IMPROVING PARTITION ROUTING IN APPROXIMATE NEAREST NEIGHBOR SEARCH WITH QUERY EXPANSION ROUTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Approximate Nearest Neighbor (ANN) search methods often rely on space partitioning, where clusters are formed and partition routing is typically based solely on the query distance to cluster centroids. This approach neglects the incremental information gained during the search process. We present a novel query-expansion strategy that dynamically expands the set of scanned partitions using partial results from intermediate search stages. During indexing, we precompute neighborhood relationships between vectors and the partitions containing their nearest neighbors. At query time, initial results are retrieved based on centroid distances. Subsequently, an iterative expansion phase leverages these initial results to extend the partition ranking. This feedback loop can improve the recall–throughput trade-off by reducing unnecessary partition scans while maintaining high accuracy. We refer to this method as Query Expansion Routing (QER). We implemented QER in ScaNN and LoRANN, observing improvements on 14 of the 19 evaluated datasets for ScaNN-QER and 16 of 19 for LoRANN-QER. The gains are most pronounced in the high-recall regime. Across the five datasets with the largest gains for each method, ScaNN-QER and LoRANN-QER achieve geometric-mean QPS speedups of $1.34\times$ and $1.94\times$, respectively, computed for each dataset over the recall range jointly attainable by the baseline and the QER variant, within $\mathrm{Recall@100} \in [0.95, 1.0]$. We designed experiments to isolate the influence of approximate distance computation in ScaNN and LoRANN from QER. These experiments show that the method achieves the same recall while analyzing fewer partitions and scanning fewer vectors, demonstrating the value of candidate-driven feedback as a complementary signal for partition selection in clustering-based ANN search.

1 INTRODUCTION

Vector similarity search is a widely studied problem, as many modern machine learning applications use vector embeddings to represent data. The computational cost of exhaustive k-nearest neighbor search over large, high-dimensional datasets motivates an efficient solution to the approximate nearest neighbor search problem. The widespread use of vector search in databases (Wang et al., 2021; Pan et al., 2024), retrieval-augmented generation (RAG) (Lewis et al., 2020; Borgeaud et al., 2022; Wang et al., 2024), and recommendation systems (Covington et al., 2016; Safar et al., 2025) has further driven the development of new ANN methods. Two major families of ANN methods are graph-based and clustering-based approaches. Neither family consistently outperforms the other across all scenarios (Wang et al., 2021). Graph-based methods have demonstrated strong performance in ANN-Benchmarks (Aumüller et al., 2020), while clustering-based approaches can outperform them in particular settings (Jääsaari et al., 2024). For instance, Guo et al. (2020) reported a state-of-the-art throughput-recall trade-off on *GloVe1.2M*, with higher throughput than the evaluated baselines at high recall. Clustering-based indexes can also offer lower memory consumption and are well suited to distributed processing (Jääsaari et al., 2024).

Clustering-based ANN methods typically involve two main stages at query time: partition selection and candidate scoring. During index construction, the dataset is divided into clusters. At query time, a subset of these clusters is selected as space partitions for scanning, reducing the number

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.

of vectors that need to be scored. A common design uses k-means variants to construct the partitions and selects partitions based on query–centroid proximity. This selection strategy, known as centroid-based routing (CBR), is used in ScaNN (Guo et al., 2020), LoRANN (Jääsaari et al., 2024), and Inverted File (IVF) indexes implemented in Faiss (Douze et al., 2024). The common aspect of these implementations is selecting partitions at the beginning of the query phase. In this paper, we present a method of adjusting CBR with a query expansion (QE) strategy which allows us to use partial information obtained during candidate scoring to select promising partition candidates. During index construction, we associate each vector with the distinct partitions that contain its N nearest neighbors (excluding its own partition). This produces at most N candidate partition identifiers per vector. In the query phase, we expand the set of partitions to scan with these partitions. The method is lightweight, incurring low computational cost during the query phase and only small, parameter-controlled memory overhead. Our experiments show that this technique is complementary to CBR, allowing us to achieve the same recall levels with fewer scanned vectors. We will refer to this routing method as QER.

In summary, our contributions are:

- We propose a new Query Expansion Routing (QER) method for partition routing that uses intermediate candidate results to identify additional partitions beyond those selected by centroid-based routing.
- We isolate the effect of partition routing through controlled experiments with exact scoring within scanned partitions, showing that our strategy reaches a given recall after scoring fewer vectors.
- We implemented our technique in ScaNN and LoRANN and demonstrated higher query throughput at matched high-recall levels compared with the unmodified implementations. Our implementation is available at `https://github.com/` [1]

2 Related Works

Graph-based methods rely on high-quality routing decisions. During the query phase, they navigate a proximity graph by iteratively selecting promising neighboring nodes based on their distance to the query. This data-dependent navigation continues until a termination criterion is met. However, graph traversal induces irregular memory access patterns and frequent random memory accesses, which can significantly degrade performance on modern CPUs (Gou et al., 2025; Yue et al., 2024; Zhong et al., 2025; Coleman et al., 2022; Braun & Litz, 2019). Clustering-based methods, often referred to as inverted file (IVF) methods (Jääsaari et al., 2024; Jégou et al., 2011), partition vectors into groups that can be stored and processed contiguously (Douze et al., 2024). Such organization enables cache-friendly sequential memory access and efficient SIMD-based distance computations, making these methods particularly well suited to modern CPU architectures. Several clustering-based ANN implementations, including ScaNN, Faiss, and RaBitQ, employ SIMD-optimized kernels for batched approximate distance computation (André et al., 2015; Douze et al., 2024; Gao & Long, 2024). These methods prioritize memory layout and regular data access patterns as key factors for improving query performance. Chen et al. (2021) point out that the true nearest neighbors of a query may lie outside the partitions whose centroid is closest to the query. Consequently, CBR may probe irrelevant partitions. Since the efficiency of IVF methods strongly depends on the number of probed partitions, reducing this probing overhead is important for improving the recall–latency trade-off. Zeng et al. (2025) refer to this limitation as the Curse of Partition Pruning. They further observe that "hard partitioning cannot inherently achieve low $nprobe$ due to the long-tailed distribution of kNN". To address these limitations, Zeng et al. (2025) propose LIRA, a query-aware probing model that predicts which partitions are likely to contain the nearest neighbors of a given query. In addition, it employs a learning-based redundancy strategy that replicates selected vectors across multiple partitions to reduce the effect of the long-tailed kNN distribution. Similar redundant-assignment strategies have also been explored in SOAR (Sun et al., 2023), SPANN (Chen et al., 2021), and RAIRS (Yang & Chen, 2026).

IVF and graph-based approaches are difficult to combine, as data-dependent routing decisions can easily disrupt sequential memory access patterns. Zhong et al. (2025) observe that FastScan pro-

[1] Due to anonymity requirements, the URL will be revealed in the camera-ready version.

cesses vectors in fixed-size batches, which makes it difficult to efficiently exclude already visited candidates during graph traversal and may therefore result in redundant computations, thereby wasting SIMD bandwidth. SymphonyQG (Gou et al., 2025) addresses this issue by introducing a graph refinement strategy that better aligns the graph structure with the underlying memory layout, enabling batched distance computations in a FastScan-like manner.

Common IVF methods determine the partition routing order at the beginning of query processing. This approach limits the possibility to explore other parts of the space during the scanning process based on collected partial information. Mohoney et al. (2025) address this issue by adaptively selecting partitions to scan based on the geometry of the partitioning and intermediate results of the query. Bruch et al. (2024) present a method of result expansion with nearest neighbors taken from precomputed small kNN graphs for a learned sparse representation for maximum inner product search (MIPS). Query expansion has been used to describe retrieval strategies in which highly ranked results of an initial search are used to initiate additional searches, potentially in an iterative manner (Chum et al., 2007; Hartmann et al., 2016). Our method uses high-quality candidates obtained during an IVF scan to expand the set of partitions associated with the query. Query expansion is built on top of an existing method. We refer to this existing method as the *primary* method.

3 PROPOSED SOLUTION

In this paper, we propose a new query-expansion-based strategy that enriches IVF methods with higher-quality routing decisions without sacrificing their memory-friendly data layout. For each indexed vector, we store identifiers of other partitions containing its nearest neighbors and use these associations to guide iterative partition expansion. Each iteration consists of relatively large blocks of sequential computation executed by the primary method, followed by a lightweight, intermediate result-guided routing step. This separation reduces the performance impact of irregular memory accesses while preserving the advantages of informed routing. Let $\mathrm{kNN}_L(v; X)$ denote a function returning up to L nearest neighbors of v from the set X. We define $p(v)$ as a function that assigns to a vector v the ID of the partition containing v. Because vectors located close to the cluster centroid are likely to be surrounded by vectors belonging to the same partition, collecting N neighbours from other partitions may require considering up to M nearest neighbours overall; we therefore introduce an upper bound M on the number of nearest neighbors considered. We find $\mathcal{N}_M(v) = \mathrm{kNN}_M(v; X)$, then we restrict the ranking to at most $N (N \leq M)$ nearest neighbors outside v's partition. To apply our technique, we precompute a set of candidate partition IDs $\mathcal{P}_N(v)$ for each vector v in the index based on its local neighborhood. This returns at most N unique candidate partition identifiers per vector

$$\mathcal{N}_N(v) = \mathrm{kNN}_N(v; \{u \mid u \in \mathcal{N}_M(v),\ p(u) \neq p(v)\}). \tag{1}$$

$$\mathcal{P}_N(v) = \{p(u) \mid u \in \mathcal{N}_N(v)\}\,. \tag{2}$$

In the query phase, we combine the sets $\mathcal{P}_N(v)$ for all vectors v in the primary top-k results V_p into a new set of partitions to scan. After scanning and scoring, we collect new results V_{qe} and repeat the process until the stop condition is met. V_{qe} contains merged top-k from every QER iteration. We scan any partition only once, so a natural stop condition for this method is scanning until there are no new candidates. In practice, we introduce a *patience* parameter that terminates QER after τ consecutive iterations without improvement. We define improvement as a decrease in the distance between the query and the worst-ranked vector in the current intermediate top-k results. We also introduce a parameter k_{qe} that controls the number of top-ranked vectors used to select additional partitions. The routing step uses the top k_{qe} vectors from V_p for the initial expansion and from V_{qe} for subsequent iterations. This simple procedure allows us to use partial information collected while processing the query. Authors in Fahim et al. (2023) notice that k-means can only form convex clusters of the dataset, which would fail to correctly partition more complex data distributions. Our method does not change the partition boundaries. Instead, it uses proximity information associated with retrieved candidates to identify additional partitions, reducing reliance on centroid distances alone. Figure 1 illustrates a partition selection with our technique.

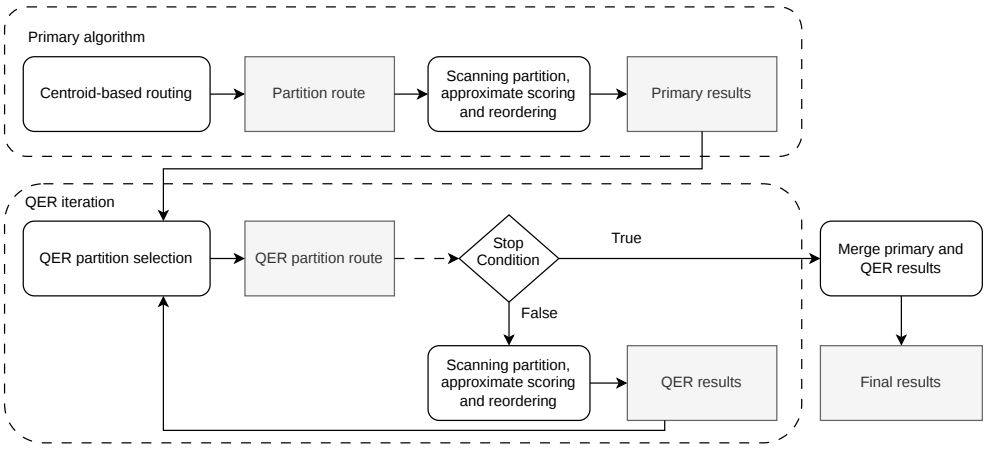

Figure 1: Partition selection with Query Expansion Routing.

3.1 Candidate pre-filtering

While developing the QER extensions for ScaNN and LoRANN, we identified an implementation detail. ScaNN applies preliminary SIMD-based filtering to candidates, using the threshold condition

$$\hat{d}(q, v) < \epsilon,$$

where $\hat{d}(q, v)$ is the approximate distance between the query q and candidate v. Let r denote a reordering parameter. The threshold ϵ is periodically updated using an approximate median-based selection procedure on a buffer used for top-r candidate selection. This procedure retains at least the r best candidates according to their approximate distances. Candidates that fail the threshold test are rejected before individual result-buffer updates, reducing the overhead of candidate processing. Since buffered distances are bounded above by the current threshold, subsequent threshold updates cannot increase ϵ. The filter becomes progressively tighter as the scan advances, utilizing the results obtained from previously scanned partitions. Similar pre-filtering is presented in André et al. (2021). The original LoRANN implementation did not include an analogous cross-partition filter. We implemented one for the QER iterations using the same threshold condition. In both ScaNN-QER and LoRANN-QER, the initial threshold ϵ is obtained from the primary search. ScaNN-QER reuses the final value of ϵ maintained by the primary top-r result buffer. LoRANN-QER uses a more conservative initial threshold corresponding to the i-th smallest approximate distance among the primary candidates, where

$$i = r + \max\{r, 2k_{\text{qe}}\}. \tag{3}$$

3.2 Memory consumption

Our method stores at most N additional partition identifiers per indexed vector, so an index containing n vectors holds no more than nN identifiers. Multiple external nearest neighbors of a vector may belong to the same partition, which can reduce the number of distinct identifiers stored per vector. When partition identifiers fit in 16 bits, storing them requires at most $2N$ bytes per vector, or $2nN$ bytes for the entire index, excluding auxiliary metadata and padding. An example of measured memory consumption for five datasets is reported in Table 1. For fixed N, the upper bound on the additional storage per vector is independent of vector dimensionality. Consequently, the relative overhead is smaller for higher-dimensional vectors, provided that the primary method's per-vector storage increases with dimensionality.

Table 1: Steady-state private memory of ScaNN, LoRANN, and their QER variants.

Dataset		ScaNN-QER			LoRANN-QER		
Name	**Dim**	**Base**	**N=10**	**N=63**	**Base**	**N=10**	**N=63**
agnews-mxbai	1024	3.583	3.604 (+0.6%)	3.678 (+2.7%)	3.207	3.229 (+0.7%)	3.303 (+3.0%)
gooaq-distilroberta	768	4.720	4.761 (+0.9%)	4.902 (+3.9%)	4.626	4.672 (+1.0%)	4.814 (+4.1%)
imagenet-clip	512	2.743	2.778 (+1.3%)	2.901 (+5.8%)	2.709	2.750 (+1.5%)	2.873 (+6.0%)
laion-clip	512	2.147	2.175 (+1.3%)	2.271 (+5.8%)	2.122	2.154 (+1.5%)	2.250 (+6.0%)
yandex	200	0.875	0.903 (+3.2%)	0.999 (+14.2%)	0.859	0.891 (+3.7%)	0.987 (+14.9%)

Memory usage is reported in GB. Values in parentheses indicate the relative QER overhead with respect to the corresponding primary index. Dim denotes the vector dimensionality. $N = 63$ is the largest value for which the 16-bit count and partition IDs fit within a 128-byte aligned row; $N = 64$ exceeds this boundary, increasing the padded row size to 192 bytes.

4 Experiments

All experiments were conducted on a workstation equipped with an AMD Ryzen 9700X CPU, 64 GB of DDR5 memory and an RTX 5080 GPU. We used ANN-Benchmarks (Aumüller et al., 2020) and VIBE (Jääsaari et al., 2026). For VIBE, we evaluated all datasets with a total size up to 5 GB. Larger datasets were excluded due to hardware limitations. For each method, the baseline and its QER-extended variant were compiled with identical build configurations. Throughout this paper, we denote the extended variants with the suffix -QER: ScaNN-QER and LoRANN-QER. Details about the experimental setup are available in Appendix A.

Approximation errors in LoRANN and ScaNN can arise from both routing and approximate candidate scoring. To evaluate routing quality independently of scoring approximations, we conducted an isolated experiment using exact candidate scoring. We first selected promising QER parameter configurations based on the ScaNN-QER and LoRANN-QER results obtained on VIBE. We then used the GPU to evaluate these configurations under exact candidate scoring and compared the Pareto-optimal operating points of QER with those of CBR. We can quantify the effectiveness of our technique using the recall–partition curve. However, since partitions may differ in size, the number of visited partitions does not directly reflect the number of evaluated candidates. Therefore, we also use the recall–candidate curve for comparison.

Let X be a dataset, Q a nonempty set of queries, and $P = \{p_1, \ldots, p_m\}$ a partition of X, i.e., $\forall x \in X \;\; \exists! \, p \in P \;:\; x \in p$. For each query $q \in Q$, let t_q denote the number of partitions visited by the routing procedure for a query q, and let $p_i^{(q)}$, $i = 1, \ldots, t_q$, denote the i-th visited partition, where no partition is visited more than once. Let $G_q \subseteq X$ denote the nonempty ground-truth result set for query q defined by the benchmark. After visiting the first t partitions, the set of examined candidates is

$$S(q, t) = \bigcup_{i=1}^{t} p_i^{(q)}.$$

Let $T = \frac{1}{|Q|} \sum_{q \in Q} t_q$ denote the average number of partitions visited per query. We define the average recall achieved over these query routes as

$$R(Q, G, P) = \frac{1}{|Q|} \sum_{q \in Q} \frac{|G_q \cap S(q, t_q)|}{|G_q|}. \tag{4}$$

The corresponding average number of scanned candidates per query is

$$C(Q, P) = \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{t_q} |p_i^{(q)}| = \frac{1}{|Q|} \sum_{q \in Q} |S(q, t_q)|. \tag{5}$$

We quantify routing effectiveness using the recall–partition and recall–candidate curves, constructed from the points $(T, R(Q, G, P))$ and $(C(Q, P), R(Q, G, P))$, respectively, obtained across the evaluated routing configurations.

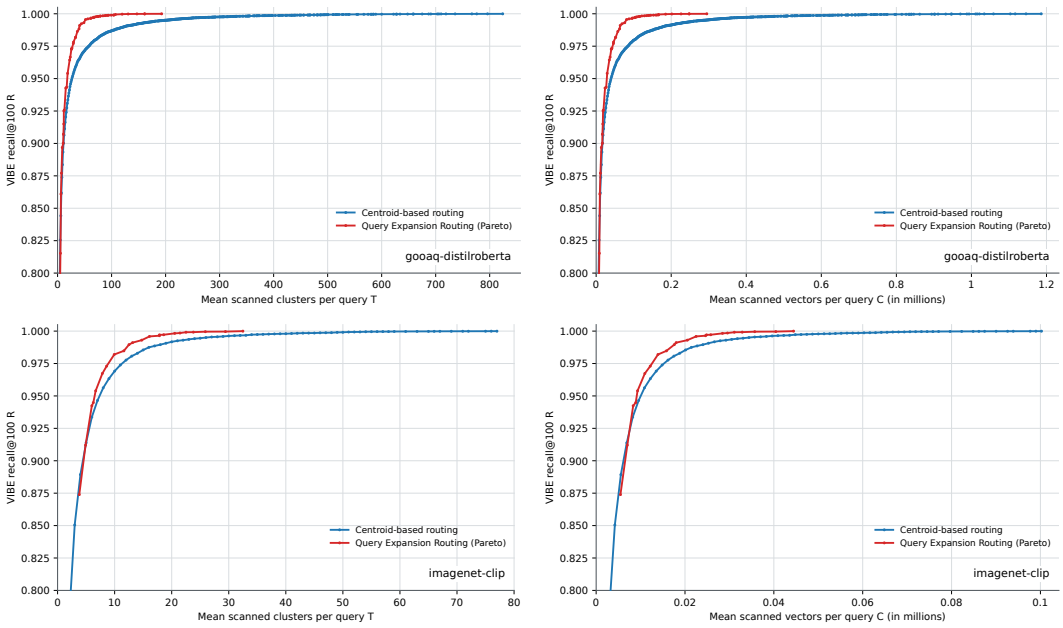

Figure 2: Relationship between average recall@100 R and the average number of scanned clusters T and scanned candidates C for the `gooaq-distilroberta` and `imagenet-clip` datasets from the VIBE benchmark.

4.1 Routing Efficiency under Exact Scoring

At $\text{Recall@100} = 0.99$, QER reduced the number of scanned vectors by up to 64.1%, the largest reduction observed on `gooaq-distilroberta`. For `imagenet-clip`, the reduction was 28.8%, close to the average reduction of 29.9% across the 18 datasets on which QER improved the result. Figure 2 shows that QER achieves better recall–partition and recall–candidate trade-offs than CBR on these datasets. Similar gains in the high-recall regime, for recall@100 above 0.97, were observed on 18 of the 19 evaluated datasets, while routing quality slightly degraded on `llama`. Table 2 further demonstrates that QER is particularly beneficial in the high-recall regime. Averaged across all datasets, QER requires 42.5% fewer vectors to be scanned than CBR to achieve recall@100 of 0.999. However, our experiments with ScaNN-QER and LoRANN-QER show that more effective routing does not always translate into a better recall–QPS trade-off. QER introduces additional computational overhead, which may exceed the cost of scanning a few additional clusters selected by the centroid-based strategy. Another factor is routing error caused by inaccuracies in the candidate scoring and reordering procedure, which can lead QER to prioritize suboptimal partitions. As shown in column (b) of Figure 3, QER is not activated for all queries. For `gooaq-distilroberta`, QER was activated for 96.7% of queries. In contrast, for `imagenet-clip`, QER did not perform a single expansion iteration for 47.4% of queries. This occurs when all candidate partitions proposed by the expansion step have already been visited or when all neighbors of the retrieved vectors lie in the vectors' own partition. This is a desirable property of the method, as QER naturally remains inactive when the local neighborhood provides no evidence for exploring additional partitions. In such cases, CBR may already provide sufficiently accurate partition selection.

4.2 Evaluation of ScaNN-QER and LoRANN-QER

Figure 4 presents QPS–recall@10 curves for two datasets from ANN-Benchmarks, while results for the 19 VIBE datasets are reported in Table 3. For each target recall, we use the highest measured QPS among configurations attaining at least that recall. We summarize high-recall speedup using the recall-uniform geometric mean (GM) of the $\text{QPS}_{\text{QER}}/\text{QPS}_{\text{base}}$ ratio over $[0.95, b]$, where b is the smaller of the maximum measured recalls of the baseline and of the QER variant, determined per dataset and method pair. GM is omitted when $b \leq 0.95$, which occurs for three dataset–method pairs. Consequently, the GM speedup was evaluated on 18 datasets for LoRANN-QER and 17

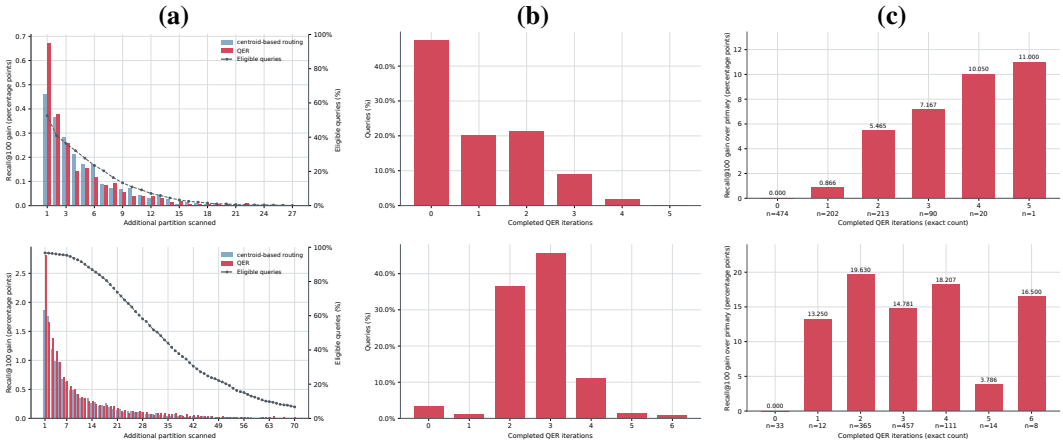

Figure 3: QER diagnostics for `imagenet-clip` (top) and `gooaq-distilroberta` (bottom). (a) Recall gain from the d-th additional partition, weighted by the fraction of all queries whose QER paths reach that position. QER and CBR are evaluated on the same set of eligible queries for which QER was activated. (b) Distribution of the number of completed QER iterations across all queries. (c) Recall gain over primary, averaged within disjoint groups of queries completing exactly the indicated number of QER iterations. Both configurations achieve approximately $\mathrm{Recall@100} = 0.99$. For readability, panel (a) for `gooaq-distilroberta` shows only the first 70 additional partitions.

Table 2: Vector scanning cost ratios of query expansion routing to centroid-based routing.

Dataset	**Ratio $C_{\mathrm{QER}}/C_{\mathrm{CBR}}$ at target recall R**										
Name	**0.90**	**0.91**	**0.92**	**0.93**	**0.94**	**0.95**	**0.96**	**0.97**	**0.98**	**0.99**	**0.999**
agnews-mxbai	1.006	1.039	**0.954**	**0.875**	**0.839**	**0.833**	**0.839**	**0.788**	**0.673**	**0.479**	**0.392**
gooaq-distilroberta	**0.980**	**0.894**	**0.810**	**0.868**	**0.715**	**0.731**	**0.677**	**0.585**	**0.500**	**0.359**	**0.243**
imagenet-clip	1.017	1.017	1.013	1.013	**0.871**	**0.857**	**0.898**	**0.820**	**0.795**	**0.712**	**0.485**
laion-clip	**0.808**	**0.799**	**0.757**	**0.761**	**0.765**	**0.757**	**0.714**	**0.714**	**0.668**	**0.609**	**0.512**
yandex	**0.870**	**0.842**	**0.851**	**0.854**	**0.783**	**0.801**	**0.766**	**0.775**	**0.721**	**0.725**	**0.649**
glove	**0.924**	**0.936**	**0.945**	**0.925**	**0.944**	**0.958**	**0.952**	**0.974**	**0.963**	**0.979**	**0.940**
yahoo-minilm	1.011	**0.949**	**1.000**	**0.950**	**0.898**	**0.877**	**0.830**	**0.820**	**0.804**	**0.696**	**0.476**
coco-nomic	1.005	1.057	1.057	1.157	1.046	**0.925**	**0.922**	**0.851**	**0.845**	**0.795**	**0.607**
ccnews-nomic	**0.926**	**0.877**	**0.932**	**0.822**	**0.922**	**0.783**	**0.790**	**0.727**	**0.704**	**0.575**	**0.480**
celeba-resnet	1.038	1.130	1.022	**0.976**	1.077	**0.958**	**0.996**	1.001	**0.960**	**0.860**	**0.752**
landmark-dino	1.160	1.047	1.007	**0.994**	1.035	1.044	**0.956**	1.003	**0.957**	**0.871**	**0.591**
landmark-nomic	1.100	1.100	1.100	**0.920**	1.025	**0.883**	**0.879**	**0.928**	**0.869**	**0.717**	**0.485**
simplewiki-openai	1.025	**0.984**	1.023	**0.932**	**0.914**	**0.862**	**0.868**	**0.823**	**0.803**	**0.602**	**0.479**
arxiv-nomic	**0.892**	1.109	1.001	**0.837**	**0.923**	**0.903**	**0.799**	**0.797**	**0.782**	**0.755**	**0.582**
imagenet-align	1.063	1.248	1.042	**0.895**	**0.900**	**0.871**	**0.897**	**0.786**	**0.791**	**0.678**	**0.398**
codesearchnet-jina	**0.862**	**0.933**	**0.798**	**0.779**	**0.736**	**0.749**	**0.662**	**0.625**	**0.559**	**0.418**	**0.319**
yi	**0.979**	**0.979**	**0.936**	**0.946**	**0.944**	**0.947**	**0.930**	**0.938**	**0.922**	**0.926**	**0.923**
llama	1.012	1.018	1.009	1.029	1.004	1.017	1.013	1.012	1.013	1.010	1.002
inaturalist-resnet	1.112	1.112	1.024	1.022	1.019	**0.953**	**0.972**	1.026	**0.951**	**0.860**	**0.611**
Mean	**0.989**	1.004	**0.962**	**0.924**	**0.914**	**0.879**	**0.861**	**0.842**	**0.804**	**0.717**	**0.575**

CBR denotes centroid-based routing, and QER denotes Query Expansion Routing. C is the mean number of scanned vectors per query. The table reports $C_{\mathrm{QER}}/C_{\mathrm{CBR}}$ at the indicated minimum mean VIBE recall@100.

datasets for ScaNN-QER. A GM speedup above one was observed for LoRANN-QER on 16 of the 18 evaluated VIBE datasets and for ScaNN-QER on 14 of the 17 evaluated datasets, relative to their respective base methods. Example improvements in the QPS–recall trade-off are shown in Figure 5.

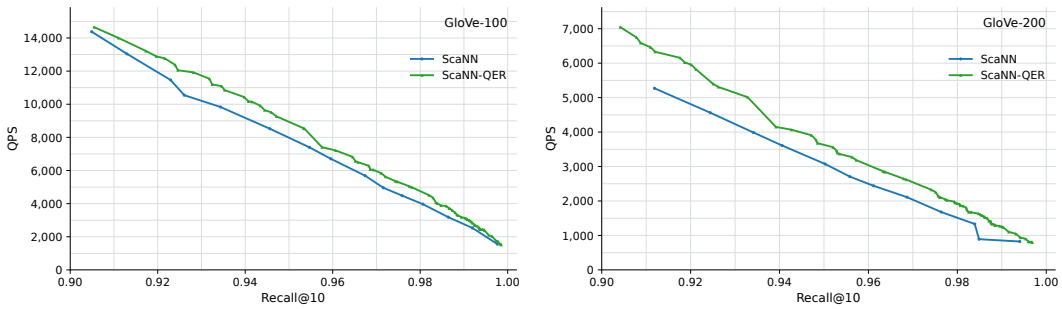

Figure 4: QPS–recall curves for glove-100-angular and glove-200-angular datasets from ANN-Benchmarks.

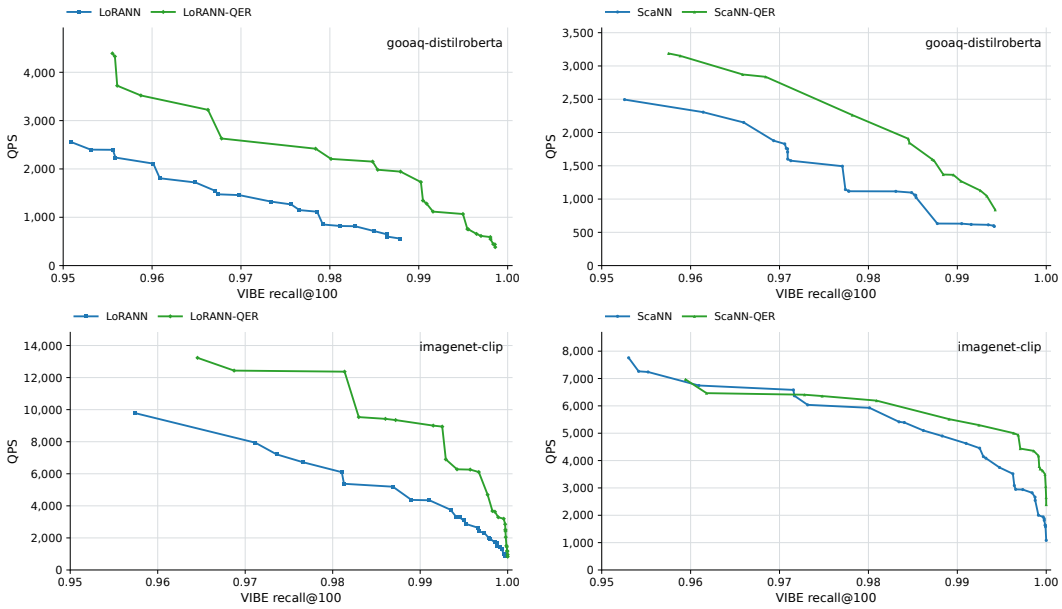

Figure 5: QPS–recall curves for gooaq-distilroberta and imagenet-clip datasets from VIBE.

5 Discussion

Centroid distance is an imperfect proxy for partition relevance, whereas partial information obtained from intermediate search results can provide additional evidence for extending the routing path. QER and CBR are therefore complementary to some extent, as neither provides a perfect partition ranking in all cases. When a vector lies close to its partition centroid, the query-to-centroid distance provides a more accurate approximation of the query-to-vector distance, making CBR more reliable. Moreover, vectors located close to the centroid are more likely to lie in the interior of the partition, with a larger fraction of their local neighborhood contained within the same partition. Consequently, when the nearest neighbors of a query are concentrated around such vectors, CBR is more likely to identify the relevant partition directly, reducing the need for query-expanded routing. On the other hand, QER can provide additional routing information when centroid proximity alone does not sufficiently capture partition relevance, although the expansion itself may still fail to identify the most relevant partitions. QER does not primarily improve the identification of the most obvious partitions. Instead, its benefit becomes apparent when increasingly less obvious partitions must be discovered to achieve high recall. Note that we use the partition of a neighboring vector rather than the vector itself, which provides some tolerance to errors in neighborhood estimation. If, during indexing, the method incorrectly identifies one of the k nearest neighbors, but the selected vector belongs to the same partition as the true neighbor, the behavior of the method remains unchanged.

Table 3: Throughput at target recall and geometric mean speedups of query expansion.

Dataset	$R \geq 0.95$				$R \geq 0.97$				$R \geq 0.99$				GM*	
	ScaNN		**LoRANN**		**ScaNN**		**LoRANN**		**ScaNN**		**LoRANN**		**ScaNN**	**LoRANN**
Name	**Base**	**QER**	**Base**	**QER**	**Base**	**QER**	**Base**	**QER**	**Base**	**QER**	**Base**	**QER**	QER/Base	QER/Base
agnews-mxbai	**2.69**	2.67	2.43	**3.54**	1.75	**2.09**	1.95	**2.15**	0.98	**1.38**	–	1.14	**1.208**	**1.392**
gooaq-distilroberta	2.49	**3.19**	2.55	**4.39**	1.83	**2.26**	1.32	**2.42**	0.63	**1.27**	–	1.73	**1.547**	**2.046**
imagenet-clip	**7.76**	6.97	9.78	**13.23**	**6.58**	6.41	7.94	**12.37**	4.63	**5.30**	4.35	**9.00**	**1.086**	**1.769**
laion-clip	2.52	**2.94**	1.79	**2.59**	2.02	**2.38**	1.13	**1.90**	1.18	**1.76**	–	0.70	**1.250**	**1.650**
yandex	6.37	**6.92**	5.36	**6.85**	4.60	**5.79**	3.31	**5.25**	2.89	**4.70**	1.65	**2.85**	**1.290**	**1.360**
arxiv-nomic	**3.51**	3.37	3.64	**5.89**	3.00	**3.03**	2.94	**3.99**	1.97	**2.04**	1.33	**2.28**	**1.038**	**1.577**
ccnews-nomic	4.11	**4.12**	4.87	**5.35**	3.36	**4.12**	3.55	**5.35**	2.07	**2.68**	**1.84**	1.67	**1.242**	**1.232**
celeba-resnet	3.22	**3.27**	**7.44**	5.61	**2.98**	2.21	4.11	**4.58**	**2.29**	2.21	2.03	**2.42**	0.873	0.981
coco-nomic	**6.18**	5.26	12.92	**13.59**	**5.37**	4.45	9.32	**13.59**	**3.87**	3.27	5.50	**8.00**	0.879	**1.417**
codesearchnet-jina	1.57	**2.02**	1.12	**2.17**	1.11	**1.41**	0.74	**1.69**	–	–	–	0.94	**1.416**	**2.236**
glove	**2.18**	2.13	–	0.53	–	1.63	–	–	–	1.17	–	–	0.976	–
imagenet-align	4.96	**5.29**	4.49	**8.88**	4.08	**4.16**	2.41	**5.68**	2.18	**2.41**	0.83	**2.27**	**1.057**	**2.049**
inaturalist-resnet	2.75	**2.81**	4.38	**7.01**	**2.56**	2.53	2.87	**5.37**	**1.57**	1.43	1.71	**3.14**	**1.023**	**1.573**
landmark-dino	4.83	**4.90**	6.65	**7.78**	**4.13**	3.70	4.47	**7.78**	3.11	**3.44**	3.72	**3.93**	**1.042**	**1.405**
landmark-nomic	6.38	**6.48**	11.77	**13.55**	5.45	**5.59**	11.77	**13.11**	4.56	**4.68**	7.47	**11.97**	**1.069**	**1.429**
llama	–	–	2.89	**2.91**	–	–	**2.89**	2.64	–	–	–	–	–	0.967
simplewiki-openai	1.26	**1.34**	1.43	**2.01**	1.10	**1.22**	1.01	**1.13**	0.67	**0.69**	**0.67**	0.51	**1.051**	**1.036**
yahoo-minilm	**6.65**	6.01	4.66	**5.86**	4.29	**5.77**	3.14	**4.59**	2.38	**4.10**	1.50	**2.28**	**1.228**	**1.342**
yi	–	–	2.40	**3.19**	–	–	–	1.62	–	–	–	–	–	**1.150**

Throughput is reported in 10^3 queries per second, using CPU measurements with parallelism 4 and $k = 100$. GM denotes the recall-uniform geometric mean QER-to-baseline speedup over $[0.95, b]$, using the target-recall convention defined in the text.

We observed that QER is more effective in the high-recall regime. An analysis of the number of scanned vectors indicates that QER converges faster toward $\mathrm{Recall@100} = 1.0$ than CBR, with its advantage increasing as the target recall rises. The computational overhead of the proposed method is small, but remains relevant for highly optimized ANN methods. In some operating regions QER scans fewer vectors than CBR, but the additional fixed overhead prevents this from improving the final benchmark results; in more demanding configurations the overhead is minor relative to the underlying search cost, so its relative impact decreases. Consequently, its relative impact decreases as more computationally demanding configurations are used, which is particularly relevant in the high-recall regime. This is precisely the regime encountered when targeting very high recall. We also observe that the relative improvement is more pronounced for LoRANN than for ScaNN. A likely explanation is the candidate pre-filtering applied only to the QER part of LoRANN-QER, which is not present in the original LoRANN.

6 Summary

We studied partition selection in clustering-based ANN search, where routing decisions are conventionally based on query-to-centroid distances. We introduced a query-expansion routing method that uses partial search results to identify additional promising partitions. Across experiments, QER improved performance in the high-recall regime on the majority of the evaluated datasets. In the routing experiment, QER reduced the number of scanned vectors by an average of 42.5% across all evaluated VIBE datasets at $\mathrm{Recall@100} = 0.999$. On the dataset with the largest gain, ScaNN-QER and LoRANN-QER achieved speedups of $1.547\times$ and $2.046\times$, respectively, over their baselines. Our results indicate that centroid proximity alone does not fully capture partition relevance and that information obtained during the search can be used to improve routing decisions while preserving the optimizations of the underlying ANN method, including memory layout and SIMD-based processing.

7 REPRODUCIBILITY STATEMENT

In Section 3, we provide a detailed description of our method. In Appendix A, we discuss the hardware setup used to run our experiments. Additional implementation details, scripts, and source code required for a full reproduction of the experiments are provided in the supplementary materials. Our implementation is also available at `https://github.com/` [2]

8 AI DISCLOSURE

In this work, we used generative AI tools for: provide feedback on research methodology or experiments, implement methods, support qualitative and thematic data analysis. 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, assist in the writing of proofs, propose or refine hypotheses, design research methodology or experiments. Additionally, we used generative AI tools for create or edit software code, creation of artifacts, searching for information, identify relevant literature, correct grammar and language. We have reviewed all AI-assisted work. LLM-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. Cache locality is not enough: high-performance nearest neighbor search with product quantization fast scan. *Proc. VLDB Endow.*, 9(4):288–299, December 2015. ISSN 2150-8097. doi: 10.14778/2856318.2856324. URL `https://doi.org/10.14778/2856318.2856324`.

Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. Quicker adc : Unlocking the hidden potential of product quantization with simd. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1666–1677, 2021. doi: 10.1109/TPAMI.2019.2952606.

Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374, 2020. ISSN 0306-4379. doi: https://doi.org/10.1016/j.is.2019.02.006. URL `https://www.sciencedirect.com/science/article/pii/S0306437918303685`.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George Bm Van Den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego De Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2206–2240. PMLR, 17–23 Jul 2022. URL `https://proceedings.mlr.press/v162/borgeaud22a.html`.

Peter Braun and Heiner Litz. Understanding memory access patterns for prefetching. In *International Workshop on AI-assisted Design for Architecture (AIDArc), held in conjunction with ISCA*, 2019. URL `https://par.nsf.gov/biblio/10187649-understanding-memory-access-patterns-prefetching`.

Sebastian Bruch, Franco Maria Nardini, Cosimo Rulli, and Rossano Venturini. Pairing clustered inverted indexes with -nn graphs for fast approximate retrieval over learned sparse representations. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, CIKM '24, pp. 3642–3646, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704369. doi: 10.1145/3627673.3679977. URL `https://doi.org/10.1145/3627673.3679977`.

[2] Due to anonymity requirements, the URL will be revealed in the camera-ready version.

Qi Chen, Bing Zhao, Haidong Wang, Mingqin Li, Chuanjie Liu, Zengzhong Li, Mao Yang, and Jingdong Wang. Spann: highly-efficient billion-scale approximate nearest neighbor search. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

O Chum, J Philbin, J Sivic, M Isard, and A Zisserman. Total recall: automatic query expansion with a generative feature model for object retrieval. In *IEEE 11th International Conference on Computer Vision (ICCV 2007)*. IEEE, 2007.

Benjamin Coleman, Santiago Segarra, Alex Smola, and Anshumali Shrivastava. Graph reordering for cache-efficient near neighbor search. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pp. 191–198, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340359. doi: 10.1145/2959100.2959190. URL https://doi.org/10.1145/2959100.2959190.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

Abrar Fahim, Mohammed Eunus Ali, and Muhammad Aamir Cheema. Unsupervised space partitioning for nearest neighbor search. In *International Conference on Extending Database Technology*, 2023. URL https://api.semanticscholar.org/CorpusID:249712089.

Jianyang Gao and Cheng Long. Rabitq: quantizing high-dimensional vectors with a theoretical error bound for approximate nearest neighbor search, 2024. URL https://hdl.handle.net/10356/221432.

Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. Symphonyqg: Towards symphonious integration of quantization and graph for approximate nearest neighbor search. *Proc. ACM Manag. Data*, 3(1), February 2025. doi: 10.1145/3709730. URL https://doi.org/10.1145/3709730.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In Hal Daumé, III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3887–3896. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/guo20h.html.

Wilfried Hartmann, Michal Havlena, and Konrad Schindler. Recent developments in large-scale tie-point matching. *ISPRS Journal of Photogrammetry and Remote Sensing*, 115:47–62, 2016. ISSN 0924-2716. doi: https://doi.org/10.1016/j.isprsjprs.2015.09.005. URL https://www.sciencedirect.com/science/article/pii/S0924271615002063. Theme issue 'State-of-the-art in photogrammetry, remote sensing and spatial information science'.

Elias Jääsaari, Ville Hyvönen, and Teemu Roos. Lorann: Low-rank matrix factorization for approximate nearest neighbor search. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, Advances in Neural Information Processing Systems (NeurIPS), pp. 102121–102153, United States, December 2024. Curran Associates Inc. Conference on Neural Information Processing Systems, NeurIPS ; Conference date: 09-12-2024 Through 15-12-2024.

Elias Jääsaari, Ville Hyvönen, Matteo Ceccarello, Teemu Roos, and Martin Aumüller. VIBE: Vector index benchmark for embeddings. *Journal of Data-centric Machine Learning Research*, 2026. URL https://openreview.net/forum?id=6Sx6ra1QLz.

Herve Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011. doi: 10.1109/TPAMI.2010.57.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf.

Jason Mohoney, Devesh Sarda, Mengze Tang, Shihabur Rahman Chowdhury, Anil Pacaci, Ihab F. Ilyas, Theodoros Rekatsinas, and Shivaram Venkataraman. Quake: adaptive indexing for vector search. In *Proceedings of the 19th USENIX Conference on Operating Systems Design and Implementation*, OSDI '25, USA, 2025. USENIX Association. ISBN 978-1-939133-47-2.

James Jie Pan, Jianguo Wang, and Guoliang Li. Vector database management techniques and systems. In *Companion of the 2024 International Conference on Management of Data*, SIGMOD '24, pp. 597–604, New York, NY, USA, 2024. Association for Computing Machinery. ISBN 9798400704222. doi: 10.1145/3626246.3654691. URL https://doi.org/10.1145/3626246.3654691.

Seema Safar, Babita Roslind Jose, Jimson Mathew, and T. Santhanakrishnan. Recommendation systems with llm-based semantic embeddings and faiss similarity search. *Neurocomputing*, 649:130753, 2025. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2025.130753. URL https://www.sciencedirect.com/science/article/pii/S0925231225014250.

Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. Soar: improved indexing for approximate nearest neighbor search. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Boxin Wang, Wei Ping, Lawrence McAfee, Peng Xu, Bo Li, Mohammad Shoeybi, and Bryan Catanzaro. Instructretro: instruction tuning post retrieval-augmented pretraining. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. Milvus: A purpose-built vector data management system. In *Proceedings of the 2021 International Conference on Management of Data*, SIGMOD '21, pp. 2614–2627, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383431. doi: 10.1145/3448016.3457550. URL https://doi.org/10.1145/3448016.3457550.

Zehai Yang and Shimin Chen. Rairs: Optimizing redundant assignment and list layout for ivf-based ann search. *Proc. ACM Manag. Data*, 4(1), April 2026. doi: 10.1145/3786687. URL https://doi.org/10.1145/3786687.

Qiang Yue, Xiaoliang Xu, Yuxiang Wang, Yikun Tao, and Xuliyuan Luo. Routing-guided learned product quantization for graph-based approximate nearest neighbor search. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pp. 4870–4883, 2024. doi: 10.1109/ICDE60146.2024.00370.

Ximu Zeng, Liwei Deng, Penghao Chen, Xu Chen, Han Su, and Kai Zheng. Lira: A learning-based query-aware partition framework for large-scale ann search. In *Proceedings of the ACM on Web Conference 2025*, WWW '25, pp. 2729–2741, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712746. doi: 10.1145/3696410.3714633. URL https://doi.org/10.1145/3696410.3714633.

Xiaoyao Zhong, Haotian Li, Jiabao Jin, Mingyu Yang, Deming Chu, Xiangyu Wang, Zhitao Shen, Wei Jia, George Gu, Yi Xie, Xuemin Lin, Heng Tao Shen, Jingkuan Song, and Peng Cheng. Vsag: An optimized search framework for graph-based approximate nearest neighbor search. *Proc. VLDB Endow.*, 18(12):5017–5030, August 2025. ISSN 2150-8097. doi: 10.14778/3750601.3750624. URL https://doi.org/10.14778/3750601.3750624.

A EXPERIMENTAL SETUP

All experiments were conducted on a workstation equipped with an AMD Ryzen 9700X CPU with AVX-512 enabled, 64 GB of DDR5-6400 CL32 memory, a Crucial T710 SSD, and an RTX 5080 GPU. We used ANN-Benchmarks (Aumüller et al., 2020) and VIBE (Jääsaari et al., 2026). For ANN-Benchmarks, we evaluated our method on the `glove-100-angular` and `glove-200-angular` datasets. For VIBE, we evaluated all datasets with a total size up to 5 GB. Larger datasets were excluded due to hardware limitations. All query-performance measurements were performed on the CPU. For each vector v, the candidate partition sets $\mathcal{P}_N(v)$ were precomputed using the GPU. In all experiments, we set the upper bound on the number of considered candidates to $M = 100$.

For all experiments involving ScaNN and LoRANN, we used versions compiled from source with native CPU optimizations enabled. We implemented QER as an extension to the original ScaNN (Guo et al., 2020) and LoRANN (Jääsaari et al., 2024) implementations. For each method, the baseline and its QER-extended variant were compiled with identical build configurations. Throughout this paper, we denote the extended variants with the suffix `-QER`: ScaNN-QER and LoRANN-QER.

For ANN-Benchmarks, we used ScaNN as the only primary method and performed exhaustive grid searches over the parameters for both the baseline and its QER-extended variant. All ANN-Benchmarks experiments were run with `--parallelism 1`.

For VIBE, we adopted a different tuning strategy to limit the computational cost of evaluating the expanded parameter space. Since QER introduces three additional hyperparameters, jointly tuning the primary method and QER would substantially increase the number of configurations to evaluate. We used the primary-method hyperparameter configurations from the VIBE repository as a starting point and focused tuning on the additional QER parameters. All VIBE experiments were run with `--parallelism 4`.