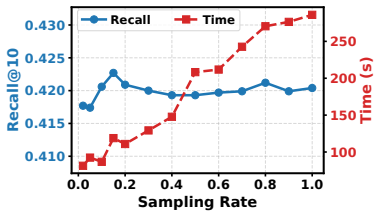


Figure 5: Effect of sampling rate on retrieval accuracy and offline runtime.

Ablation Recall that the offline phase of AC-FDE, i.e., Algorithm 1 consists of five steps, including sampling, repetition selection, FDE vector generation, dimensionality reduction and indexing stage. We therefore report the runtime breakdown for these steps in Table 6. As this table suggests, the offline phase of AC-FDE adds only marginal overhead beyond FDE generation and indexing.

We further evaluate the effect of sampling on offline computation and the subsequent retrieval accuracy by repeating Algorithm 1 with varied sampling rates ranging between 10% and 100%. These results are shown in Figure 5, which shows that the retrieval accuracy is nearly constant for varied sampling rates, while runtime increases up to $10\times$, validating Theorem 5.3. Other ablation studies, including the effect of repetition selection and blockwise PCA, are included in Appendix M.1.

We include the results of other experiments, e.g., offline time analysis and empirical validation of the assumptions in our analysis in Appendix L.2.

9 Related Work

Single-vector retrieval Single-Vector Retrieval (SVR) encodes queries and documents into fixed-dimensional dense vectors [24], enabling efficient similarity search via optimized Maximum Inner Product Search (MIPS) [41; 38]. Numerous indexing methods further accelerate retrieval with minimal accuracy loss [33; 34; 13; 20; 23; 17; 2; 36; 18; 11]. However, the single-vector representation imposes a "lossy compression" that often fails to capture fine-grained semantics or resolve lexical mismatches, particularly at low dimensions [25; 16].

Multi-vector retrieval Multi-Vector Retrieval (MVR) mitigates SVR's representational bottleneck by employing token- or phrase-level embeddings and computing similarity via MaxSim [25; 30]. This expressiveness incurs significant computational and storage costs, making MVR orders of magnitude slower than SVR. Acceleration strategies include token pruning during MaxSim computation [28; 29], direct indexing of multi-vector embeddings [39; 3; 15], and transformation into single vectors to leverage efficient MIPS [19; 31; 22]. The latter approach is implemented either via learned neural mappings [31; 22] or randomized transformations with concatenation [19].

10 Conclusion

AC-FDE improves MUVERA's accuracy–efficiency trade-off through query-independent repetition selection and sampled blockwise compression. Our analysis characterizes proxy distortion, reconstruction error, and sampled-subspace stability under stated assumptions, while retrieval improvements are established empirically. AC-FDE-hybrid further combines compressed FDE representations with single-vector embeddings to exploit complementary retrieval signals.

11 AI USE STATEMENT

In this work, we used generative AI tools for "assist in the writing of proofs" and "design or provide feedback on research methodology or experiments". We have not used generative AI tools for "Generate synthetic data sets", "help develop theoretical models or conceptual frameworks", "formulate mathematical claims", "provide critical ingredients for proving mathematical claims", "propose or refine hypotheses", "implement methods", "assist with translation", "clean and reformat dataset", "support qualitative and thematic data analysis", "interpret results". We have reviewed all AI-assisted work, and we checked LLM-generated research ideas for potential plagiarism through a manual literature survey. In addition, LLM-generated code was verified and tested for correctness by 2 authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *CACM*, 18(9):509–517, 1975. doi: 10.1145/361002.361007.

[3] Zheng Bian, Man Lung Yiu, and Bo Tang. Igp: Efficient multi-vector retrieval via proximity graph index. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2524–2533. ACM, 2025. doi: 10.1145/3726302.3730004.

[4] Danushka Bollegala. Learning meta word embeddings by unsupervised weighted concatenation of source embeddings. *International Joint Conference on Artificial Intelligence*, pp. 4058–4064, 2022. doi: 10.48550/arXiv.2204.12386.

[5] Daniel Fernando Campos, Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, L. Deng, and Bhaskar Mitra. Ms marco: A human generated machine reading comprehension dataset. *CoCo@NIPS*, 2016.

[6] Antoine Chaffin. GTE-ModernColBERT. `https://huggingface.co/lightonai/GTE-ModernColBERT-v1`, 2025.

[7] Antoine Chaffin. Party is over: regularizing ColBERT models to fix efficient ANN methods. `https://huggingface.co/blog/lightonai/lateon-regularization`, 2026.

[8] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Symposium on the Theory of Computing*, pp. 380–388. ACM, 2002. doi: 10.1145/509907.509965.

[9] Arman Cohan, Sergey Feldman, Iz Beltagy, Doug Downey, and Daniel S. Weld. SPECTER: Document-level Representation Learning using Citation-informed Transformers. In *Annual Meeting of the Association for Computational Linguistics*, pp. 2270–2282. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.207.

[10] Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms Condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 758–759, 2009. doi: 10.1145/1571941.1572114.

[11] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04*, pp. 253–262. ACM, 2004. doi: 10.1145/997817.997857.

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, 2019. doi: 10.18653/v1/N19-1423.

[13] Wei Dong, M. Charikar, and K. Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *The Web Conference*, pp. 577–586. ACM, 2011. doi: 10.1145/1963405.1963487.

[14] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936. doi: 10.1007/BF02288367.

[15] Joshua Engels, Benjamin Coleman, Vihan Lakshman, and Anshumali Shrivastava. Dessert: An efficient algorithm for vector set search with vector set queries. *Neural Information Processing Systems*, 36, 2022. doi: 10.48550/arXiv.2210.15748.

[16] Luyu Gao, Zhuyun Dai, and Jamie Callan. Coil: Revisit exact lexical match in information retrieval with contextualized inverted list. In *North American Chapter of the Association for Computational Linguistics*, pp. 3030–3042. Association for Computational Linguistics, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.241.

[17] Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. Optimized product quantization. *IEEE transactions on pattern analysis and machine intelligence*, 36(4):744–755, 2014. doi: 10.1109/TPAMI.2013.240.

[18] Sariel Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Theory of Computing*, pp. 604–613, 2012. doi: 10.4086/toc.2012.v008a014.

[19] Rajesh Jayaram, Laxman Dhulipala, Majid Hadian, Jason D Lee, and Vahab Mirrokni. Muvera: Multi-vector retrieval via fixed dimensional encoding. *Neural Information Processing Systems*, 37:101042–101073, 2024. doi: 10.48550/arXiv.2405.19504.

[20] 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, 2011. doi: 10.1109/TPAMI.2010.57.

[21] W. Johnson and J. Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):189–206, 1984. doi: 10.1090/conm/026/737400.

[22] Elias Jääsaari, Ville Hyvönen, and Teemu Roos. Lemur: Learned multi-vector retrieval, 2026. URL `https://arxiv.org/abs/2601.21853`.

[23] H. Jégou, R. Tavenard, Matthijs Douze, and L. Amsaleg. Searching in one billion vectors: re-rank with source coding. In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 861–864. IEEE, IEEE, 2011. doi: 10.1109/ICASSP.2011.5946540.

[24] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick SH Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Conference on Empirical Methods in Natural Language Processing*, pp. 6769–6781. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

[25] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 39–48, 2020. doi: 10.1145/3397271.3401075.

[26] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, S. Kakade, Prateek Jain, et al. Matryoshka representation learning. *Neural Information Processing Systems*, 35:30233–30249, 2022. doi: 10.52202/068431-2192.

[27] T. Kwiatkowski, J. Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, D. Epstein, I. Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466, 2019. doi: 10.1162/tacl_a_00276.

[28] Minghan Li, Sheng-Chieh Lin, Barlas Oguz, Asish Ghoshal, Jimmy Lin, Yashar Mehdad, Wen-tau Yih, and Xilun Chen. Citadel: Conditional token interaction via dynamic lexical routing for efficient and effective multi-vector retrieval. In *Annual Meeting of the Association for Computational Linguistics*, pp. 11891–11907, 2022. doi: 10.48550/arXiv.2211.10411.

[29] Minghan Li, Sheng-Chieh Lin, Xueguang Ma, and Jimmy Lin. Slim: Sparsified late interaction for multi-vector retrieval with inverted indexes. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 1954–1959. ACM, 2023. doi: 10.1145/3539618.3591977.

[30] Yaoyang Liu, Junlin Li, Yinjun Wu, and zhen chen. POQD: Performance-oriented query decomposer for multi-vector retrieval. In *International Conference on Machine Learning*, 2025. doi: 10.48550/arXiv.2505.19189.

[31] Sean MacAvaney, Antonio Mallia, and Nicola Tonellotto. Efficient constant-space multi-vector retrieval. In *European Conference on Information Retrieval*, pp. 237–245. Springer, Springer Nature Switzerland, 2025. doi: 10.1007/978-3-031-88714-7_22.

[32] Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: financial opinion mining and question answering. In *The Web Conference*, pp. 1941–1942, 2018. doi: 10.1145/3184558.3192301.

[33] 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, 2016. doi: 10.1109/TPAMI.2018.2889473.

[34] Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Information Systems*, 45: 61–68, 2014. doi: 10.1016/j.is.2013.10.006.

[35] Ben Ayad Mohamed Ayoub, Michael Dinzinger, Kanishka Ghosh Dastidar, Jelena Mitrović, and Michael Granitzer. Compressed concatenation of small embedding models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 4613–4617, 2025.

[36] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *International Conference on Computer Vision Theory and Applications*, 2 (331-340):331–340, 2009. doi: 10.5220/0001787803310340.

[37] Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. Efficient multi-vector dense retrieval with bit vectors. In *European Conference on Information Retrieval*, pp. 3–17. Springer, Springer Nature Switzerland, 2024. doi: 10.1007/978-3-031-56060-6_1.

[38] Parikshit Ram and Alexander G Gray. Maximum inner-product search using cone trees. In *Knowledge Discovery and Data Mining*, pp. 931–939. ACM, 2012. doi: 10.1145/2339530.2339677.

[39] Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. Plaid: an efficient engine for late interaction retrieval. In *International Conference on Information and Knowledge Management*, pp. 1747–1756, 2022. doi: 10.1145/3511808.3557325.

[40] Jan Luca Scheerer, Matei Zaharia, Christopher Potts, Gustavo Alonso, and Omar Khattab. Warp: An efficient engine for multi-vector retrieval. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 2504–2512. ACM, 2025. doi: 10.1145/3726302.3729904.

[41] Anshumali Shrivastava and Ping Li. Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips). *Neural Information Processing Systems*, 27, 2014.

[42] Nandan Thakur, Nils Reimers, Andreas Ruckl'e, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *NeurIPS Datasets and Benchmarks*, 2021. URL https://openreview.net/forum?id=wCu6T5xFjeJ.

[43] Yao Tian, Zhoujin Tian, Xi Zhao, Ruiyuan Zhang, and Xiaofang Zhou. Gem: A native graph-based index for multi-vector retrieval. *European Conference on Information Retrieval*, 2026.

[44] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[45] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv.org*, 2025. doi: 10.48550/arXiv.2506.05176.

A MUVERA: A FIXED-DIMENSIONAL ENCODING (FDE) FRAMEWORK FOR MVR

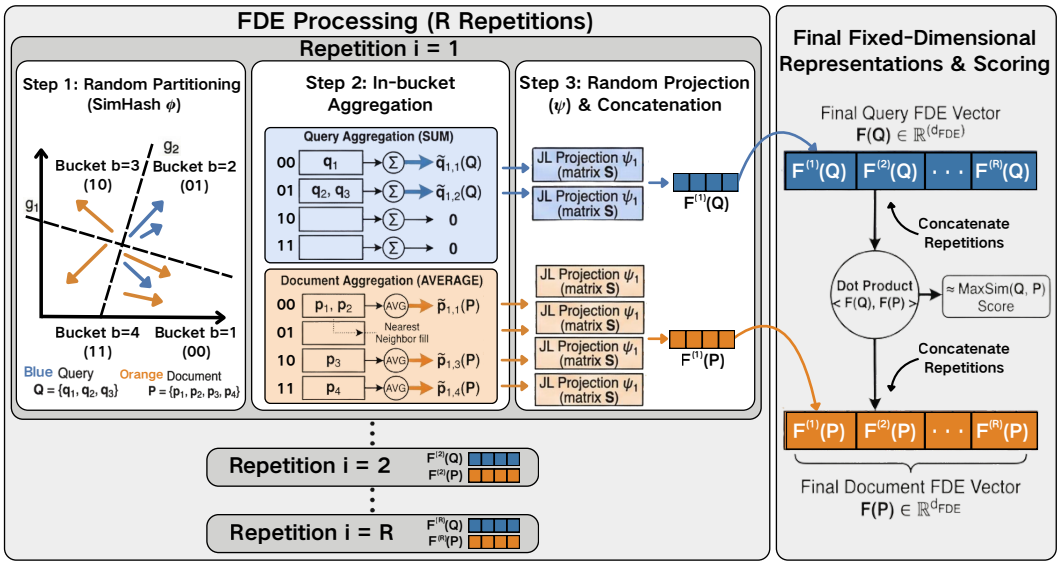


Figure 6: The Muvera pipeline for generating FDE vectors

To leverage SVR optimization techniques while preserving MVR retrieval performance, the **Fixed Dimension Encoder (FDE)** framework [19] was introduced (Figure 6). It maps a variable-sized set $X \subset \mathbb{R}^d$ to a fixed-length vector $F(X) \in \mathbb{R}^{d_{\text{FDE}}}$, such that the similarity between transformed vectors approximates the MaxSim score in the original space:

$$\langle F(Q), F(P) \rangle \approx \text{MaxSim}(Q, P)$$

Formally, let $X = \{x_1, \ldots, x_m\} \subset \mathbb{R}^d$ be a generic multi-vector object (either a query Q or document P). The encoding $F(X)$ is constructed through R *independent repetitions*. Each repetition i goes through the following three steps:

Step 1: Random Partitioning FDE first invokes a random partitioning function $\phi_i : \mathbb{R}^d \to [B]$ to map the space of $\mathbb{R}^d$ into B disjoint buckets. By default, ϕ_i is configured as SimHash [8] which first generates k random Gaussian vectors, $g_1, \cdots, g_k$. These k Gaussian vectors are then used to produce a k-bit signature $\phi_i(x) = [\mathbf{1}(\langle x, g_1 \rangle), \cdots, \mathbf{1}(\langle x, g_k \rangle)]$ for each $x \in X$. Vectors sharing the same bit vector are placed in the same bucket, yielding $B = 2^k$ buckets. This ensures that semantically similar vectors are likely to be co-located in the same bucket. Each query vector $q \in Q$ and each document vector $p \in P$ is assigned to one of the B buckets based on its signature, with $\text{Bit}(b)$ denoting the k-bit signature of the bucket b.

Step 2: in-bucket aggregation An *aggregate vector* is then calculated for the subset of vectors from a multi-vector query Q or a multi-vector document P that fall into the same bucket $b \in [B]$. The aggregation differs between Q and P:

- **Query Aggregation (Sum):** For a query Q, FDE calculates the sum of its vectors within a bucket $b \in [B]$: $\tilde{q}_{i,b}(Q) = \sum_{q \in Q : \phi_i(q) = b} q$.
- **Document Aggregation (Average):** For a document P, FDE calculates the average of its vectors within a bucket $b \in [B]$. If a bucket is empty, a random vector from the closest non-empty bucket is selected as the surrogate average for the bucket, where the closeness between buckets is determined

by the L_0 norm between their k-bit signatures. This is formalized as follows:

$$\tilde{p}_{i,b}(P) = \begin{cases} \dfrac{1}{n_{i,b}(P)} \displaystyle\sum_{p \in P : \phi_i(p) = b} p, & n_{i,b}(P) > 0, \\ p \in P \text{ with } \phi_i(p) = b', & n_{i,b}(P) = 0, \\ & b' = \displaystyle\operatorname*{argmin}_{b'', n_{i,b''}(P) > 0} \|\text{Bit}(p) - \text{Bit}(b'')\|_0 \end{cases}$$

in which, we use $n_{i,b}(P) := \left|\{p \in P : \phi_i(p) = b\}\right|$ to denote the number of document vectors in P falling into the bucket b. When the context is clear, we abbreviate $n_{i,b}(P)$ as $n_{i,b}$.

Step 3: Random Projection and Concatenation FDE then concatenates either $\tilde{q}_{i,b}(Q)$ or $\tilde{p}_{i,b}(P)$ across all buckets for each Q or P to form one vector, yielding a high-dimensional vector of size $B \cdot d$. To mitigate this, a random projection function, $\psi_i(x) = (1/\sqrt{d_{\text{proj}}})Sx$, is applied to $\tilde{q}_{i,b}(Q)$ and $\tilde{p}_{i,b}(P)$, where $S \in \mathbb{R}^{d_{\text{proj}} \times d}$ is a Rademacher Johnson-Lindenstrauss (JL) matrix [21] with random ± 1 entries. After applying this random projection to $\tilde{p}_{i,b}(P)$ and $\tilde{q}_{i,b}(Q)$, the dimensionality of the resulting vectors is reduced. In addition, the inner product between the resulting vectors is guaranteed to remain close to the inner product of the original vectors.

Finally, the projected aggregate vectors are concatenated across buckets within each repetition as follows:

$$\begin{aligned} F_i(Q) &:= \left(\psi_i\big(\tilde{q}_{i,1}(Q)\big)^\top \parallel \cdots \parallel \psi_i\big(\tilde{q}_{i,B}(Q)\big)^\top\right) \in \mathbb{R}^{1 \times B \cdot d_{\text{proj}}}, \\ F_i(P) &:= \left(\psi_i\big(\tilde{p}_{i,1}(P)\big)^\top \parallel \cdots \parallel \psi_i\big(\tilde{p}_{i,B}(P)\big)^\top\right) \in \mathbb{R}^{1 \times B \cdot d_{\text{proj}}}. \end{aligned} \tag{5}$$

In the above formula, $\parallel$ denotes the concatenation operation between vectors or matrices. □

The above three steps are repeated R times, and each time different random functions ϕ_i and ψ_i are selected. The final query representation $F(Q)$ or document representation $F(P)$ is the concatenation of all $F_i(Q)$ or all $F_i(P)$ across all R repetitions, yielding vectors of total dimensions of $d_{\text{FDE}} = R \cdot B \cdot d_{\text{proj}}$:

$$\begin{aligned} F(Q) &:= \Big(F_1(Q) \parallel \cdots \parallel F_R(Q)\Big) \in \mathbb{R}^{1 \times R \cdot B \cdot d_{\text{proj}}}, \\ F(P) &:= \Big(F_1(P) \parallel \cdots \parallel F_R(P)\Big) \in \mathbb{R}^{1 \times R \cdot B \cdot d_{\text{proj}}}, \end{aligned}$$

In what follows, the above two vectors are referred to as the *query FDE vector* and *document FDE vector*, respectively. To facilitate our subsequent formulation, we organize all $F(P)$ across the N documents in $\mathcal{P}$ into the following $N \times d_{\text{FDE}}$ *document FDE matrix*:

$$\mathscr{P} = [F(P_1)^\top \parallel F(P_2)^\top \parallel \cdots \parallel F(P_N)^\top]^\top \in \mathbb{R}^{N \times d_{\text{FDE}}}. \tag{6}$$

Likewise, we introduce two new notations, $\mathscr{P}^{(i)}$ and $\mathcal{P}^{(i,b)}$ to denote stacked vectors across all documents within each repetition and each bucket, respectively:

$$\mathscr{P}^{(i)} = [F_i(P_1)^\top \parallel F_i(P_2)^\top \parallel \cdots \parallel F_i(P_N)^\top]^\top \in \mathbb{R}^{N \times B \cdot d_{\text{FDE}}} \tag{7}$$

$$\mathscr{P}^{(i,b)} = [\psi_i\big(\tilde{p}_{i,b}(P_1)\big)^\top \parallel \cdots \parallel \psi_i\big(\tilde{p}_{i,1}(P_N)\big)^\top]^\top \in \mathbb{R}^{N \times d_{\text{proj}}} \tag{8}$$

These two matrices are also referred to as the *single-repetition sub-matrix* and *single-bucket sub-matrix*, which are matrices comprised of a subset of columns and all rows of $\mathscr{P}$.

Finally, the MaxSim score between P and Q can be approximated by the inner product, $\langle F(Q), F(P)\rangle$, which is equivalent to the sum of the inner product between $F_i(Q)$ and $F_i(Q)$ across all repetitions:

$$\text{MaxSim}(Q, P) \approx \langle F(Q), F(P)\rangle = \frac{1}{R}\sum\nolimits_i \langle F_i(Q), F_i(P)\rangle. \tag{9}$$

In what follows, $\langle F(Q), F(P)\rangle$ is referred to as the *FDE score*, and we also denote $\frac{1}{|Q|}\langle F_i(Q), F_i(P)\rangle$ by $\text{FDE}_i(Q, P)$. Analysis in [19] suggests that *the gap between the FDE score and the true MaxSim score converges to zero, provided that the number of repetitions, R, the number of random partitions per repetition, k, and the projection dimension, d_{proj}, are sufficiently large.*

Theorem A.1. *[19] Assume the number of random Gaussian vectors* $k = O(\frac{\log(m)}{\varepsilon})$, $d_{\text{proj}} = O(\frac{\log(m/\epsilon)}{\varepsilon^2})$, *and* $R = O(\frac{\log N}{\epsilon^2})$, *then the following estimation errors holds with high probability:*

$$\frac{1}{|Q|}\left[\text{MaxSim}(Q,P) - \text{FDE}_i(Q,P)\right] = O(\epsilon)$$

The proof is given in [19].

Time Complexity For each $P \in \mathcal{P}$, we denote the number of vectors in P as $|P|$ vectors in $\mathbb{R}^d$, and further denote $M = \sum_{P\in\mathcal{P}} |P|$. According to the analysis in [19], the time complexity of performing Step 1, Step 2, and Step 3 for each document P is $O(|P| \cdot d \cdot k)$, $O(|P| \cdot 2^k \cdot k)$, and $O(|P| \cdot d \cdot d_{\text{proj}})$, respectively. Summing up these complexity expressions across all R repetitions and all documents in $\mathcal{P}$, the overall complexity of producing all FDE vectors becomes $O(M \cdot R \cdot d \cdot k) + O(M \cdot R \cdot 2^k \cdot k) + O(M \cdot R \cdot d \cdot d_{\text{proj}})$. According to Theorem A.1, *to ensure the closeness between the FDE score and MaxSim score,* d_{proj} *has to be far larger than* k, which indicates that **Step 3 is the dominant step among the above three steps**.

B Proof of Theorem 4.1

In this section, we first provide essential lemmas and corollaries before presenting the full formal description of Theorem 4.1 and the proof of it.

Lemma B.1. *For any* $q \in Q, p \in P$, *let* $\theta(q,p) \in [0,\pi]$ *denote the angle between them and* $\text{Ham}(\phi_i(q), \phi_i(p))$ *the Hamming distance between their signatures, setting* $k_{sim} = O\left(\frac{\log(m/\delta)}{\varepsilon}\right) = O\left(\frac{\log(m)}{\varepsilon}\right)$, *then with at least probability 1-*δ*:*

$$\Pr\left(\left|\text{Ham}(\phi_i(q),\phi_i(p)) - k_{sim}\cdot\frac{\theta(q,p)}{\pi}\right| > \sqrt{\varepsilon}k_{sim}\right) \le \exp^{-2\epsilon k_{sim}} \le \frac{\varepsilon\delta}{m^2}.$$

where m represents the sum of the number of vectors in one multi-vector query Q and one multi-vector document P.

The proof is given in [19].

B.1 Formal description and proof of Theorem 4.1

Lemma B.2 (Average SimHash distortion)**.** *Fix nonempty sets of unit vectors* $Q, P \subset \mathbb{R}^d$. *Let* ϕ_i *use* k *independent Gaussian hyperplanes, and define*

$$\theta(q,p) = \arccos\langle q,p\rangle, \qquad h_i(q,p) = \|\phi_i(q) - \phi_i(p)\|_0.$$

For each query token, let

$$T_i(q) = \max_{p\in P}\left|\frac{h_i(q,p)}{k} - \frac{\theta(q,p)}{\pi}\right|.$$

Then, for any $\delta_{\text{h}} \in (0,1)$, *with probability at least* $1-\delta_{\text{h}}$,

$$\frac{1}{|Q|}\sum_{q\in Q} T_i(q) \le \sqrt{\frac{\log(2|P|)}{2k}} + \sqrt{\frac{\log(1/\delta_{\text{h}})}{2k}}.$$

Proof. For fixed q and p, a Gaussian random hyperplane separates them with probability $\theta(q,p)/\pi$. Thus,

$$Z_{q,p} = \frac{h_i(q,p)}{k} - \frac{\theta(q,p)}{\pi}$$

is an average of k independent centered Bernoulli variables. Hoeffding's lemma gives

$$\mathbb{E}\exp(\lambda Z_{q,p}) \le \exp\left(\frac{\lambda^2}{8k}\right),$$

with the same inequality for $-Z_{q,p}$.

For $\lambda > 0$, the maximum defining $T_i(q)$ satisfies

$$\mathbb{E}\exp(\lambda T_i(q)) \le \sum_{p\in P} [\mathbb{E}\exp(\lambda Z_{q,p}) + \mathbb{E}\exp(-\lambda Z_{q,p})]$$
$$\le 2|P|\exp\left(\frac{\lambda^2}{8k}\right).$$

Jensen's inequality therefore implies

$$\mathbb{E}[T_i(q)] \le \frac{\log(2|P|)}{\lambda} + \frac{\lambda}{8k}.$$

Optimizing over λ yields

$$\mathbb{E}[T_i(q)] \le \sqrt{\frac{\log(2|P|)}{2k}}.$$

Consequently, for

$$\overline{T}_i = \frac{1}{|Q|}\sum_{q\in Q} T_i(q),$$

we have

$$\mathbb{E}[\overline{T}_i] \le \sqrt{\frac{\log(2|P|)}{2k}}.$$

Replacing one hyperplane changes each normalized Hamming distance by at most $1/k$. Hence it changes each $T_i(q)$, and their average $\overline{T}_i$, by at most $1/k$. McDiarmid's inequality gives

$$\Pr\left(\overline{T}_i - \mathbb{E}[\overline{T}_i] \ge t\right) \le \exp(-2kt^2).$$

Setting $t = \sqrt{\log(1/\delta_{\mathrm{h}})/(2k)}$ proves the result. □

Lemma B.3 (JL preservation of token similarities)**.** *Fix nonempty sets of unit vectors $Q, P \subset \mathbb{R}^d$. Let $\psi_i(x) = S_i x/\sqrt{d_{\mathrm{proj}}}$, where S_i has independent Rademacher entries. For $\epsilon_{\mathrm{JL}}, \delta_{\mathrm{JL}} \in (0,1)$, if*

$$d_{\mathrm{proj}} \ge C_0 \epsilon_{\mathrm{JL}}^{-2} \log\left(\frac{4|Q||P|}{\delta_{\mathrm{JL}}}\right),$$

for a sufficiently large universal constant C_0, then, with probability at least $1-\delta_{\mathrm{JL}}$,

$$|\langle\psi_i(q), \psi_i(p)\rangle - \langle q, p\rangle| \le \epsilon_{\mathrm{JL}}$$

simultaneously for all $q \in Q$ and $p \in P$.

Proof. The standard norm-concentration inequality for a Rademacher JL projection states that, for every fixed vector v and $\alpha \in (0,1)$,

$$\Pr\left(\left|\|\psi_i(v)\|_2^2 - \|v\|_2^2\right| > \alpha\|v\|_2^2\right) \le 2\exp(-c\, d_{\mathrm{proj}}\alpha^2),$$

where $c > 0$ is a universal constant.

Apply this inequality with $\alpha = \epsilon_{\mathrm{JL}}$ to the at most $2|Q||P|$ vectors

$$\{q+p,\ q-p : q \in Q,\ p \in P\}.$$

The stated projection dimension and a union bound ensure that all corresponding norm inequalities hold with probability at least $1-\delta_{\mathrm{JL}}$. Zero vectors satisfy these inequalities automatically.

On this event, polarization gives

$$|\langle\psi_i(q), \psi_i(p)\rangle - \langle q, p\rangle|$$
$$\le \frac{1}{4}\left|\|\psi_i(q+p)\|_2^2 - \|q+p\|_2^2\right| + \frac{1}{4}\left|\|\psi_i(q-p)\|_2^2 - \|q-p\|_2^2\right|$$
$$\le \frac{\epsilon_{\mathrm{JL}}}{4}\left(\|q+p\|_2^2 + \|q-p\|_2^2\right) = \epsilon_{\mathrm{JL}},$$

where the final equality uses $\|q\|_2 = \|p\|_2 = 1$. □

Proof of Theorem 4.1. Fix repetition i. For each query token q, let $\mathcal{B}_i(q, P)$ denote the document vectors in the selected nonempty bucket: the bucket of q when it is nonempty, and otherwise a nearest nonempty bucket under Hamming distance. Define the unprojected local maximum

$$m_i^0(q, P) = \max_{p \in \mathcal{B}_i(q,P)} \langle q, p \rangle.$$

We first bound the error from restricting comparisons to this bucket and then account for JL distortion.

Step 1: Bucket-selection error. Every vector $p \in \mathcal{B}_i(q, P)$ satisfies

$$h_i(q, p) = \min_{p' \in P} h_i(q, p').$$

This holds both when the query bucket is nonempty, in which case the minimum distance is zero, and when the nearest-nonempty-bucket fallback is used.

Choose

$$p^\star \in \operatorname*{argmax}_{p \in P} \langle q, p \rangle.$$

Using $T_i(q)$ from Lemma B.2, for every $p \in \mathcal{B}_i(q, P)$,

$$\begin{aligned} \frac{\theta(q,p)}{\pi} &\le \frac{h_i(q,p)}{k} + T_i(q) \\ &\le \frac{h_i(q,p^\star)}{k} + T_i(q) \\ &\le \frac{\theta(q,p^\star)}{\pi} + 2T_i(q). \end{aligned}$$

Because $p^\star$ minimizes the angle to q,

$$0 \le \theta(q,p) - \theta(q,p^\star) \le 2\pi T_i(q).$$

The cosine function is 1-Lipschitz on $[0, \pi]$. Therefore,

$$0 \le \mathrm{MaxSim}(q, P) - m_i^0(q, P) \le 2\pi T_i(q).$$

Apply Lemma B.2 with $\delta_{\mathrm{h}} = \delta/2$. With probability at least $1 - \delta/2$,

$$\frac{1}{|Q|} \sum_{q \in Q} \left[\mathrm{MaxSim}(q, P) - m_i^0(q, P) \right] \le 2\pi a_k,$$

where

$$a_k = \sqrt{\frac{\log(2|P|)}{2k}} + \sqrt{\frac{\log(2/\delta)}{2k}}.$$

Using $(\sqrt{a} + \sqrt{b})^2 \le 2(a + b)$,

$$a_k \le \sqrt{\frac{\log(4|P|/\delta)}{k}}.$$

Consequently, when

$$k \ge \frac{16\pi^2}{\epsilon^2} \log\left(\frac{4|P|}{\delta}\right),$$

the normalized bucket-selection error is at most $\epsilon/2$.

Step 2: JL projection error. Apply Lemma B.3 with

$$\epsilon_{\mathrm{JL}} = \epsilon/2, \qquad \delta_{\mathrm{JL}} = \delta/2.$$

It is sufficient to choose

$$d_{\mathrm{proj}} \ge \frac{4C_0}{\epsilon^2} \log\left(\frac{8|Q||P|}{\delta}\right).$$

With probability at least $1 - \delta/2$, every compared token similarity changes by at most $\epsilon/2$.

The projected proxy

$$m_i(q, P) = \max_{p \in \mathcal{B}_i(q,P)} \langle \psi_i(q), \psi_i(p) \rangle$$

and $m_i^0(q, P)$ maximize over the same candidate set. Hence,

$$|m_i(q, P) - m_i^0(q, P)| \le \epsilon/2.$$

Averaging over query tokens gives

$$\left| \frac{1}{|Q|} \sum_{q \in Q} \big[m_i(q, P) - m_i^0(q, P) \big] \right| \le \epsilon/2.$$

Step 3: Combining the errors. By a union bound, the events in Steps 1 and 2 hold jointly with probability at least $1 - \delta$. On this event,

$$\begin{aligned} &\frac{\big|\mathrm{MaxSim}(Q, P) - \overline{\mathrm{MaxSim}}_i(Q, P)\big|}{|Q|} \\ &\le \frac{1}{|Q|} \sum_{q \in Q} \big[\mathrm{MaxSim}(q, P) - m_i^0(q, P) \big] + \left| \frac{1}{|Q|} \sum_{q \in Q} \big[m_i^0(q, P) - m_i(q, P) \big] \right| \\ &\le \epsilon/2 + \epsilon/2 = \epsilon. \end{aligned}$$

Choosing the universal constant in the theorem as $C \ge \max\{16\pi^2, 4C_0\}$ satisfies both parameter requirements and completes the proof. □

C Supplementary Material for Adaptive Repetition Selection

C.1 Detailed Selection Algorithm

Algorithm 3 outlines the full procedure for generating candidate repetitions and selecting the optimal subset based on the query-independent distortion score Score_i.

Algorithm 3: Adaptive Repetition Selection

Input : Document set $\mathcal{P}$, number of candidate repetitions: $R^{(\mathrm{cand})}$, number of repetitions to retain: R (where $R < R^{(\mathrm{cand})}$), number of buckets B
Output : A subset of R optimal repetition functions $\{(\phi_i, \psi_i)\}$

```
Initialize R ← ∅, S ← ∅
for i ← 1 to R^(Cand) do
    Generate random partitioning function φ_i and JL matrix ψ_i;
    Partition document vectors in P into B buckets using φ_i;
    Compute Score_i using Equation equation 3;
    Append Score_i to S;
    Append {(φ_i, ψ_i)} to R;
Sort R by Score_i in ascending order;
return repetitions in R with the Top-R scores.
```

C.2 An example for justifying the score defined in Equation equation 3

Example 1. *Figure 7 intuitively illustrates why the distortion score can be query-independent. The left and right subfigures depict two random partitioning scenarios that differ only in the position of the random Gaussian vector g_2, which alters the composition of buckets 1 and 4. In the left subfigure, eight diverse vectors (blue solid arrows) are assigned to bucket 1, resulting in a single centroid (red dashed arrow). This centroid lies far from the document vectors near the bucket boundary. Consequently, for any query vector q assigned to this bucket, if its most similar documents are near the boundary, the centroid-based FDE score will deviate significantly from the ground-truth MaxSim score. Conversely, in the right subfigure, these eight vectors are split into both bucket 1 and 4. This reduces the number of vectors within each bucket, ensuring that the centroids remain closer to their respective document vectors. As a result, the FDE score approximates the MaxSim score more accurately for arbitrary query vectors, regardless of their specific position.* □

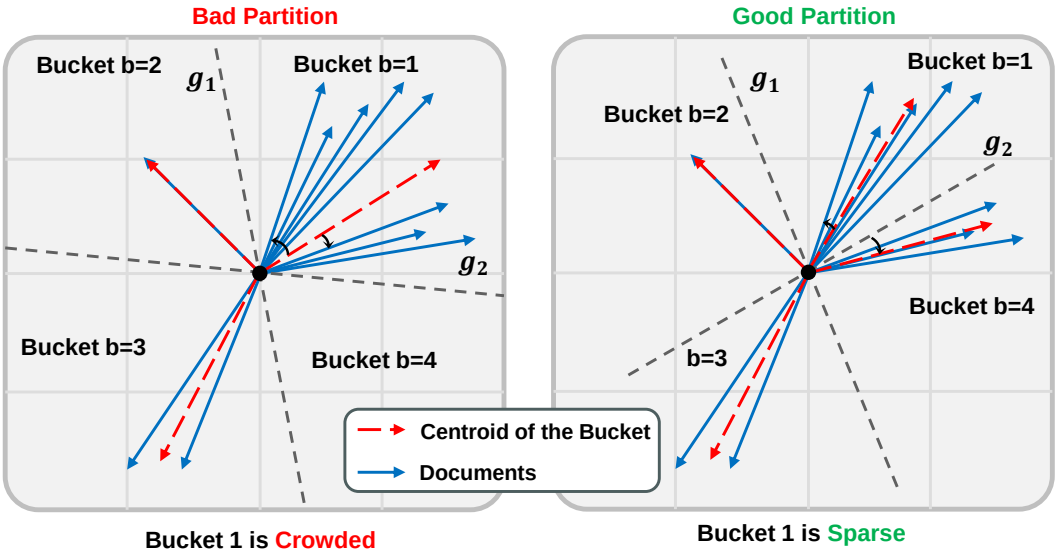


Figure 7: Visualization of the distortion score to distinguish between bad repetitions and good repetitions.

D Proof of Theorem 4.2

Before presenting the proof of Theorem 4.2, we provide an important lemma.

D.1 An important lemma

Lemma D.1 (Maximum–mean gap). *Let $a_1, \ldots, a_n \in [-M, M]$, where $M \geq 0$ and $n \geq 1$. Then*

$$0 \leq \max_j a_j - \frac{1}{n}\sum_{j=1}^{n} a_j \leq 2M\left(1 - \frac{1}{n}\right).$$

Proof. Let $a_{\max} = \max_j a_j$. The lower bound follows because an average cannot exceed the maximum. Moreover,

$$\frac{1}{n}\sum_{j=1}^{n} a_j \geq \frac{a_{\max}}{n} - \frac{n-1}{n}M.$$

Consequently,

$$a_{\max} - \frac{1}{n}\sum_{j=1}^{n} a_j \leq \left(1 - \frac{1}{n}\right)(a_{\max} + M) \leq 2M\left(1 - \frac{1}{n}\right).$$

□

D.2 Proof of Theorem 4.2

Proof. By JL norm preservation and the stated projection dimension, with probability at least $1 - \delta$,

$$\|\psi_i(x)\|_2^2 \leq (1 + \epsilon_{\mathrm{JL}})\|x\|_2^2 = 1 + \epsilon_{\mathrm{JL}}$$

simultaneously for all normalized $x \in Q \cup P$. Condition on this event.

Fix $q \in Q$, and let $\mathcal{B}_i(q, P)$ contain the document vectors in its selected nonempty bucket, using the same nearest-bucket and tie-breaking rules as the FDE construction. Write

$$n = |\mathcal{B}_i(q, P)|, \qquad a_j = \langle \psi_i(q), \psi_i(p_j) \rangle, \quad p_j \in \mathcal{B}_i(q, P).$$

Then $1 \le n \le d_i^{\max}(P)$, and Cauchy–Schwarz gives

$$|a_j| \le \|\psi_i(q)\|_2 \|\psi_i(p_j)\|_2 \le 1 + \epsilon_{\mathrm{JL}}.$$

Let $s_i(q, P)$ denote the contribution of q to $\langle F_i(Q), F_i(P)\rangle$. For a nonempty query bucket, linearity of ψ_i implies

$$s_i(q, P) = \frac{1}{n} \sum_{j=1}^{n} a_j.$$

For an empty query bucket, uniform surrogate selection gives

$$\mathbb{E}_{\mathrm{surr}}[s_i(q, P)] = \frac{1}{n} \sum_{j=1}^{n} a_j.$$

Thus, in either case, the maximum–mean gap bound yields

$$\begin{aligned} 0 &\le m_i(q, P) - \mathbb{E}_{\mathrm{surr}}[s_i(q, P)] \\ &\le 2(1 + \epsilon_{\mathrm{JL}}) \left(1 - \frac{1}{n}\right) \\ &\le 2(1 + \epsilon_{\mathrm{JL}}) \left(1 - \frac{1}{d_i^{\max}(P)}\right). \end{aligned}$$

Finally, query aggregation and linearity imply

$$\langle F_i(Q), F_i(P)\rangle = \sum_{q \in Q} s_i(q, P).$$

Summing the preceding inequalities over $q \in Q$ and dividing by $|Q|$ proves the result. □

E Proof of Theorem 5.1

We first justify the projected-aggregate alignment condition and prove the idealized bound. We then account explicitly for surrogate-filled rows.

E.1 Alignment of genuine bucket aggregates

Fix a repetition and bucket. Suppose its original unit token vectors satisfy

$$\langle p, p' \rangle \ge 1 - \epsilon_{\mathrm{g}}.$$

Let a_j and a_ℓ be the mean token vectors of two genuinely nonempty document buckets. By averaging the pairwise inequalities,

$$\langle a_j, a_\ell \rangle \ge 1 - \epsilon_{\mathrm{g}}, \qquad \|a_j\|_2, \|a_\ell\|_2 \le 1.$$

Suppose the JL projection preserves inner products between these aggregates, including self-products, to additive error at most ϵ_{JL}. For $u_j = \psi_i(a_j)$ and $u_\ell = \psi_i(a_\ell)$,

$$\langle u_j, u_\ell \rangle \ge 1 - \epsilon_{\mathrm{g}} - \epsilon_{\mathrm{JL}}, \qquad \|u_j\|_2, \|u_\ell\|_2 \le \sqrt{1 + \epsilon_{\mathrm{JL}}}.$$

When $\epsilon_{\mathrm{g}} + \epsilon_{\mathrm{JL}} < 1$, these vectors are nonzero, and

$$\frac{\langle u_j, u_\ell \rangle}{\|u_j\|_2 \|u_\ell\|_2} \ge \frac{1 - \epsilon_{\mathrm{g}} - \epsilon_{\mathrm{JL}}}{1 + \epsilon_{\mathrm{JL}}} = 1 - \frac{\epsilon_{\mathrm{g}} + 2\epsilon_{\mathrm{JL}}}{1 + \epsilon_{\mathrm{JL}}}.$$

Thus, the main theorem's alignment condition holds with an angular tolerance of order $O(\epsilon_{\mathrm{g}} + \epsilon_{\mathrm{JL}})$. The following analysis is conditional on this alignment event.

E.2 Proof without surrogate filling

Proof of Theorem 5.1. Fix a nonzero bucket block $M = \mathscr{P}_0^{(i,b)}$, and write its rows as u_j. Let

$$W = \|M\|_F^2 = \sum_j \|u_j\|_2^2, \qquad v_j = \frac{\|u_j\|_2}{\sqrt{W}}.$$

Then $\|v\|_2 = 1$. The alignment condition implies

$$\langle u_j, u_\ell \rangle \geq (1-\epsilon)\|u_j\|_2\|u_\ell\|_2.$$

This inequality also holds when either row is zero.

By the variational characterization of the spectral norm,

$$\begin{aligned}
\|M\|_2^2 &\geq v^\top M M^\top v \\
&= \frac{1}{W}\sum_{j,\ell} \|u_j\|_2\|u_\ell\|_2\langle u_j, u_\ell\rangle \\
&\geq \frac{1-\epsilon}{W}\sum_{j,\ell}\|u_j\|_2^2\|u_\ell\|_2^2 \\
&= (1-\epsilon)W.
\end{aligned}$$

Consequently,

$$\operatorname{srank}(M) = \frac{\|M\|_F^2}{\|M\|_2^2} \leq \frac{1}{1-\epsilon}.$$

No row normalization is performed.

For the full matrix, column-block concatenation gives

$$\|\mathscr{P}_0\|_F^2 = \sum_{i,b} \|\mathscr{P}_0^{(i,b)}\|_F^2,$$

while

$$\|\mathscr{P}_0\|_2^2 \geq \|\mathscr{P}_0^{(i,b)}\|_2^2$$

for every block. Therefore, omitting zero blocks,

$$\begin{aligned}
\operatorname{srank}(\mathscr{P}_0) &= \sum_{i,b} \frac{\|\mathscr{P}_0^{(i,b)}\|_F^2}{\|\mathscr{P}_0\|_2^2} \\
&\leq \sum_{i,b} \operatorname{srank}(\mathscr{P}_0^{(i,b)}) \\
&\leq \frac{RB}{1-\epsilon}.
\end{aligned}$$

The asymptotic expression follows by expanding $(1-\epsilon)^{-1}$ as $\epsilon \to 0$. □

E.3 Extension to surrogate-filled FDE matrices

Let $\mathscr{P}$ denote the actual document FDE matrix. Its genuine bucket rows agree with those of $\mathscr{P}_0$, but its empty-bucket rows contain surrogates instead of zeros.

For each nonzero block, define

$$\rho_{i,b} = 1 - \frac{\|\mathscr{P}_0^{(i,b)}\|_F^2}{\|\mathscr{P}^{(i,b)}\|_F^2}.$$

Thus, $\rho_{i,b}$ is the fraction of the block's squared Frobenius norm contributed by surrogate rows.

Theorem E.1 (Effect of surrogate filling)**.** *Under the genuine-row alignment condition of Theorem 5.1,*

$$\operatorname{srank}(\mathscr{P}) \leq \sum_{i,b} \min\left\{N, d_{\text{proj}}, \frac{1}{(1-\epsilon)(1-\rho_{i,b})}\right\},$$

where the fraction is interpreted as $+\infty$ when $\rho_{i,b} = 1$, and zero blocks are omitted. In particular, if $\bar{\rho} = \max_{i,b} \rho_{i,b} < 1$, then

$$\mathrm{srank}(\mathscr{P}) \le \frac{RB}{(1-\epsilon)(1-\bar{\rho})}.$$

Proof. Fix a nonzero actual block $M = \mathscr{P}^{(i,b)}$ and its zero-filled counterpart $M_0 = \mathscr{P}_0^{(i,b)}$. Write

$$M = M_0 + S,$$

where S contains only surrogate rows. Since M_0 and S have disjoint nonzero row supports,

$$M^\top M = M_0^\top M_0 + S^\top S.$$

Hence

$$\|M\|_2^2 \ge \|M_0\|_2^2.$$

When $\rho_{i,b} < 1$, the idealized block bound gives

$$\|M_0\|_2^2 \ge (1-\epsilon)\|M_0\|_F^2 = (1-\epsilon)(1-\rho_{i,b})\|M\|_F^2.$$

Therefore,

$$\mathrm{srank}(M) \le \frac{1}{(1-\epsilon)(1-\rho_{i,b})}.$$

The generic bound

$$\mathrm{srank}(M) \le \mathrm{rank}(M) \le \min\{N, d_{\mathrm{proj}}\}$$

holds regardless of $\rho_{i,b}$. Combining these bounds and applying the same column-block concatenation inequality as above proves the proposition. □

Interpretation. In our experiments, the maximum surrogate-energy fraction $\bar{\rho}$ remains small, so surrogate filling introduces only a modest increase in the stable-rank bound. Moreover, the observed spectral decay of the actual surrogate-filled FDE matrices indicates that their energy is concentrated in a relatively small number of principal components. These observations support the practical compressibility of the FDE representations despite surrogate filling.

F Additional Details for Section 5

F.1 Block-wise PCA is not equivalent to full PCA

We show a concrete counterexample where performing principal component analysis (PCA) separately on two sub-matrices (blocks) and then concatenating the resulting eigenvectors does *not* yield the eigenvectors of the full data matrix. The failure arises because block-wise PCA discards all cross-covariance between variables belonging to different blocks.

Let the full data matrix $\mathbf{X}$ (already column-centered) be composed of two blocks:

$$\mathbf{X} = [\mathbf{X}_A \quad \mathbf{X}_B],$$

where $\mathbf{X}_A$ contains the first two variables and $\mathbf{X}_B$ the last two. We choose four observations (rows) as follows:

$$\mathbf{X} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & -1 & -1 \\ -2 & -2 & 0 & 0 \end{bmatrix}.$$

Each column has zero mean. The sample covariance matrix (using $n-1=3$) is

$$\mathbf{C} = \frac{1}{3}\mathbf{X}^T\mathbf{X} = \frac{1}{3}\begin{bmatrix} 6 & 5 & 0 & -1 \\ 5 & 6 & -1 & 0 \\ 0 & -1 & 2 & 1 \\ -1 & 0 & 1 & 2 \end{bmatrix}.$$

F.1.1 FULL PCA

The eigenvectors of $\mathbf{C}$ (principal directions) are

$$\mathbf{v}_1 \approx \begin{bmatrix} 0.56 \\ 0.56 \\ -0.40 \\ 0.46 \end{bmatrix}, \quad \mathbf{v}_2 \approx \begin{bmatrix} 0.29 \\ 0.29 \\ 0.77 \\ -0.48 \end{bmatrix},$$
$$\mathbf{v}_3 \approx \begin{bmatrix} -0.42 \\ 0.42 \\ -0.22 \\ -0.78 \end{bmatrix}, \quad \mathbf{v}_4 \approx \begin{bmatrix} 0.64 \\ -0.64 \\ -0.42 \\ 0.09 \end{bmatrix}.$$

Notice that the first two eigenvectors have non-zero entries in all four variables, mixing information from both blocks.

F.1.2 BLOCK-WISE PCA

Perform PCA independently on each block.

Block A (variables 1 and 2):

$$\mathbf{C}_A = \frac{1}{3}\begin{bmatrix} 6 & 5 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 2 & 1.6667 \\ 1.6667 & 2 \end{bmatrix}.$$

Eigenvectors:

$$\mathbf{u}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \mathbf{u}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

Block B (variables 3 and 4):

$$\mathbf{C}_B = \frac{1}{3}\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0.6667 & 0.3333 \\ 0.3333 & 0.6667 \end{bmatrix}.$$

Eigenvectors:

$$\mathbf{w}_1 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \end{bmatrix}, \qquad \mathbf{w}_2 = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

If we naïvely concatenate the block eigenvectors (placing each into the positions of its block and filling the other block with zeros), we obtain the four vectors

$$\begin{bmatrix} \mathbf{u}_1 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{u}_2 \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ \mathbf{w}_1 \end{bmatrix}, \begin{bmatrix} \mathbf{0} \\ \mathbf{w}_2 \end{bmatrix}.$$

None of these equals (or is proportional to) any $\mathbf{v}_i$ from the full PCA. One might instead pair a block-A eigenvector with a block-B eigenvector and concatenate them directly (without zero-padding), e.g. $[\mathbf{u}_1; \mathbf{w}_1]$, $[\mathbf{u}_1; \mathbf{w}_2]$, $[\mathbf{u}_2; \mathbf{w}_1]$, $[\mathbf{u}_2; \mathbf{w}_2]$. Even these candidates fail to match the true eigenvectors; for instance,

$$[\mathbf{u}_1; \mathbf{w}_1] = \frac{1}{\sqrt{2}}\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

is not proportional to $\mathbf{v}_1 \approx [0.56, 0.56, -0.40, 0.46]^T$.

F.1.3 WHY CONCATENATION FAILS

Block-wise PCA ignores the cross-covariance between $\mathbf{X}_A$ and $\mathbf{X}_B$. In the full covariance matrix $\mathbf{C}$, the off-diagonal blocks

$$\mathbf{C}_{AB} = \frac{1}{3}\begin{bmatrix} 0 & -1 \\ -1 & 0 \end{bmatrix}$$

are non-zero. These entries capture the correlation between variables of different blocks. Because the true principal components must account for this cross-structure, they necessarily mix variables from

both blocks in a way that cannot be expressed as a simple concatenation of eigenvectors obtained from independent block-wise analyses. Consequently, PCA on the full matrix is not equivalent to performing block-wise PCA and then concatenating the results. Any analysis that requires capturing correlations across blocks must operate on the full data matrix.

F.2 BLOCKWISE PCA ALGORITHM AND TIME COMPLEXITY

Algorithm 4 outlines the blockwise PCA procedure, which may be performed efficiently using a sampled document subset to compute the covariance and eigen-bases $\widehat{\mathscr{V}}^{(i)}$ before projecting the full matrix $\mathscr{P}^{(i)}$.

Algorithm 4: Blockwise PCA using a sampled document subset

Input : Full FDE matrix $\mathscr{P}$, sampled FDE matrix $\widehat{\mathscr{P}}$, repetitions R, buckets B
Output : Compact FDE matrix $\mathscr{U}$, projection matrices $\mathcal{V}$

1 Initialize lists $\mathcal{L} \leftarrow []$, $\mathcal{V} \leftarrow []$;
2 **for** $i \leftarrow 1$ ***to*** R **do**
3 $\mathrm{Cov}(\widehat{\mathscr{P}}^{(i)}) = \frac{1}{N-1}(\widehat{\mathscr{P}}^{(i)})^\top \widehat{\mathscr{P}}^{(i)}$;
4 Perform PCA on $\mathrm{Cov}(\widehat{\mathscr{P}}^{(i)})$ to obtain top-E eigenvectors $\widehat{\mathscr{V}}^{(i)}$;
5 $\mathscr{U}^{(i)} = \mathscr{P}^{(i)} \cdot \widehat{\mathscr{V}}^{(i)}$;
6 Append $\mathscr{U}^{(i)}$ to $\mathcal{L}$, $\widehat{\mathscr{V}}^{(i)}$ to $\mathcal{V}$;
7 $\mathscr{U} \leftarrow \texttt{Concat}(\mathcal{L})$, $\mathscr{V} \leftarrow \texttt{Concat}(\mathcal{V})$;
8 **return** $\mathscr{U}$, $\mathscr{V}$

Time Complexity. Direct PCA on $\mathscr{P}$ requires: (1) covariance computation $O(N \cdot d_{\mathrm{FDE}}^2)$; (2) PCA $O(d_{\mathrm{FDE}}^3)$; (3) projection $O(N \cdot d_{\mathrm{FDE}} \cdot E)$. In contrast, blockwise PCA reduces these to $O(N d_{\mathrm{FDE}}^2/R)$ and $O(d_{\mathrm{FDE}}^3/R^2)$ for steps (1) and (2), respectively, yielding a factor $1/R$ and $1/R^2$ speedup.

F.3 PROOF OF THEOREM 5.2

Proof. The squared Frobenius reconstruction error separates across column blocks:

$$\left\| \mathcal{P} - U \,\mathrm{diag}\big(V^{(1)\top}, \ldots, V^{(R)\top}\big) \right\|_F^2 = \sum_{i=1}^{R} \left\| \mathcal{P}^{(i)} - \mathcal{P}^{(i)} V^{(i)} V^{(i)\top} \right\|_F^2 .$$

By the Eckart–Young theorem, each summand is minimized by the top-E right singular vectors of $\mathcal{P}^{(i)}$, with minimum value $\sum_{\ell > E} \sigma_\ell(\mathcal{P}^{(i)})^2$. Since the blockwise choices are unconstrained across repetitions, minimizing each summand minimizes their sum. □

G DETAILS OF SAMPLING FOR BLOCKWISE PCA

G.1 ALGORITHMIC STEPS

Algorithm 4 constructs the FDE matrix $\widehat{\mathscr{P}}$ from the subset $\widehat{\mathcal{P}}$ via Equation equation 6. Blockwise PCA on $\widehat{\mathscr{P}}$ produces projection matrices $\widehat{\mathscr{V}}^{(i)}$ (Lines 3–4). The full matrix $\mathscr{P}$ is then projected using these matrices (Line 5).

G.2 PROOF OF THEOREM 5.3

We first provide a complete version of Theorem 5.3 and then provide the complete proof of this theorem.

Theorem G.1 (Stability of sampled blockwise PCA under adaptive selection)**.** *Fix a document corpus $\{P_j\}_{j=1}^N$ and a pool of $R^{(\mathrm{cand})}$ candidate FDE repetitions generated independently of the document sample used below. Condition on all candidate encoding randomness, including surrogate choices, so that each $F_i(P_j)$ is fixed. Write $D_0 = B d_{\mathrm{proj}}$.*

For each candidate $i \in \{1, \dots, R^{(\mathrm{cand})}\}$, define the full-corpus second-moment matrix

$$C_i := \frac{1}{N} \sum_{j=1}^{N} F_i(P_j)^\top F_i(P_j).$$

Draw $J_1, \dots, J_{N_s}$ independently and uniformly from $\{1, \dots, N\}$, and define

$$\widehat{C}_i := \frac{1}{N_s} \sum_{s=1}^{N_s} F_i(P_{J_s})^\top F_i(P_{J_s}).$$

Let $\widehat{\mathcal{I}} \subseteq \{1, \dots, R^{(\mathrm{cand})}\}$ contain R candidates selected by any rule that may depend on this same sample.

Suppose $\|F_i(P_j)\|_2 \le L_i$ for every i, j. For a fixed $1 \le E < D_0$, let $V^{(i)}$ and $\widehat{V}^{(i)}$ contain orthonormal eigenvectors corresponding to the E largest eigenvalues of C_i and $\widehat{C}_i$, respectively. Assume that every candidate has a positive full-corpus eigengap

$$\gamma_i := \lambda_E(C_i) - \lambda_{E+1}(C_i) > 0.$$

Define

$$\nu_i^2 := \left\| \frac{1}{N} \sum_{j=1}^{N} \left(F_i(P_j)^\top F_i(P_j) - C_i \right)^2 \right\|_2 .$$

For $\delta \in (0, 1)$, set

$$t = \log\left(\frac{2R^{(\mathrm{cand})} D_0}{\delta} \right), \qquad \eta_i := \sqrt{\frac{2\nu_i^2 t}{N_s}} + \frac{2L_i^2 t}{3N_s},$$

and

$$\epsilon_{\mathrm{samp}} := \max_{1 \le i \le R^{(\mathrm{cand})}} \frac{\eta_i}{\gamma_i}.$$

Given a sufficiently large N_s so that $\epsilon_{\mathrm{samp}} \ll 1/2$, then, with probability at least $1 - \delta$ over document sampling, simultaneously for all selected repetitions $i \in \widehat{\mathcal{I}}$,

$$\sin\theta_i \le \frac{\eta_i}{\gamma_i - \eta_i} \le \frac{\epsilon_{\mathrm{samp}}}{1 - \epsilon_{\mathrm{samp}}} \le 2\epsilon_{\mathrm{samp}},$$

where

$$\theta_i := \max_{\substack{x \in \mathrm{span}(\widehat{V}^{(i)}) \\ \|x\|_2 = 1}} \min_{\substack{y \in \mathrm{span}(V^{(i)}) \\ \|y\|_2 = 1}} \arccos\left(|x^\top y| \right).$$

Thus, the guarantee permits repetition selection and PCA fitting to use the same document sample. For a fixed corpus, finite candidate pool, and confidence level, $\epsilon_{\mathrm{samp}} \to 0$ as $N_s \to \infty$, and

$$\max_{i \in \widehat{\mathcal{I}}} \theta_i = O(\epsilon_{\mathrm{samp}})$$

on the stated high-probability event.

Proof. All expectations and probabilities are over document sampling, conditional on the corpus and the entire candidate pool, including its encoding randomness.

Step 1: Unbiased estimation for each fixed candidate. For every candidate index i, uniform sampling gives

$$\mathbb{E}\left[F_i(P_{J_s})^\top F_i(P_{J_s}) \right] = \frac{1}{N} \sum_{j=1}^{N} F_i(P_j)^\top F_i(P_j) = C_i.$$

Consequently,

$$\mathbb{E}[\widehat{C}_i] = C_i.$$

This identity holds for the entire second-moment matrix, including its cross-bucket blocks, without requiring independence between buckets. It is asserted for each fixed candidate; we do not condition on that candidate being selected.

Step 2: Uniform concentration over the candidate pool. For a fixed candidate i, define

$$X_{i,s} := F_i(P_{J_s})^\top F_i(P_{J_s}) - C_i.$$

These matrices are independent across sample draws and satisfy $\mathbb{E}[X_{i,s}] = 0$. Since

$$0 \preceq F_i(P_j)^\top F_i(P_j) \preceq L_i^2 I, \qquad 0 \preceq C_i \preceq L_i^2 I,$$

we have

$$\|X_{i,s}\|_2 \le L_i^2.$$

Moreover,

$$\left\| \sum_{s=1}^{N_s} \mathbb{E}[X_{i,s}^2] \right\|_2 = N_s \nu_i^2.$$

Applying the self-adjoint matrix Bernstein inequality to

$$\widehat{C}_i - C_i = \frac{1}{N_s} \sum_{s=1}^{N_s} X_{i,s}$$

yields, for every $u > 0$,

$$\Pr\Big(\|\widehat{C}_i - C_i\|_2 \ge u \Big) \le 2D_0 \exp\left(-\frac{N_s u^2}{2\nu_i^2 + \frac{2}{3} L_i^2 u} \right).$$

With the stated definition of η_i, this implies

$$\Pr\Big(\|\widehat{C}_i - C_i\|_2 > \eta_i \Big) \le 2D_0 e^{-t}.$$

Taking a union bound over all candidates gives

$$\Pr\Big(\exists i \in \{1, \dots, R^{(\mathrm{cand})}\} : \|\widehat{C}_i - C_i\|_2 > \eta_i \Big) \le 2R^{(\mathrm{cand})} D_0 e^{-t} = \delta.$$

No independence across candidates is needed for this union bound, even though their estimates use the same document sample. Therefore, with probability at least $1 - \delta$, the event

$$\mathcal{E} := \Big\{ \|\widehat{C}_i - C_i\|_2 \le \eta_i \le \epsilon_{\mathrm{samp}} \gamma_i \text{ for every candidate } i \Big\}$$

holds.

Step 3: Subspace stability after adaptive selection. On $\mathcal{E}$, the concentration bound holds for every candidate and hence for every member of the sample-dependent set $\widehat{\mathcal{I}}$. Fix any such selected index i. Weyl's inequality gives

$$\lambda_E(\widehat{C}_i) - \lambda_{E+1}(C_i) \ge \gamma_i - \eta_i > 0.$$

The Davis–Kahan sin-Θ theorem therefore implies

$$\left\| \left(I - V^{(i)} V^{(i)\top} \right) \widehat{V}^{(i)} \right\|_2 \le \frac{\eta_i}{\gamma_i - \eta_i}.$$

The largest principal angle satisfies

$$\sin\theta_i = \left\| \left(I - V^{(i)} V^{(i)\top} \right) \widehat{V}^{(i)} \right\|_2.$$

Consequently, simultaneously for all $i \in \widehat{\mathcal{I}}$,

$$\sin\theta_i \le \frac{\eta_i}{\gamma_i - \eta_i} \le \frac{\epsilon_{\mathrm{samp}}}{1 - \epsilon_{\mathrm{samp}}} \le 2\epsilon_{\mathrm{samp}}.$$

Finally, for a fixed finite candidate pool, the quantities L_i, ν_i, and $\gamma_i > 0$ are fixed. Thus $\epsilon_{\mathrm{samp}} \to 0$ as $N_s \to \infty$. Using $\arcsin(u) = O(u)$ near zero gives the stated uniform angle bound. □

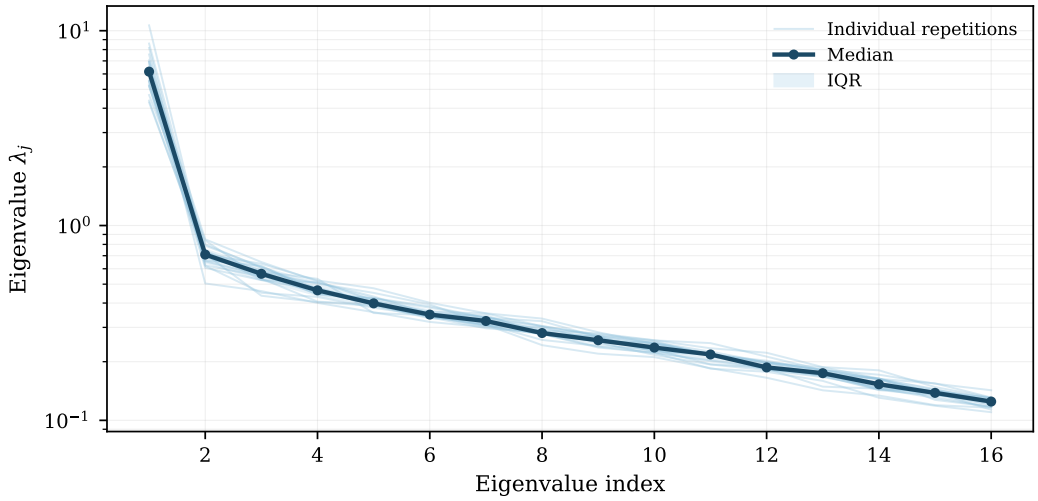


Figure 8: Empirical evidence of large enough gaps between eigen-values of $(\mathscr{P}^{(i)})^\top \mathscr{P}^{(i)}$.

Remark Indeed, the assumption that the gap between different eigen-values of $(\mathscr{P}^{(i)})^\top \mathscr{P}^{(i)}$ holds in practice. In Figure 8, we show the magnitude of all eigen-values of $(\mathscr{P}^{(i)})^\top \mathscr{P}^{(i)}$ in descending order across all repetitions on the SciDoc dataset, which clearly shows that the gap between these eigen-values are not approaching zero.

H Additional Details of Online Query Processing

H.1 Algorithm walk-through

We elaborate on the steps of Algorithm 2. The transformation of Q into $F(Q)$ (line 1) employs the random partitioning functions ϕ_i and random projection functions ψ_i collected during preprocessing. The projection to the eigen-space (line 2) uses the matrix $\mathscr{V}$ obtained via PCA on the document FDE matrix. The resulting compressed query vector $\mathscr{U}^{(Q)}$ is then used with the index $\mathcal{I}$ to retrieve the most relevant documents.

H.2 Time complexity analysis

According to [19], converting a multi-vector query Q into an FDE vector of dimension d_{FDE} requires $O(|Q| \cdot R \cdot d \cdot (k + d_{\text{proj}}))$ time. The subsequent compression via $\mathscr{V}$ to size $R \cdot E$ adds $O(d_{\text{FDE}} \cdot E)$ operations. Retrieval involves comparing the compressed query vector against the compressed document matrix, incurring a worst-case cost of $O(N \cdot R \cdot E)$. Consequently, for large collections (N), retrieval dominates the total query execution time—a trend confirmed experimentally in Section 8.

I Details of AC-FDE-hybrid

Algorithm 5 gives the complete procedure. Arbitrary fusion strategies can be used, e.g., direct concatenation or concatenation with learned weights [4] that weight the two vectors before concatenation.

J Appendix: Detailed Experimental Setup and Supplementary Results

J.1 Detailed dataset description

Datasets We perform experiments with the following datasets from the BEIR benchmark [42]:

Algorithm 5: AC-FDE-hybrid

Input : A document set: $\mathcal{P}$, an embedding model for producing high-dimensional vectors: $\mathcal{E}$, a multi-vector query: Q.
Output : Retrieval results of executing Q

```
// offline FDE matrix and index construction
```

1 Invoke Algorithm 4 to obtain the compressed FDE matrix $\mathscr{U}$, the projection matrix $\mathscr{V}$, and repetitions $\{(\phi_i, \psi_i)\}_{i=1}^{R}$.
2 Encode each document in $P \in \mathcal{P}$ into one embedding using the embedding model $\mathcal{E}$, which is fused with each row embedding in $\mathscr{U}$, yielding one augmented matrix, $\mathscr{U}'$.
3 Build one index $\mathcal{I}$ on $\mathscr{U}'$.

```
// online query processing
```

4 Go through Line 1-2 of Algorithm 2 to transform Q into a compressed FDE vector $\mathscr{U}^{(Q)}$.
5 Encode Q into one embedding with $\mathcal{E}$ and fuse this embedding with $\mathscr{U}^{(Q)}$, yielding one hybrid query embedding $\mathscr{U}^{(Q)'}$.
6 **return** the retrieval results of querying with $\mathscr{U}^{(Q)'}$ on $\mathcal{I}$.

1. **MS MARCO** [5]: a large-scale web retrieval benchmark with 1M+ queries and 8.8M passages.
2. **SciDocs** [9]: a scientific retrieval benchmark with 25.7K papers, 1K queries, and 29.9K relevant pairs (avg 4.9/docs).
3. **FiQA** [32]: a financial QA and retrieval dataset with 6.6K QA pairs and 57.6K documents.
4. **NQ** [27]: an open-domain QA benchmark with 307K Google queries and 21M Wikipedia passages containing annotated answer spans.

Hardware configuration All experiments are conducted on a Linux server equipped with two Intel Xeon Gold 6442Y CPUs (48 cores / 96 threads in aggregate), 512 GB of system memory, and four NVIDIA RTX A6000 GPUs (48 GB each). GPUs are used exclusively for embedding-model inference and for training LEMUR and Fusing-2; FDE construction, blockwise PCA, ANN index building, and online retrieval are executed on CPU. Both index construction and online retrieval rely on the default OpenMP-based parallelism of `faiss` across all 96 hardware threads, and QPS is reported after a warm-up pass under this configuration, identically for every method.

J.2 Detailed Baseline Configurations

We compare AC-FDE against the state-of-the-art methods to accelerate MVR. First, we compare against methods that transform multi-vector representations into single-vectors:

- **MUVERA**: We apply product quantization to MUVERA for vector compression [19].
- **LEMUR** [22]: A neural network is trained to perform the transformation. We follow [22] to build a two-layer MLP. Other training configurations also keep the same as [22].

To measure the efficiency at different recall levels, we vary the number of dimensions for single-vector representations produced by AC-FDE, MUVERA, and LEMUR. For AC-FDE, we vary the number of retained primary components in blockwise PCA. For MUVERA and LEMUR, the value of R and the output dimension of the MLP layer are varied, respectively. For a fixed dimension in those methods, we tune other hyper-parameters, such as $R^{(\text{cand})}$ and k for AC-FDE, to achieve the Pareto optimal between retrieval accuracy and efficiency on a small validation set. Other relevant works such as ConstBERT [31] are excluded for comparison since their code has not been fully released.

We also compare AC-FDE against other methods to accelerate MVR, including:

- **Learned Sparse Representations** (LSR) which learns to collect a sparse subset of embeddings so that the multi-vector representations of each document are compressed. We primarily compare AC-FDE against two state-of-the-art solutions along this line, including **CITADEL** [28] and **SLIM** [29].
- **Multi-vector indexing methods** which construct indexes on multi-vector documents directly for acceleration. We compare AC-FDE against state-of-the-art indexing methods, including **IGP** [3], **PLAID** [39], and **DESSERT** [15].

LEMUR training and inference We re-train LEMUR independently on each dataset over the full corpus, following the training recipe of the official implementation [22]: a two-layer MLP (hidden width 512, GELU activation) is optimised under MSE loss to regress, for every ColBERT input vector, its MaxSim similarities to 8192 anchor documents sampled from the corpus, using Adam with learning rate 3×10^{-3}, batch size 512, 10 epochs, and gradient-norm clipping at 0.5; the linear read-out matrix W is then fitted by ordinary least squares on 16384 additional samples. The retrieval-side embedding dimension is swept over an identical set of candidate values across all four datasets, and the resulting query-side transformation overhead is reported in Table 2.

HNSW index configuration All single-vector retrieval is performed with the HNSW implementation provided by `faiss` (`IndexHNSWFlat`) under the inner-product metric. We adopt the same build-time configuration for every method (AC-FDE, AC-FDE-hybrid, MUVERA, MUVERA-PCA, LEMUR, SVR, and the Fusing-1/Fusing-2 variants): $M = 32$ ($\mathrm{maxM} = 32$, $\mathrm{maxM}_0 = 64$) and $\mathrm{efConstruction} = 200$. Each method's recall–QPS curve is then traced by sweeping $\mathrm{efSearch}$ over $\{16, 32, 64, 128, 256, 512, 1024\}$, ensuring that every baseline reaches its own Pareto frontier under a matched index-build budget.

J.3 Detailed Hybrid Setup

The primary SVR vector is from Qwen3-0.6B (1024-d). The secondary SVR-2 vector is from BERT-base (768-d). For *Fusing-2*, we minimize contrastive loss to learn per-dimension fusion weights. Candidate counts for HNSW are varied to cover different recall levels.

Fusing-2 training details A separate Fusing-2 model is trained on each dataset using its BEIR training queries. The model learns per-dimension positive weights for the primary SVR vector and the FDE vector produced by AC-FDE, MUVERA, LEMUR, or random noise. These weights are all initialized to one. For every training query, we sample one positively relevant document and 64 irrelevant documents, and minimize a pairwise softplus margin loss on the scale-modulated inner-product score, augmented with a small L_2 regularizer that anchors the scales near one. Optimization uses AdamW (learning rate 2×10^{-3}, weight decay 10^{-5}, batch size 256) for 10 epochs.

K Supplementary Retrieval Results

K.1 Effect of Candidate Budget on NQ

Table 3 repeats the candidate-budget comparison of Table 1 on NQ, measuring the candidates before re-ranking. With ColBERT, AC-FDE attains a higher 1-Recall@N than LEMUR for every $N \geq 50$ at all three dimensions, with gains of 1.9–8.3 points at $N = 100$. With LateOn, AC-FDE leads for every $N \leq 50$, and the gain reaches 9.8 points at $N = 10$ for $D = 5{,}120$.

		ColBERT (1-Recall@N)						LateOn (1-Recall@N)					
D	Method	1	10	50	100	200	500	1	10	50	100	200	500
1280	AC-FDE	**17.5**	**53.4**	**73.3**	**78.9**	**82.7**	**85.5**	**20.2**	**53.5**	**72.8**	**77.8**	**81.6**	**86.6**
	LEMUR	14.4	45.8	65.2	70.6	74.7	78.8	14.2	46.3	67.8	74.8	80.1	84.5
2560	AC-FDE	**17.5**	**59.2**	**81.2**	**84.8**	**87.6**	**90.3**	**23.5**	**58.0**	**76.7**	**82.7**	**87.4**	**91.0**
	LEMUR	17.2	55.4	75.8	80.6	84.0	86.1	19.0	53.1	76.5	82.4	86.2	89.3
5120	AC-FDE	**19.8**	**61.4**	**82.8**	**87.8**	**89.9**	**92.0**	**28.1**	**66.5**	**81.9**	**86.7**	**88.4**	**91.5**
	LEMUR	18.5	60.8	81.8	85.9	88.5	90.3	19.3	56.7	78.0	84.3	88.0	90.9

Table 3: 1-Recall@N (%) of AC-FDE and LEMUR on NQ (1,000 queries used for parameter selection, efSearch$= 500$, without re-ranking).

K.2 Results on Anisotropic Multi-Vector Encoders

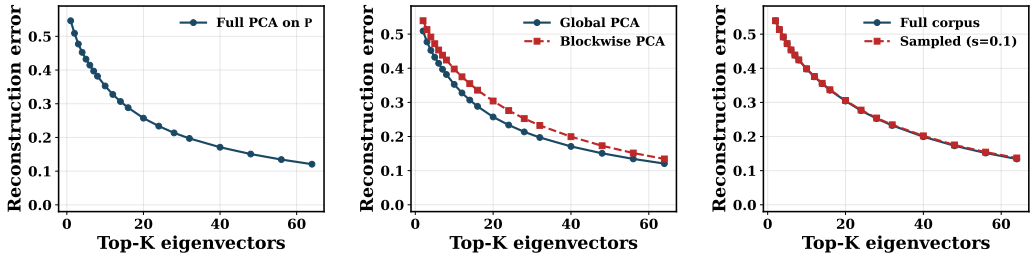
Recent ColBERT models such as GTE-ModernColBERT [6] produce anisotropic token embeddings. The vectors concentrate around a dominant mean direction, and the thin discriminative residual is

difficult to capture with compact random projections [7]. Mean-centering removes the dominant direction. LateOn-regularized [7] is trained so that discriminative information concentrates in a low-dimensional subspace aligned between queries and documents. We evaluate three encoder settings on the full MS MARCO corpus (8.84M documents, 6,980 queries) at $D = 5{,}120$: raw GTE-ModernColBERT, GTE-ModernColBERT after mean-centering, and LateOn-regularized after mean-centering. Centering subtracts the same collection mean from document and query tokens. All methods retrieve candidates by exact inner-product search, and the ground truth is the top-1 document under the raw MaxSim score of each encoder. Table 4 reports 1-Recall@N, the fraction of queries whose exact MaxSim top-1 document appears among the top N candidates.

On raw GTE-ModernColBERT, no method recovers more than 0.30 of the exact top-1 documents at $N = 100$. LEMUR learns a nonlinear mapping on the corpus and still reaches only 0.1910 at $N = 100$, while AC-FDE obtains 2.33 times its 1-Recall@10 and 1.56 times its 1-Recall@100. Centering raises all three methods. After centering, AC-FDE leads at $N \leq 20$, and LEMUR leads at $N = 50$ and $N = 100$ by 0.0107 and 0.0169. On LateOn-regularized, AC-FDE achieves the highest 1-Recall@N at every N, with margins over LEMUR ranging from 0.0061 at $N = 100$ to 0.0493 at $N = 5$. MUVERA stays below 0.60 at $N = 100$ in all three settings.

		1-Recall@N					
Encoder	Method	1	5	10	20	50	100
GTE-ModernColBERT (raw)	AC-FDE	**0.0388**	**0.1024**	**0.1403**	**0.1822**	**0.2424**	**0.2983**
	LEMUR	0.0120	0.0387	0.0602	0.0908	0.1427	0.1910
	MUVERA	0.0023	0.0057	0.0089	0.0125	0.0185	0.0262
GTE-ModernColBERT (centered)	AC-FDE	**0.2507**	**0.5372**	**0.6506**	**0.7543**	0.8523	0.9063
	LEMUR	0.2357	0.5199	0.6401	0.7481	**0.8630**	**0.9232**
	MUVERA	0.0950	0.2440	0.3225	0.4099	0.5209	0.5967
LateOn-regularized (centered)	AC-FDE	**0.2900**	**0.5887**	**0.7029**	**0.7918**	**0.8775**	**0.9244**
	LEMUR	0.2448	0.5394	0.6562	0.7630	0.8653	0.9183
	MUVERA	0.0997	0.2354	0.3059	0.3852	0.4971	0.5842

Table 4: 1-Recall@N on MS MARCO at $D = 5{,}120$ with exact inner-product candidate search. Bold marks the best method for each encoder setting and N.



(a) (b) (c)

Figure 9: Reconstruction error of $\mathscr{P}$ using the Top-K eigen-vectors collected via PCA on $\mathscr{P}$ (Figure 9a), blockwise PCA on $\mathscr{P}$ (Figure 9b) and blockwise PCA on subsampled $\mathscr{P}$ (Figure 9c), respectively

Empirical validations on theoretical analysis We empirically validate our theoretical analysis on the SciDoc dataset. For the FDE matrix (each row a 256-dimensional vector) produced by vanilla MUVERA, we first verify its dimensionality reduction potential (Theorem 5.1) by evaluating reconstruction error using the top-K eigenvectors. As Figure 9a shows, the first 64 eigenvectors reduce the error by 80%, while the remaining ones contribute negligibly—confirming low rank. Repeating the experiment with blockwise PCA eigenvectors (using the full document set and a subsampled set) yields similar convergence (Figures 9b and 9c), demonstrating the effectiveness of blockwise PCA and supporting Theorem 5.2 and Theorem 5.3. Finally, we empirically validate Theorem 4.2 (i.e., the positive correlation between Score_i and $\overline{\mathrm{MaxSim}}_i(Q, P) - \mathrm{FDE}_i(Q, P)$) in Appendix M.3.

	MS MARCO				SciDocs				FiQA				NQ			
	MUVERA	MUVERA +PCA	LEMUR	AC-FDE	MUVERA	MUVERA +PCA	LEMUR	AC-FDE	MUVERA	MUVERA +PCA	LEMUR	AC-FDE	MUVERA	MUVERA +PCA	LEMUR	AC-FDE
Transformation	0.8	40.6	25.4	1.8	2.8	74.2	30.9	4.1	4.1	68.7	31.7	6.6	2.7	64.5	33.1	3.8
Retrieval	3381.6	2871.2	266.0	233.9	420.7	232.3	100.7	83.8	449.0	971.6	129.0	122.4	3497.3	3086.2	316.7	281.2
Rerank	20.7	19.2	22.5	20.8	21.2	21.0	45.4	34.8	47.6	43.3	49.5	43.1	22.1	26.8	38.4	27.0
Total	3403.2†	2930.9†	313.9	**256.5**	444.7†	327.5†	177.1	**122.7**	500.8†	1083.6†	210.2	**172.1**	3522.2†	3177.6†	388.2	**312.1**

Table 5: Amortized per-query time (μs) of each online stage with LateOn-regularized ($D = 2{,}560$), at the Recall@10 that both LEMUR and AC-FDE reach at efSearch$= 100$ (MS MARCO 0.522, SciDocs 0.204, FiQA 0.519, NQ 0.626). Bold: lowest total. †: target not reached; the highest-recall efSearch is shown.

Dataset	Offline stage	**AC-FDE**	MUVERA	MUVERA+PCA	LEMUR	PLAID	DESSERT
SciDocs Encoding 62.17 s	Preprocess	6.53	0.00	88.70 (**13.6**×)	689.30 (**105.6**×)	–	–
	Vector generation	9.35	9.35	30.93 (**3.3**×)	61.90 (**6.6**×)	–	–
	Index building	10.60	10.60	10.60	10.60	252.00	252.00
	Pre. + Gen.	**15.88**	9.35	119.63 (**7.5**×)	751.20 (**47.3**×)	–	–
FiQA Encoding 115.98 s	Preprocess	9.90	0.00	95.60 (**9.7**×)	643.80 (**65.0**×)	–	–
	Vector generation	15.60	15.60	63.65 (**4.1**×)	146.40 (**9.4**×)	–	–
	Index building	31.80	31.80	31.80	31.80	72.00	108.00
	Pre. + Gen.	**25.50**	15.60	159.25 (**6.2**×)	790.20 (**31.0**×)	–	–
NQ Encoding 5081.60 s	Preprocess	22.63	0.00	100.80 (**4.5**×)	667.30 (**29.5**×)	–	–
	Vector generation	930.51	930.51	2121.59 (**2.3**×)	4782.00 (**5.1**×)	–	–
	Index building	2128.00	2128.00	2128.00	2128.00	116532.00	183708.00
	Pre. + Gen.	**953.14**	930.51	2222.39 (**2.3**×)	5449.30 (**5.7**×)	–	–
MS MARCO Encoding 13556.66 s	Preprocess	34.20	0.00	95.40 (**2.8**×)	650.30 (**19.0**×)	–	–
	Vector generation	2477.70	2477.70	5743.66 (**2.3**×)	11319.00 (**4.6**×)	–	–
	Index building	7716.00	7716.00	7716.00	7716.00	450000.00	522000.00
	Pre. + Gen.	**2511.90**	2477.70	5839.06 (**2.3**×)	11969.30 (**4.8**×)	–	–

Table 6: Offline runtime breakdown (seconds). The ColBERT token encoding is shared by all methods and is listed under each dataset. Pre. + Gen. is the method-specific cost of turning token embeddings into single vectors (Preprocess + Vector generation). PLAID and DESSERT index the token embeddings directly (–: stage not applicable). Numbers in parentheses give each baseline's cost relative to AC-FDE. The Pre. + Gen. cost of AC-FDE is 4.8–$47.3\times$ lower than LEMUR and 2.3–$7.5\times$ lower than MUVERA+PCA, and AC-FDE adds only 6.5–34.2 s of preprocessing over data-oblivious MUVERA.

L Online and Offline Cost

L.1 Online Query Time Breakdown with LateOn-regularized

Table 5 repeats the breakdown of Table 2 with LateOn-regularized embeddings. All methods re-rank the top-100 candidates with the same MaxSim implementation. The query transformation of AC-FDE takes 1.8–6.6 μs, compared with 25.4–33.1 μs for the MLP of LEMUR and 40.6–74.2 μs for MUVERA+PCA, and it accounts for at most 3.8% of the total time of AC-FDE versus 8–17% for LEMUR. AC-FDE also retrieves and re-ranks faster than LEMUR and has the lowest total time on all four datasets, 1.2–$1.4\times$ lower than LEMUR.

L.2 Offline time analysis

Table 6 compares the offline cost of AC-FDE with the baselines. Preprocess covers document sampling and repetition selection for AC-FDE and MLP training for LEMUR, and vector generation produces the single vector of every document. The only additional step of AC-FDE over MUVERA is this lightweight preprocessing. LEMUR is slower in both stages because it applies an MLP to every token embedding, and PLAID and DESSERT build indexes on the token embeddings directly, which dominates their offline cost on NQ and MS MARCO.

M Additional Analyses of Repetition Selection and Blockwise PCA

This section reports further analyses of the offline components of AC-FDE. We measure first-stage quality with 1-Recall@100, the fraction of queries whose exact MaxSim top-1 document appears among the 100 retrieved candidates, and end-to-end quality with Recall@10 after MaxSim reranking of these candidates.

M.1 Ablation Study Table

We perform an ablation study on the FIQA dataset to assess the effect of Stage 2 (repetition selection) and Stage 4 (blockwise PCA) in Algorithm 1. The results are included in Table 7.

First of all, the first two rows in Table 7 suggest that, by disabling repetition selection, the recall is reduced by $\sim$1% in comparison to vanilla MUVERA, despite the slight speed-ups ($\sim$0.7% faster) on the offline phase. These results demonstrate the necessity of performing repetition selection to enhance retrieval performance.

Furthermore, we perform dimensionality reduction on the full FDE matrix produced by MUVERA rather than via blockwise PCA, yielding the results in the last row of Table 7. The comparison between this row and the first row suggest that blockwise PCA is crucial in accelerating the offline process with marginal drops in the retrieval performance. This matches the analysis for the zero-covariance of single-repetition sub-matrices in Theorem 5.2.

Configuration	Recall	QPS	Offline time (s)
AC-FDE	0.4242	49.53	312.96
w/o repetition selection	0.4154	49.18	294.33
w/o blockwise PCA	0.4253	36.73	2193.13

Table 7: Ablation on FiQA at 95% of full MVR recall.

M.2 Repetition Selection Strategies

We compare the distortion-score selection of AC-FDE with four alternatives that draw R repetitions from the same pool of $R^{(\mathrm{cand})}$ candidates and leave all other components of AC-FDE unchanged. The mixed strategy takes half of the R repetitions by score and the other half at random, while random selection draws all of them uniformly. Anti-selection keeps the R candidates with the largest scores. The query-aware oracle assumes knowledge of the query distribution and keeps the R repetitions with the smallest true MaxSim approximation error, so it cannot be run offline and serves only as a reference for query-independent selection.

Table 8 shows that 1-Recall@100 follows the order selected $>$ mixed $>$ random $>$ anti-selected on all three datasets. Score-based selection trails the oracle by 0.0040 on SciDocs and is on par with it on FiQA. The distortion score is also correlated with the actual MaxSim approximation error on all four datasets (Appendix M.4).

Dataset	Selected	Mixed (50%)	Random	Anti-selected	Oracle
SciDocs	0.9002	0.8962 ± 0.0026	0.8902 ± 0.0016	0.872	0.9042
FiQA	0.8773	0.8664 ± 0.0059	0.8557 ± 0.0092	0.823	0.8767
MS MARCO	0.8748	0.8682 ± 0.0019	0.8561 ± 0.0057	0.8248	–

Table 8: 1-Recall@100 of AC-FDE under different repetition selection strategies. Randomized strategies report mean $\pm$ standard deviation over repeated draws.

M.3 Additional empirical validations for theoretical analysis

Validataion for Theorem 4.2 We validate Theorem 4.2 by measuring the correlation between the per-document score $\mathrm{Score}_i(P) = 2(1 - 1/d_i^{\max}(P))$ in Eq. equation 3 and the realized distortion gap

$\Delta_i(Q,P) = \overline{\mathrm{MaxSim}}_i(Q,P) - \mathrm{FDE}_i(Q,P)$ on FiQA, pooled across all R=16 repetitions (B=32 buckets, 300 documents, 50 queries, identity projection so FDE_i carries no sketching noise). Table 9 shows that $\mathrm{Score}_i(P)$ tracks the realized gap with strong and highly significant positive correlations across all 4,800 (rep, doc) pairs (Spearman ρ=0.62, $p<10^{-300}$; Pearson 0.56; Kendall τ=0.44), confirming that ranking repetitions by smaller Score_i effectively retains those with smaller true distortion.

Table 9: Correlation between the per-document score $\mathrm{Score}_i(P)=2(1-1/d_i^{\max}(P))$ (Eq. equation 3) and the realized distortion gap $\Delta_i(Q,P)=\overline{\mathrm{MaxSim}}_i(Q,P)-\mathrm{FDE}_i(Q,P)$ on FiQA, pooled across all R=16 repetitions (N=4,800 (rep, doc) pairs; B=32, 300 docs, 50 queries, identity projection). Higher is better (↑).

Dataset	Spearman ρ ↑	Pearson r ↑	Kendall τ ↑
FiQA	**0.623** ($p<10^{-300}$)	0.563	0.440

M.4 Distortion Score and the Actual MaxSim Approximation Error

Repetitions are selected offline before any query is observed, so the selection criterion has to be query-independent. Theorem 4.2 bounds the gap between FDE_i and the proxy $\overline{\mathrm{MaxSim}}_i$ (Table 9). We further test whether Score_i tracks the error with respect to the exact MaxSim score. For each candidate repetition i of the $D = 2{,}560$ ColBERT configuration, we average $|\mathrm{MaxSim}(Q,P) - \mathrm{FDE}_i(Q,P)|/|Q|$ over the evaluation queries and the ten highest exact-MaxSim documents of each query. Table 10 shows that this error is positively correlated with Score_i on all four datasets (Spearman $\rho = 0.34$–0.53), so repetitions with lower scores approximate MaxSim more accurately on the documents that determine the ranking.

Table 10: Correlation between Score_i and the MaxSim approximation error of repetition i over the $N = R^{(\mathrm{cand})}$ candidate repetitions (ColBERT, $D = 2{,}560$; top-10 exact-MaxSim documents per query; no inner projection).

Dataset	N	Spearman ρ ↑	Pearson r ↑	Kendall τ ↑
SciDocs	3,840	0.493 ($p<10^{-234}$)	0.501	0.341
FiQA	2,879	0.526 ($p<10^{-203}$)	0.555	0.366
NQ	8,000	0.342 ($p<10^{-217}$)	0.386	0.232
MS MARCO	1,320	0.436 ($p<10^{-61}$)	0.467	0.300

M.5 Stability under Random Seeds

Repetition selection is deterministic once the candidate pool is fixed, so randomness enters only through the generation of the candidate pool and through document sampling. We vary each source separately. First, we regenerate the candidate pool with different seeds and rerun repetition selection and blockwise PCA. Across these runs, the standard deviation stays at or below 0.0155 for 1-Recall@100 and at or below 0.0074 for Recall@10 (Table 11).

Second, we fix the candidate pool, redraw the 10% document sample used by both offline steps, and compare against the same pipeline fitted on the full corpus (Table 12). The sampled runs stay within 0.020 of the full corpus in 1-Recall@100 and within 0.004 in Recall@10, which supports fitting repetition selection and blockwise PCA on a 10% sample.

M.6 Retrieval-Level Effect of the PCA Rank

Figure 9 relates the number of retained eigenvectors to reconstruction error. To measure the effect on retrieval, we fix the parent FDE on the full MS MARCO corpus (8.84M documents) at $R = 30$, $k = 5$, and $d_{\mathrm{proj}} = 16$, which gives 15,360 dimensions, and vary only the rank retained in each block. Table 13 reports Recall@10 after MaxSim reranking, search QPS, index memory, and offline time excluding the shared ColBERT encoding.

Dataset	Dimension	1-Recall@100	Recall@10
SciDocs	30,720 (no PCA)	0.8772 ± 0.0017	0.1609 ± 0.0009
	15,360 (2×)	0.8652 ± 0.0072	0.1601 ± 0.0009
	7,680 (4×)	0.8206 ± 0.0092	0.1570 ± 0.0011
FiQA	30,720 (no PCA)	0.8219 ± 0.0064	0.3807 ± 0.0025
	15,360 (2×)	0.7944 ± 0.0155	0.3793 ± 0.0064
	7,680 (4×)	0.7437 ± 0.0151	0.3667 ± 0.0074

Table 11: AC-FDE with independently generated candidate pools (mean $\pm$ standard deviation). Compression ratios are relative to the 30,720-dimensional FDE without PCA.

		1-Recall@100		Recall@10	
Dataset	D	10% sample	Full corpus	10% sample	Full corpus
SciDocs	2,000	0.6323 ± 0.0008	0.6389	0.1402 ± 0.0010	0.1407
	7,000	0.7901 ± 0.0011	0.81	0.1564 ± 0.0003	0.1564
FiQA	2,000	0.5888 ± 0.0023	0.5918	0.3043 ± 0.0024	0.3043
	7,000	0.7346 ± 0.0010	0.7437	0.3627 ± 0.0022	0.3667

Table 12: AC-FDE fitted on redrawn 10% document samples (mean $\pm$ standard deviation) and on the full corpus, with the candidate pool fixed.

Increasing the rank from 16 to 256 raises Recall@10 from 0.4007 to 0.6665, approaching the 0.6863 of exact MaxSim, while search QPS drops from 903.3 to 136.6 and index memory and offline time grow almost linearly with the rank. The gain saturates beyond $D = 3{,}840$. Rank 256 ($D = 7{,}680$) improves Recall@10 by 0.0210 over rank 128 ($D = 3{,}840$) at about half the QPS and twice the memory. Rank 171 ($D = 5{,}120$, 3× compression) lies 0.0300 below exact MaxSim and sits between these two points in both accuracy and cost.

Rank per block	D	Compression	Recall@10	Search QPS	Index memory (GiB)	Offline time (s)
16	480	32×	0.4007	903.3	10.15	1942
32	960	16×	0.5161	688.9	18.05	2804
64	1,920	8×	0.6002	480.4	33.86	4504
128	3,840	4×	0.6455	281.5	65.48	7930
171	5,120	3×	0.6563	220.4	86.56	10228
256	7,680	2×	0.6665	136.6	128.72	15096

Table 13: Effect of the per-block PCA rank on MS MARCO with a 15,360-dimensional parent FDE. Compression is relative to the parent FDE. Offline time excludes the shared ColBERT encoding.

N Additional Results of AC-FDE-hybrid

N.1 Query Time and Memory Breakdowns

N.2 Comparison with Rank Fusion

Relevance signals from single-vector and multi-vector retrieval can also be combined after retrieval, by searching with each model separately and merging the two rankings. We compare AC-FDE-hybrid under direct concatenation (Fusing-1) with reciprocal rank fusion (RRF) [10] that weights the two rankings equally, and we report the best RRF configuration. The SVR is the Qwen3-0.6B model used in Section 8.

Table 15 shows that AC-FDE-hybrid improves Recall@10 over the SVR alone by 0.0134 on SciDocs and 0.0076 on FiQA. RRF falls below the SVR alone on both datasets, by 0.0236 and 0.0188. Rank fusion also keeps the complete multi-vector store next to the SVR index and needs two retrieval

	MS MARCO		SciDocs		FiQA		NQ	
	SVR	AC-FDE-hybrid	SVR	AC-FDE-hybrid	SVR	AC-FDE-hybrid	SVR	AC-FDE-hybrid
Transformation time	0.00	0.43	0.00	0.10	0.00	0.05	0.00	0.21
Retrieval time	9.90	8.87	0.35	0.28	0.40	0.37	4.26	3.28
Memory (GB)	27.26	28.29	3.91	4.26	2.53	3.53	13.48	13.98

Table 14: Time (*s*) and memory breakdown for AC-FDE-hybrid vs. SVR at 98% vanilla MVR recall.

passes followed by a merge. AC-FDE-hybrid stores one fused vector per document in a single HNSW index and answers each query with one search and no MaxSim reranking.

Method	SciDocs	FiQA
SVR (Qwen3-0.6B)	0.1739	0.4751
AC-FDE-hybrid (Fusing-1)	**0.1873**	**0.4827**
Equal-weight RRF	0.1503	0.4563

Table 15: Recall@10 of AC-FDE-hybrid and of equal-weight reciprocal rank fusion of the SVR and multi-vector rankings.

N.3 A Larger SVR Model

We repeat the fusion comparison on FiQA with the 4,096-dimensional Qwen3-8B embedding model [45] as the SVR. AC-FDE-hybrid and the two baselines fuse the SVR with vectors from AC-FDE, LEMUR, or MUVERA at the same dimensions (500+3,500). Table 16 shows that AC-FDE-hybrid raises Recall@10 from 0.6250 to 0.6790, while fusing the same SVR with LEMUR or MUVERA vectors under direct concatenation (Fusing-1) lowers it to 0.3406 and 0.2014. With learned weighted concatenation (Fusing-2), fusing LEMUR or MUVERA vectors only keeps Recall@10 at the 0.6250 of the SVR alone.

Method	Dimensions	Recall@10
SVR (Qwen3-8B)	4,096	0.6250
AC-FDE-hybrid	500+3,500	**0.6790**
SVR + LEMUR	500+3,500	0.3406
SVR + MUVERA	500+3,500	0.2014
SVR + LEMUR (fusing-2)	500+3,500	0.6250
SVR + MUVERA (fusing-2)	500+3,500	0.6250

Table 16: Recall@10 on FiQA when the SVR is Qwen3-8B.