Journal of Data-centric Machine Learning Research (2026) Submitted 9/25; Revised 8/26; Published 8/26

VIBE: Vector Index Benchmark for Embeddings

Elias Jääsaari ELIAS.JAASAARI@HELSINKI.FI
Department of Computer Science
University of Helsinki, Finland

Ville Hyvönen VILLE.O.HYVONEN@HELSINKI.FI
Department of Computer Science
University of Helsinki, Finland

Matteo Ceccarello MATTEO.CECCARELLO@UNIPD.IT
Department of Information Engineering
University of Padova, Italy

Teemu Roos TEEMU.ROOS@HELSINKI.FI
Department of Computer Science
University of Helsinki, Finland

Martin Aumüller MAAU@ITU.DK
IT University of Copenhagen, Denmark

Reviewed on OpenReview: *https://openreview.net/forum?id=6Sx6ra1QLz*

Editor: Matthias Feurer

Abstract

Approximate nearest neighbor (ANN) search is a performance-critical component of many machine learning pipelines, and rigorous benchmarking is essential for assessing the performance of vector indexes for ANN search. However, the datasets of existing benchmarks no longer represent modern ANN applications, creating a need for an up-to-date benchmark. To address this gap, we introduce Vector Index Benchmark for Embeddings (VIBE), an open-source framework for benchmarking ANN algorithms. VIBE provides a pipeline for generating benchmark datasets with dense embedding models representative of modern applications, including retrieval-augmented generation (RAG). To represent real-world workloads, we also include out-of-distribution (OOD) datasets where the queries and the corpus are drawn from different distributions. These include multimodal retrieval datasets and maximum inner product search (MIPS) datasets covering two recent use cases: approximate attention computation and reductions of multi-vector retrieval to single-vector MIPS. We use VIBE to conduct a comprehensive evaluation of 22 open-source vector-index implementations across 11 in-distribution and 8 out-of-distribution datasets. The benchmark is available at `https://github.com/vector-index-bench/vibe`

Keywords: vector search, vector index, vector database, approximate nearest neighbor search, maximum inner product search, neural search, embeddings, multi-vector retrieval, approximate attention, retrieval-augmented generation

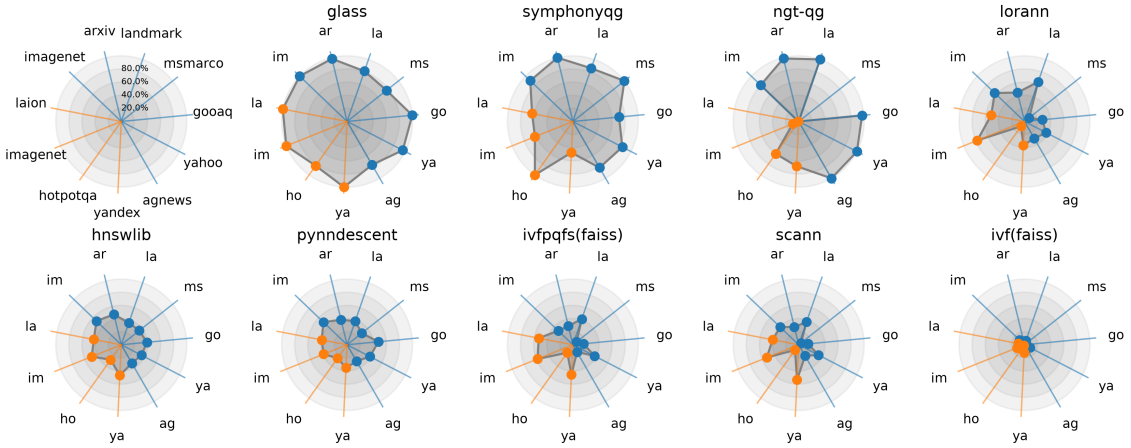


Figure 1: Relative query throughput of each algorithm's fastest configuration achieving an average recall of at least 95% on selected datasets with in-distribution and out-of-distribution queries, normalized by the throughput of the best algorithm on each dataset. Each circle corresponds to an algorithm, arranged by average normalized query throughput. Datasets are arranged in clockwise order by decreasing difficulty as measured by the median query relative contrast RC_{100} (He et al., 2012), with `imagenet-clip` being the easiest.

1 Introduction

In modern machine learning applications, data such as text and images are often represented by *embeddings* produced by neural encoders. Embeddings are typically dense, high-dimensional vector representations that are trained to have the property that relevant or semantically similar items have similar representations in the embedding space. This makes efficient similarity search in high-dimensional vector spaces an essential operation in applications where corpus items relevant to a query or user context are retrieved, such as information retrieval (Xiong et al., 2021; Izacard et al., 2022), recommendation systems (Covington et al., 2016; Yang et al., 2020), and question answering (Seo et al., 2019; Lewis et al., 2021).

Similarity search over vectors is commonly formulated as k-nearest neighbor (k-nn) search, which retrieves the k corpus vectors most similar to a query vector. Because embedding datasets are typically large and high-dimensional, vector indexes commonly use *approximate nearest neighbor* (ANN) *search* (Indyk and Motwani, 1998; Bruch, 2024) to answer these queries with low latency. An important recent use case is retrieval-augmented generation (RAG) (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2022; Wang et al., 2024).

Many important modern ANN workloads are *out-of-distribution* (OOD): the corpus and the queries have significantly different distributions. For instance, in multimodal search, the corpus may consist of images while the queries are expressed as text. Even in text retrieval, OOD workloads can arise from inherent differences between queries and documents, such as their length, as well as when instruction-tuned embedding models (Su et al., 2023) use different instructions to encode documents and queries.

Important OOD workloads also arise in maximum inner product search (MIPS). Approximate attention computation in LLMs can be formulated as a MIPS application (Bertsch et al., 2023; Liu et al., 2025; Mazaré et al., 2025), in which the task is to find the keys that have the largest inner products with the query. Another MIPS workload arises in reducing multi-vector retrieval to single-vector retrieval. Multi-vector embedding models such as ColBERT (Khattab and Zaharia, 2020) represent queries and documents as sets of embeddings and score them using the MaxSim similarity. Specialized retrieval engines (e.g., Santhanam et al., 2022a) can search these multi-vector representations directly, but single-vector MIPS reductions (Dhulipala et al., 2024; Jääsaari et al., 2026) of multi-vector retrieval are attractive because they allow the use of existing single-vector ANN infrastructure.

Because of the significance of ANN search in modern AI applications, accurate performance evaluation of vector indexes is essential. Currently, the ANN-Benchmarks project (Aumüller et al., 2020) includes the most comprehensive collection of ANN algorithms. However, most of its datasets, such as MNIST, Fashion-MNIST, SIFT, and GIST, are not representative of modern applications. These datasets have representations, such as raw pixels and image descriptors, that have been superseded by dense vector embeddings. In addition, existing systematic benchmarks do not cover important recent OOD use cases of ANN search. Hence, there is a need for a comprehensive, up-to-date benchmark suite containing both in-distribution embedding workloads and OOD workloads.

To address these issues, we introduce Vector Index Benchmark for Embeddings (VIBE), a benchmarking suite for ANN algorithms on modern embedding datasets. VIBE contains a straightforward pipeline for creating benchmark datasets: we use modern embedding models to generate representative embedding datasets from sources such as arXiv and ImageNet for both in-distribution (Section 4.1) and out-of-distribution (Section 4.2) settings. We also introduce four challenging OOD MIPS benchmark datasets covering approximate attention computation and multi-vector-to-single-vector reduction (Section 4.2). The benchmark is open, ongoing, and extensible, enabling authors of new ANN algorithms to easily integrate their implementations (Appendix A). Thus, beyond presenting an up-to-date performance evaluation, our work enables rigorous evaluation in the future. VIBE also features an interactive website that supports deeper analysis of the results (Appendix B).

We use VIBE to evaluate the performance of state-of-the-art ANN algorithms (Section 5). An overview of our evaluation for selected algorithms is presented in Figure 1. Our key findings, discussed further in Section 6.1, are as follows: (i) The top-performing graph-based (SymphonyQG, Glass, NGT-QG) and clustering-based (LoRANN, ScaNN, IVF-PQ) algorithms use various forms of quantization. (ii) On the text-retrieval and text-to-image OOD datasets, the best general-purpose methods still outperform specialized OOD methods (see Figure 7), somewhat surprisingly, even though the performance of these general-purpose methods deteriorates significantly on OOD queries (see Figure 8). (iii) On approximate attention computation datasets, general-purpose ANN methods fail to reach the highest recall levels, and specialized OOD methods outperform them (see Figure 10).

We summarize our main contributions below:

- **Benchmark for modern use cases.** VIBE is an open benchmarking framework for evaluating the performance of vector indexes on embedding datasets used in modern applications, such as retrieval-augmented generation. The framework also supports high-performance computing (HPC) environments and GPU-based implementations.

- **Extensible architecture.** VIBE contains a transparent pipeline for generating new benchmark datasets using embedding models, making the datasets easy to update. Its modular architecture makes it straightforward to add new algorithms (Appendix A).
- **Evaluation in the OOD setting.** We introduce novel OOD datasets spanning several recent use cases (Section 4.2). These datasets enable systematic comparisons between general-purpose ANN methods and OOD-specific methods (Section 5.3).
- **Comprehensive evaluation.** We use VIBE to conduct a performance evaluation of 22 vector index implementations across 11 in-distribution and 8 OOD datasets (Section 5). Our evaluation and visualization tools (Appendix B) support in-depth comparisons and reveal opportunities for future work (Section 6.1).
- **Fine-grained performance analysis.** Beyond recall–throughput curves, VIBE reports finer-grained performance breakdowns, including comparisons between easy and hard queries (Figure 5) and between in-distribution and OOD queries (Figure 8).

2 Background

2.1 ANN search

Definition Denote the m *corpus* points by $\{\mathbf{c}_1, \ldots, \mathbf{c}_m\} \subset \mathbb{R}^d$ and let $\rho\colon \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be a dissimilarity measure. The task of k-nearest neighbor (k-nn) search is to retrieve the indices of the k corpus points that are most similar to the *query point* $\mathbf{x} \in \mathbb{R}^d$, i.e., to retrieve

$$\mathrm{NN}_k(\mathbf{x}) = \{\pi(1), \ldots, \pi(k)\}, \qquad \rho(\mathbf{x}, \mathbf{c}_{\pi(1)}) \leq \cdots \leq \rho(\mathbf{x}, \mathbf{c}_{\pi(m)}),$$

where π is a permutation of $[m] = \{1, \ldots, m\}$, with ties broken according to a fixed rule. The task of *approximate nearest neighbor* (ANN) *search* is to design a *vector index* that enables approximating the exact k-nn solution in sublinear time. In this paper, we use *approximate* to refer to *inexact solutions* without guarantees (Aumüller and Ceccarello, 2023).

Dissimilarity measures The most common dissimilarity measures for ANN search are the Euclidean distance $\rho(\mathbf{a}, \mathbf{b}) = \|\mathbf{a} - \mathbf{b}\|_2$, the cosine distance $\rho(\mathbf{a}, \mathbf{b}) = 1 - \langle \mathbf{a}/\|\mathbf{a}\|_2, \mathbf{b}/\|\mathbf{b}\|_2 \rangle$, and the negative inner product $\rho(\mathbf{a}, \mathbf{b}) = -\langle \mathbf{a}, \mathbf{b} \rangle$. Minimizing the negative inner product is equivalent to *maximum inner product search* (MIPS). Embeddings are commonly normalized to unit norm, in which case all three dissimilarity measures yield the same nearest neighbors.

Performance measures For a query point, *recall* is the fraction of its k returned neighbors that are among its true k nearest neighbors. We measure the effectiveness of ANN algorithms by the *average recall*, obtained by averaging this per-query recall over all test queries. The efficiency of ANN algorithms is typically measured by the average query latency or, equivalently, by throughput, which is measured by *queries per second* (QPS).

Difficulty measures To assess the difficulty of a query $\mathbf{x}$, we consider its *relative contrast* at k, $RC_k(\mathbf{x}) = \frac{1}{\rho(\mathbf{x}, \mathbf{c}_{\pi(k)})} \sum_{j \in [m]} \rho(\mathbf{x}, \mathbf{c}_j)/m$, that is, the ratio of the average distance between the query and all the corpus points in the dataset to the distance between the query and its k-th nearest neighbor (He et al., 2012). The relative contrast is a good proxy for the effort required to distinguish nearby and distant points (Aumüller and Ceccarello, 2021; Ceccarello et al., 2025), with small values characterizing difficult queries.

Out-of-distribution (OOD) setting There is recent interest in ANN search in the *out-of-distribution* (OOD) setting (e.g., Cayton and Dasgupta, 2007; Jaiswal et al., 2022; Hyvönen et al., 2024; Chen et al., 2024; Jääsaari et al., 2024; Mazaré et al., 2025), in which the corpus points and query points are drawn from different distributions. If there is a large difference between the query distribution and the corpus distribution, using a representative sample from the query distribution to adapt the index during construction can speed up ANN search significantly (Jaiswal et al., 2022; Hyvönen et al., 2024; Chen et al., 2024).

The OOD setting is common in multimodal search, where queries and corpus points have different modalities: for example, the queries may be text while the corpus consists of images, with both mapped into a shared embedding space. ANN search has also been used to accelerate attention computation in long-context LLM inference, in which the distributions of the keys and queries differ significantly (Liu et al., 2025; Mazaré et al., 2025). Asymmetric reductions for multi-vector retrieval can likewise produce OOD single-vector MIPS workloads by mapping queries and documents differently (Dhulipala et al., 2024; Jääsaari et al., 2026). VIBE includes datasets for all of these out-of-distribution settings (Section 4.2).

2.2 Taxonomy of ANN algorithms

Broadly, approximate nearest neighbor search algorithms can be classified into graph-, clustering-, tree-, and hashing-based methods according to the index structure they use. We give a brief overview of each category below. VIBE includes methods from all four categories; Table 3 summarizes the algorithms included in the benchmark, and Table 7 in Appendix D provides citations and implementation details.

Graphs Graph-based methods, such as HNSW (Malkov and Yashunin, 2020), NSG (Fu et al., 2019), Vamana (Subramanya et al., 2019), and SymphonyQG (Gou et al., 2025), use a navigable proximity graph of the corpus as an index structure and retrieve the approximate nearest neighbors of the query point via greedy or bounded best-first graph traversal.

Clustering Clustering-based methods cluster the corpus points and evaluate the distances from the query point only to the corpus points in the clusters closest to the query point. The points in the selected clusters can be ranked using an approximate score-computation method, such as product quantization (PQ) in IVF-PQ (Jégou et al., 2011), anisotropic vector quantization in ScaNN (Guo et al., 2020), or reduced-rank regression in LoRANN (Jääsaari et al., 2024), to select a candidate set that may subsequently be reranked using exact scores.

Trees Tree-based methods use space-partitioning index structures, including k-d trees (Friedman et al., 1977), principal axis trees (Sproull, 1991), and random projection trees (Dasgupta and Freund, 2008). Efficient tree-based methods, such as Annoy (Spotify, 2013) and MRPT (Hyvönen et al., 2016), use ensembles of randomized trees.

Hashing Hashing-based methods, such as PUFFINN (Aumüller et al., 2019) and FALCONN++ (Pham and Liu, 2022), partition the space using locality-sensitive hashing or filtering schemes and evaluate the distances from the query point to only those corpus points stored in buckets probed for the query. Hashing-based methods often use several randomized hash tables to improve the search accuracy, and these approaches can provide strong probabilistic correctness guarantees.

3 Related work

The ANN-Benchmarks project [1] (Aumüller et al., 2020) is the most well-known and comprehensive benchmarking suite for vector indexes. However, its datasets are no longer representative of the datasets used in modern applications of ANN search, and the project does not contain a systematic evaluation of the OOD setting. Earlier performance evaluations (Li et al., 2020; Aumüller et al., 2020) do not contain modern embedding datasets or the recent top-performing ANN methods. More recent performance evaluations (Wang et al., 2021; Azizi et al., 2025) consider only graph-based ANN methods and do not provide an open and extensible benchmarking framework.

The big-ann-benchmarks project (Simhadri et al., 2022, 2025) has single-dataset tracks for benchmarking ANN algorithms on four specialized tasks, consisting of filtered search, streaming search, sparse search, and OOD search. Other recent studies focus specifically on streaming search (Zeng et al., 2024; Wang et al., 2026) or filtered search (Iff et al., 2026; Lim et al., 2026; Zhang et al., 2026).

Chen et al. (2026) benchmark vector indexes from the perspective of end-to-end task performance. Some vector database companies have their own benchmarks, such as VectorDBBench [2], and Kang et al. (2025) likewise compare vector database systems using an integrated benchmark that covers embedding latency and compound queries. These production-oriented systems combine a vector index with database and deployment functionality, and such benchmarks therefore address the complementary question of end-to-end system performance. In contrast, VIBE isolates open-source ANN implementations to support reproducible, algorithm-level comparisons.

4 VIBE

This section describes the datasets included in VIBE. It also presents features that support future research, including quantized datasets and broad hardware support.

4.1 In-distribution datasets

The in-distribution datasets included in VIBE are listed in Table 1. We create embedding datasets using widely used models applied to common text and image data sources. We choose text embedding models that include the older but widely used models DistilRoBERTa (Sanh et al., 2019; Liu et al., 2019; Reimers and Gurevych, 2019) and MiniLM (Wang et al., 2020; Reimers and Gurevych, 2019), as well as recent top-performing (Muennighoff et al., 2023) industry models, including nomic-embed-text-v1.5 (Nussbaum et al., 2025), mxbai-embed-large-v1 (Lee et al., 2024), jina-embeddings-v5-text-nano (Akram et al., 2026), and Qwen3-Embedding-0.6B (Zhang et al., 2025). For images, we similarly use embeddings from a ResNet-50 convolutional neural network (He et al., 2016), as well as modern image embedding models: CLIP (Radford et al., 2021), DINO (Caron et al., 2021), and nomic-embed-vision-v1.5 (Nussbaum et al., 2024). We also include the older GloVe (Pennington et al., 2014) word embedding dataset due to its prevalence in earlier benchmarks.

1. `https://github.com/erikbern/ann-benchmarks`
2. `https://github.com/zilliztech/VectorDBBench`

Table 1: VIBE in-distribution datasets. For references and details about the data sources and embedding models, see Appendix A.9. Embeddings with dissimilarity measure "any" are normalized to unit ℓ_2 norm, and each algorithm can choose an appropriate measure. Here m = number of corpus points, d = embedding dimension.

Data Source	Model	Type	m (↓)	d	Measure
DPR	Jina	Text	20 969 760	768	any
MS MARCO	Qwen	Text	8 840 823	1024	any
GooAQ	DistilRoBERTa	Text	1 475 024	768	any
arXiv	Nomic Text	Text	1 344 643	768	any
ImageNet	CLIP	Image	1 281 167	512	any
Twitter	GloVe	Word	1 192 514	200	cosine
AGNews	MXBAI	Text	769 382	1024	Euclidean
Landmark	DINO	Image	760 757	768	cosine
Landmark	Nomic Vision	Image	760 757	768	any
Yahoo	MiniLM	Text	677 305	384	any
iNaturalist	ResNet	Image	499 000	2048	cosine

To select the benchmark datasets, we start with this broad pool of widely used embedding models spanning different model families and output dimensionalities. For each model, we choose suitable public data sources (see Appendix A.9) whose document lengths fit the model's context window. We then select a small subset of model–dataset combinations by using relative contrast (see Section 2.1) to guide selection toward a diverse set of datasets, while manually ensuring that the final collection includes both text and image datasets, different model families, and a broad range of output dimensionalities.

Most of the resulting datasets contain on the order of one million corpus vectors, which keeps comprehensive evaluations across datasets, methods, and hyperparameters practical and allows using a common hyperparameter grid for each method throughout. We also include two larger datasets to help evaluate method performance with increasing corpus size.

The datasets span a wide range of difficulty, as quantified by relative contrast. Figure 2 summarizes the distribution of relative contrast values across different datasets. Visually, datasets stemming from image embeddings have a wider range of relative contrast values than those stemming from text embeddings.

For most text embeddings, the model outputs are normalized to unit norm by default. In this case, cosine similarity, Euclidean distance, and inner product induce the same nearest-neighbor ranking, and each algorithm can choose an appropriate distance measure. Otherwise, we use Euclidean distance for text embeddings to represent a clustering task. For image embeddings that are not normalized, we use cosine distance. For each in-distribution dataset, we hold out 1000 points as test queries and use the remaining points as the corpus.

4.2 Out-of-distribution datasets

We choose the OOD datasets to cover a diverse array of retrieval tasks. The eight OOD datasets comprise one text-retrieval dataset, three text-to-image retrieval datasets, two

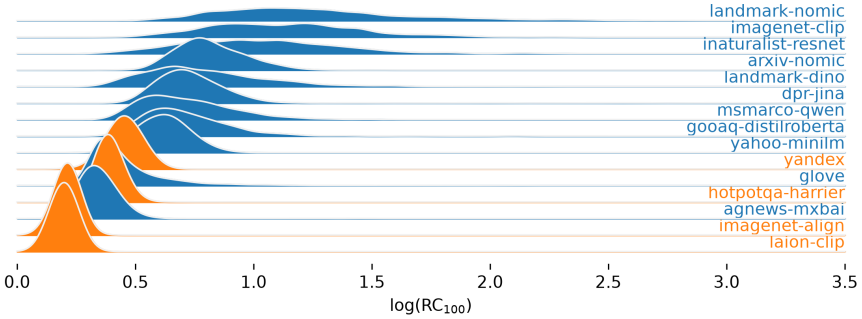


Figure 2: Distribution of the relative contrast query difficulty metric RC_{100} for selected in-distribution and out-of-distribution datasets. (Note the logarithmic scale.) Density curves are arranged by their median relative contrast. Lower relative contrast corresponds to a harder nearest neighbor search problem, while a flatter curve represents greater variation in query difficulty. In general, queries in image embedding datasets tend to be easier and span a wider range of difficulty than those in text embedding datasets, while OOD datasets pose harder nearest neighbor search problems than in-distribution datasets. Inner product datasets are omitted because relative contrast is not meaningful for negative inner product dissimilarity.

single-vector MIPS datasets obtained by reducing multi-vector retrieval, and two MIPS datasets derived from approximate attention workloads (Table 2).

We create the HotpotQA–Harrier text-retrieval dataset from HotpotQA (Yang et al., 2018) using the harrier-oss-v1-270m model (Microsoft, 2026). Its queries are out-of-distribution because the Harrier model applies a retrieval prompt only to queries, which are also typically short questions, whereas the documents are longer passages. For ImageNet–ALIGN, we use the ALIGN model (Jia et al., 2021) to embed ImageNet images as corpus vectors and text from the ImageNet-Captions dataset (Fang et al., 2022) as query vectors. We also include subsets of the existing LAION and Yandex text-to-image datasets previously used to benchmark OOD ANN algorithms (Simhadri et al., 2025; Chen et al., 2024).

For the approximate attention workloads, we construct two datasets by extracting keys and queries from one selected attention head in each of two long-context language models: Yi-6B-200K and Llama-3-8B-Instruct-262k. These attention workloads are out of distribution because the models compute keys and queries using separate learned query and key projections (Liu et al., 2025; Mazaré et al., 2025). For the multi-vector retrieval workloads, we apply MUVERA (Dhulipala et al., 2024) and LEMUR (Jääsaari et al., 2026) to 128-dimensional ColBERTv2 (Santhanam et al., 2022b) token embeddings of CQADupStack (Hoogeveen et al., 2015). These methods reduce multi-vector similarity search to single-vector MIPS, producing 5120-dimensional MUVERA vectors and 2048-dimensional LEMUR vectors. The resulting datasets are OOD because these reductions use different mappings for queries and documents. For details, see Appendix A.9.

For each OOD dataset, we include a larger training sample from the query distribution, allowing query-distribution-aware methods (Hyvönen et al., 2024; Chen et al., 2024; Jääsaari

Table 2: VIBE out-of-distribution datasets. For details about the data sources and embedding models, see Appendix A.9. Embeddings with dissimilarity measure "any" are normalized to unit norm, and each algorithm can choose an appropriate measure. The two CQADupStack reductions and the `llama` and `yi` datasets are MIPS workloads and use (negative) inner product. Here m = number of corpus points, m_{learn} = number of query distribution samples, d = embedding dimension.

Data Source	Model	Type	m (↓)	m_{learn}	d	Measure
HotpotQA	Harrier	Text	5 233 329	96 845	640	any
ImageNet	ALIGN	Text-To-Image	1 281 167	315 648	640	any
LAION	CLIP	Text-To-Image	1 000 448	999 448	512	any
Yandex	SE-ResNeXt	Text-To-Image	999 000	999 000	200	cosine
CQADupStack	MUVERA	Multi-vector	457 149	1 331 947	5120	IP
CQADupStack	LEMUR	Multi-vector	457 149	1 331 947	2048	IP
Llama-3-8B	-	Attention	256 921	255 921	128	IP
Yi-6B-200K	-	Attention	187 843	186 843	128	IP

et al., 2024) to use information about the query distribution during index construction. Some of these methods additionally require the exact nearest neighbors of the training queries in the corpus as input. We precompute and distribute these neighbors with the datasets.

For each OOD dataset, we use 1000 held-out test queries drawn from its query distribution. We quantify the distribution shift between each dataset's corpus and query distributions in Appendix C; each dataset exhibits substantial distribution shift.

4.3 Quantized datasets

State-of-the-art embeddings are increasingly trained to perform well when quantized to 8-bit integer or even binary precision due to their high computational and storage costs [3]. For example, distance computations using the Hamming distance on binarized data are much faster to evaluate using bitwise XOR and specialized population-count instructions, such as those provided by the AVX-512 VPOPCNTDQ extension. VIBE provides quantized variants of embedding datasets and functions for creating them; a subset of the default algorithms supports these variants (see Appendix F.3).

4.4 Support for research

VIBE features additional components that support future research on vector indexes. Its evaluation pipeline builds on the evaluation code of ANN-Benchmarks (Aumüller et al., 2020) and extends it to support a broader range of hardware platforms. Dropping the dependency on Docker, the benchmark pipeline supports high-performance computing (HPC) platforms through Apptainer (Singularity) containerization (Kurtzer et al., 2017) and NUMA-aware execution. The benchmark also supports GPU implementations (see Figure 6, right panel).

3. See, e.g., `https://huggingface.co/blog/embedding-quantization`

Table 3: Summary of the evaluated algorithms. '*' marks implementations that have not been included in earlier benchmarks. See Table 7 in Appendix D for citations and implementation details.

Graphs		Trees	Clustering	Hashing
Glass	PyNNDescent	ANNOY	IVF	PUFFINN
NSG	HNSW	MRPT	IVF-PQ	FALCONN++*
NGT-QG	HNSW-RaBitQ*	MLANN*	IVF-RaBitQ*	
Vamana	FlatNav*		ScaNN	
LVQ*	LeanVec*		LoRANN*	
RoarGraph*	SymphonyQG*			

The benchmark features a companion website[4] that provides interactive plots (see Appendix B) to further explore benchmark results and dataset characteristics. In particular, the website allows users to compare algorithms at different recall levels and interactively examine every tested configuration, including configurations that do not lie on the Pareto frontier. The companion website and the benchmark suite are designed to make it easy to add new datasets and algorithms (see Appendix A).

5 Performance evaluation

5.1 Experimental setup

Each experiment is run on a compute node with 384 GB of RAM and two Intel Xeon Gold 6230 (Cascade Lake) CPUs. Each CPU has 20 cores and supports AVX-512 instructions. All algorithms are benchmarked using a single worker thread on one physical CPU core, with simultaneous multithreading (SMT) disabled, and we use `numactl` to bind the algorithm execution to one CPU socket and constrain memory allocation to its associated local NUMA node. All of the compared implementations are written in C++ with the exception of PyNNDescent (Dong et al., 2011; McInnes, 2018), which uses the Numba library extensively for just-in-time compilation. All implementations are heavily optimized and use techniques such as vectorization to accelerate distance computations.

By default, we use $k = 100$ in all experiments because it is representative of real-world applications: retrieval systems commonly retrieve more candidates than they ultimately return, allowing the candidates to be filtered, diversified, or reranked using more expensive models. We use average recall to measure the effectiveness and queries per second (QPS) to measure the efficiency of the algorithms. Each query hyperparameter combination is run five times, and we use the minimum query time across these runs to compute QPS.

Algorithms We review the recent literature and the earlier benchmarking efforts (e.g., Aumüller et al., 2020) to identify the top-performing ANN methods for our performance evaluation. The inclusion criteria are as follows: (i) an implementation of the method is available as a library with Python bindings; (ii) the library is open source; (iii) the method

4. `https://vector-index-bench.github.io/`

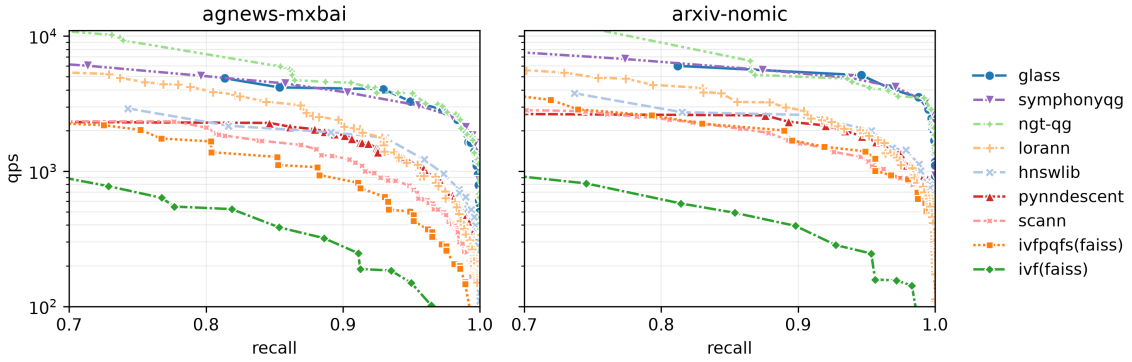


Figure 3: Recall/throughput trade-off on two **text embedding** datasets (in-distribution queries). Among the methods shown, the graph-based methods Glass, NGT-QG, and SymphonyQG achieve the highest throughput.

is top-performing within its class or is widely used in practice. See Table 3 for the included methods and Table 7 in Appendix D for citations and implementation details.

Hyperparameter settings Most implementations expose several hyperparameters whose optimal values depend on the workload, and VIBE allows for easy configuration of a grid search for these hyperparameters. Initial choices for older methods were obtained from the hyperparameter grids of the ANN-Benchmarks project (Aumüller et al., 2020) and from the library documentation and the corresponding research papers for new methods. We then expanded these grids as necessary based on intermediate results.

In the figures in this section, we report the Pareto frontiers over all evaluated configurations. To keep the comparisons consistent and avoid dataset-specific hyperparameter tuning, we use the same hyperparameter grid for every dataset. The evaluated hyperparameter grids for all methods are available in our code repository [5].

5.2 In-distribution setting

We compare the ANN algorithms on eleven in-distribution datasets. Figure 1 presents an overview for seven representative in-distribution datasets. For selected algorithms, the recall–QPS curves are reported in Figure 3 for two representative text-embedding datasets and in Figure 4 for two representative image-embedding datasets. The results for all datasets can be found in Appendix F.1. For clarity, the figures in this section show only nine selected methods. Results for all methods are available on the companion website, while Appendix F.2 presents full results for two datasets. Appendix G additionally summarizes the recall–throughput trade-offs using normalized hypervolume.

The general trend is that graph-based and clustering-based methods significantly outperform hashing-based and tree-based methods. At 95% average recall, the highest-throughput method on every dataset is one of the graph-based methods Glass, SymphonyQG, or NGT-QG. However, as shown in Appendix E, graph-based methods generally have the disadvantage of longer index construction times.

5. https://github.com/vector-index-bench/vibe/

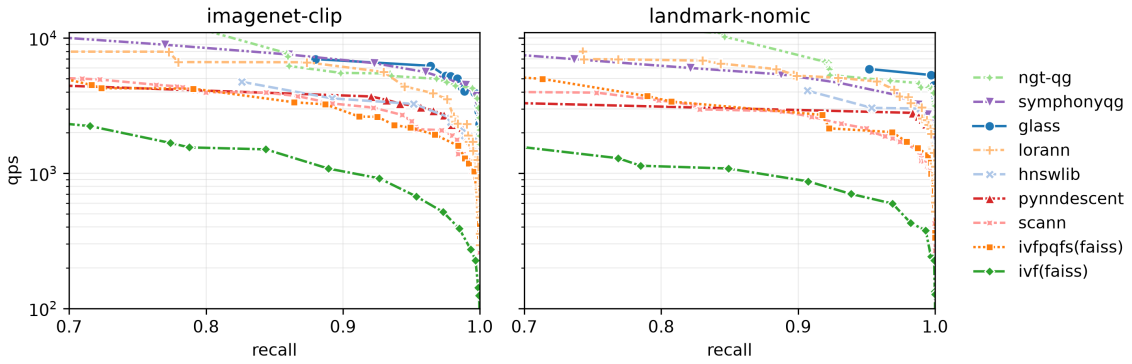


Figure 4: Recall/throughput trade-off on two **image embedding** datasets (in-distribution queries). The graph-based methods Glass, NGT-QG, and SymphonyQG, and the clustering-based method LoRANN achieve the highest throughput.

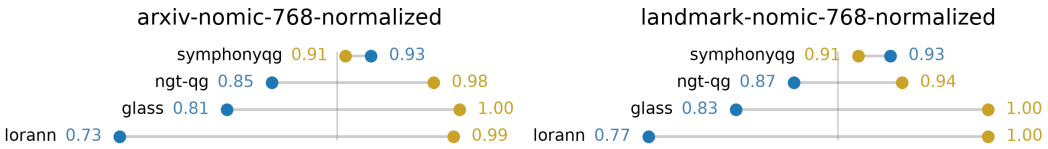


Figure 5: Average recall over the 100 hardest and easiest queries (by the RC_{100} metric) on two datasets, using each algorithm's fastest configuration achieving at least 90% average recall over all test queries (threshold marked by the vertical line). SymphonyQG is robust with respect to query difficulty, while Glass and LoRANN show greater variability.

Robustness to query difficulty The performance difference between the 100 hardest and the 100 easiest queries, as measured by relative contrast RC_{100}, is shown in Figure 5 for the four top-performing algorithms, using the fastest configurations achieving at least 90% average recall over all test queries. Glass and LoRANN achieve nearly 100% average recall on the easiest queries. On the hardest queries, Glass averages 81% recall on `arxiv-nomic` and 83% on `landmark-nomic`, while LoRANN averages 73% and 77%, respectively. SymphonyQG is robust, with a smaller easy–hard average-recall gap than Glass, LoRANN, and NGT-QG.

Higher throughput with binary datasets and GPU deployment VIBE includes support for binary datasets (with Hamming distance) and GPU algorithms (see Section 4.4). Both settings can provide higher throughput. For example, on the float `agnews-mxbai` dataset at 90% average recall, the best-performing method achieves a throughput of 4×10^3 QPS (Figure 3), whereas on the binarized version of the same dataset at 90% average recall [6], the best-performing method achieves a throughput of 8×10^3 QPS (Figure 6, left panel).

Using an NVIDIA V100 GPU, the best GPU algorithm achieves a throughput of 10^5 QPS when all 1000 test queries are processed in a single batch (Figure 6, right panel).

6. However, note that for binary datasets, 90% average recall means recovering 90% of the exact Hamming nearest neighbors, so the recall levels are not directly comparable.

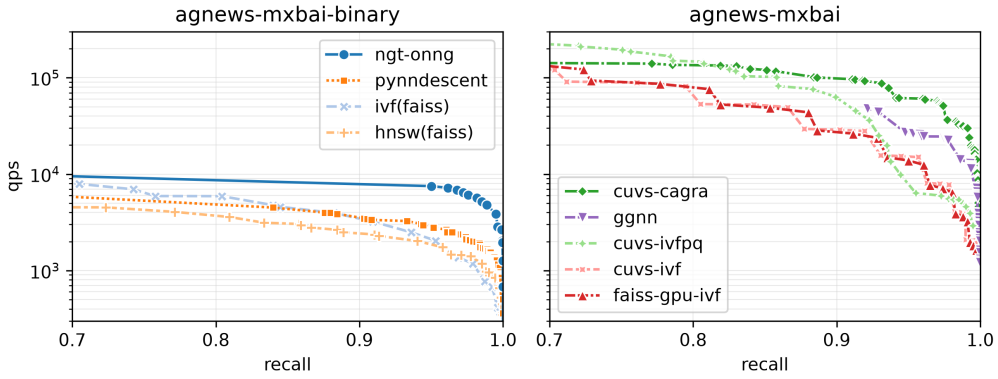


Figure 6: Left: recall/throughput trade-off on **binary data**. Right: recall/throughput trade-off for **GPU** algorithms. The graph-based NGT-ONNG has the highest throughput on the binary data, and the graph-based CAGRA is the fastest method in the GPU setting.

More results are shown in Appendix F.3: on binary datasets, the graph-based method NGT-ONNG (Iwasaki and Miyazaki, 2018) is the fastest method, while the graph-based CAGRA (Ootomo et al., 2024) is the fastest GPU method.

5.3 Out-of-distribution setting

We compare ANN algorithms on eight out-of-distribution (OOD) datasets. Figure 1 presents an overview of selected results at 95% average recall. The results for one text-retrieval dataset and one text-to-image dataset are shown in Figure 7. Representative results for the two MIPS applications are shown in Figures 9 and 10.

Performance on text and text-to-image retrieval On the text-retrieval and text-to-image OOD datasets, the most efficient general-purpose ANN methods outperform the ANN methods specifically tailored for OOD data (see Figure 7). The observed advantage of general-purpose methods may partly reflect differences in implementation maturity and optimization rather than index design alone (Kriegel et al., 2017). However, as can be seen from Figure 8, the performance of the general-purpose ANN algorithms degrades significantly on OOD queries compared to in-distribution queries (see Appendix H for additional results).

With in-distribution queries, a query typically lies near the indexed data cloud, and its nearest neighbors tend to be close together. An OOD query can instead be far from the corpus distribution, and its nearest corpus points can be far from both the query and one another. Reaching the same recall then requires more partition probes or longer graph traversal. The extent of the slowdown varies across methods because different routing and pruning rules respond differently to distribution shift, while approximate distance estimation schemes can lose accuracy by varying amounts. For example, if a quantizer is fitted to the corpus, for shifted queries, the quantized vectors can produce larger errors in the estimated candidate distances, steering graph traversal along less useful paths and increasing the number of examined candidates (Gou et al., 2025).

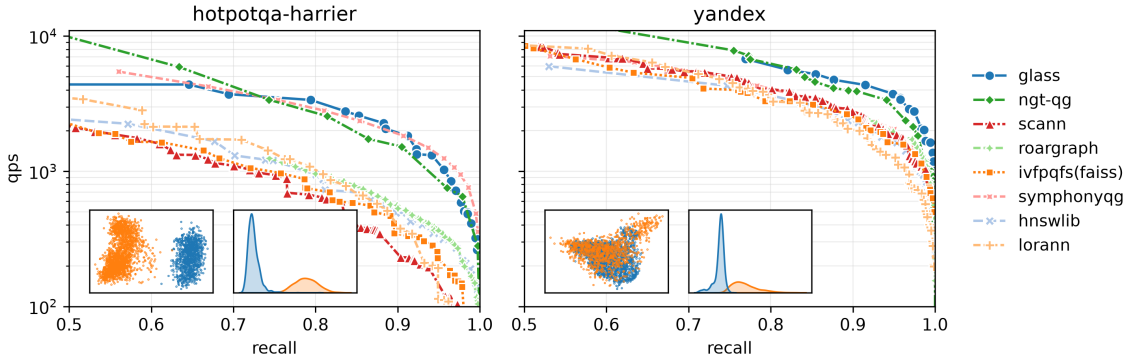


Figure 7: Recall/throughput trade-off on the **text-retrieval** `hotpotqa` dataset (left) and the **text-to-image** `yandex` dataset (right), both with **out-of-distribution** queries. The left insets show PCA projections of the corpus points and the queries, and the right insets show the Mahalanobis-distance distributions for the corpus and query points. Less PCA overlap and a larger rightward shift of the query distance distribution indicate a stronger distribution shift. At high average recall, the general-purpose graph-based methods Glass, NGT-QG, and SymphonyQG achieve the highest throughput among the methods shown on both datasets, outperforming the evaluated OOD-specific methods, including RoarGraph.

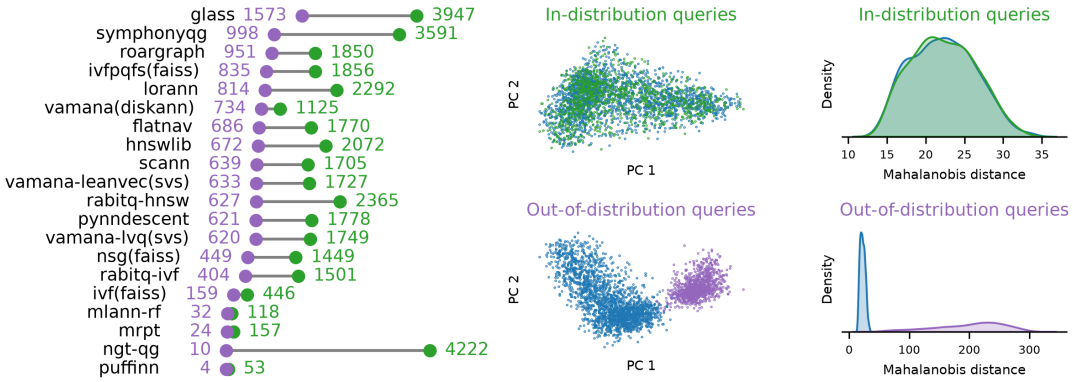


Figure 8: *Left:* Throughput (QPS) for out-of-distribution and in-distribution queries on the `laion-clip` dataset at 95% average recall; *middle:* PCA projections comparing the corpus with in-distribution queries (top) and out-of-distribution queries (bottom); *right:* the corresponding Mahalanobis distance distributions. Throughput degrades on the OOD queries for all methods shown.

Multi-vector reduction Figure 9 shows results for the `lemur` and `muvera` workloads. Compared with the text-to-image workloads, methods generally need to scan substantially more candidates on these datasets, resulting in much slower search, and many methods struggle to reach the highest recall levels with the common hyperparameter settings used

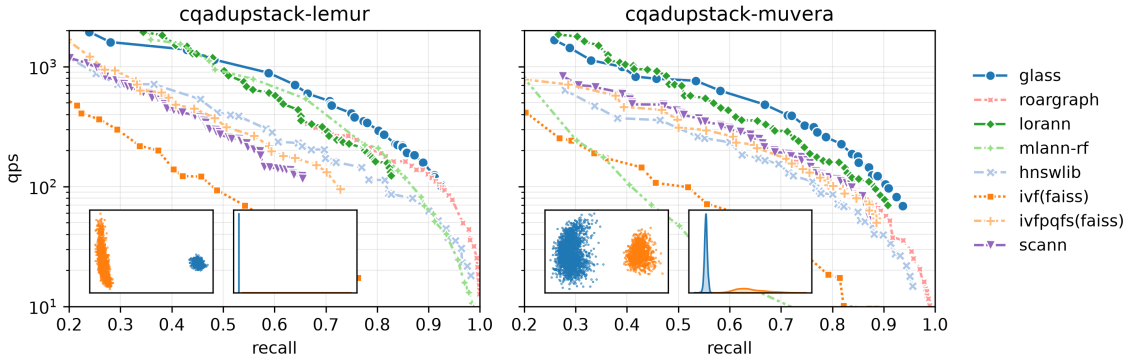


Figure 9: Recall/throughput trade-off on the **multi-vector-to-single-vector reduction** MIPS datasets `cqadupstack-lemur` (left) and `cqadupstack-muvera` (right), both with **out-of-distribution** queries. Methods generally need to scan substantially more candidates on the multi-vector reduction datasets than on the regular datasets, and many methods struggle to reach the highest recall levels with the common hyperparameter settings used across datasets.

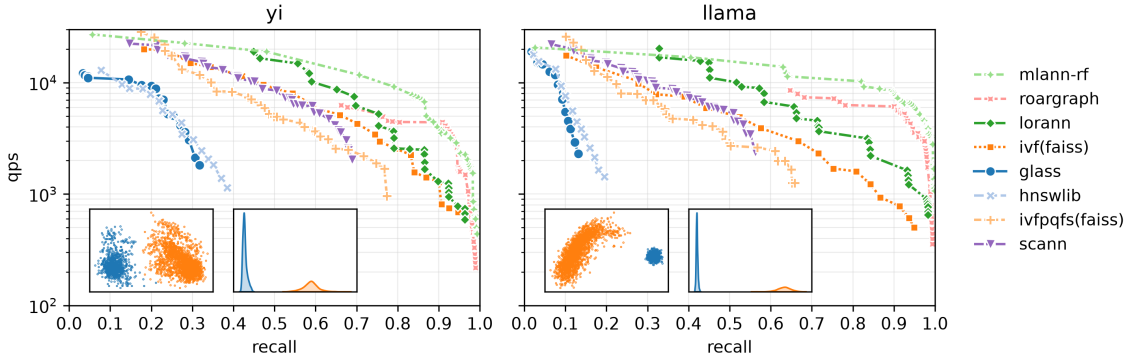


Figure 10: Recall/throughput trade-off on the **approximate attention computation** datasets `yi` (left) and `llama` (right), both with **out-of-distribution** queries. The specialized OOD methods MLANN (Random Forest), RoarGraph, and LoRANN outperform the regular ANN methods, such as Glass, that struggle to reach the highest recall levels.

across datasets. RoarGraph reaches the highest recall levels and outperforms Glass on `lemur`, but still underperforms Glass on `muvera`. The OOD method MLANN is competitive on `lemur`, but substantially underperforms the other methods on `muvera`.

Approximate attention computation General-purpose ANN methods struggle on the approximate attention computation datasets (see Figure 10 for the `yi` and `llama` workloads): for example, the graph methods Glass and hnswlib do not reach 50% average recall under any evaluated configuration. Clustering-based methods perform relatively better, but ScaNN and IVF-PQ perform worse than IVF. This is consistent with the results on the text-to-image

datasets: their corpus-trained quantizers may distort query–candidate scores more severely under more pronounced distribution shift. In contrast, methods that are specifically designed for OOD data, namely RoarGraph, MLANN (Random Forest), and LoRANN, reach the highest recall levels and achieve the best performance.[7] The performance difference between the OOD methods and the general-purpose ANN methods is more pronounced on the `llama` dataset than on the `yi` dataset. The difference between the corpus and query distributions is also larger on `llama` than on `yi` (see Appendix C).

5.4 Significance tests

To assess the robustness of our results, for each dataset we select each algorithm's fastest configuration reaching at least 95% average recall and compare algorithms using their paired per-query latencies over the 1000 test queries. We test all pairwise differences using Wilcoxon signed rank tests (Wilcoxon, 1945) with the Holm-Bonferroni correction (Holm, 1979). Almost all (2399 out of 2450 pairwise comparisons [8] of algorithms reaching 95% average recall) of the differences are statistically significant at the significance level $\alpha = 0.01$. The critical difference diagrams (Demšar, 2006) visualizing the results of the significance tests for `agnews-mxbai` and `arxiv-nomic` can be found in Figure 11.

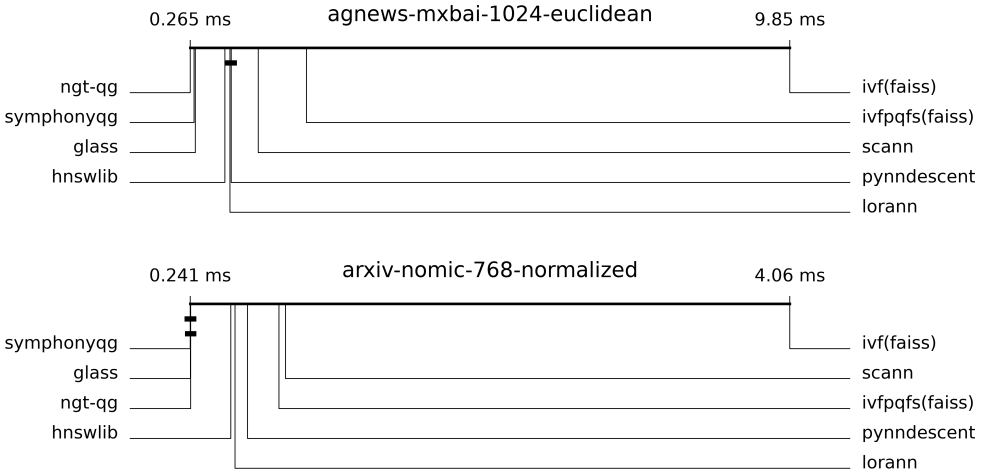


Figure 11: Critical difference diagrams visualizing the results of the significance tests for the `agnews-mxbai` and `arxiv-nomic` datasets. The bold horizontal black bars connect the algorithms whose paired per-query latency differences are not statistically significant at the level $\alpha = 0.01$ (with Holm-Bonferroni correction).

7. Recall may not be the most appropriate quality metric for approximate attention computation because the relevant objective is the error in the resulting attention output rather than exact recovery of the top-k keys (Chen et al., 2025). We report recall here to maintain consistency with the other ANN workloads.
8. There are in total 2450 possible pairwise comparisons when we add up all the comparisons for both the in-distribution and the out-of-distribution datasets. We apply the Holm-Bonferroni correction over all of these 2450 comparisons.

6 Discussion

We introduce VIBE, a benchmark for evaluating ANN algorithms on modern embedding workloads with both in-distribution and OOD queries. Our findings demonstrate the importance of evaluating ANN algorithms across diverse, representative embedding workloads.

6.1 Key findings

Comparison across method classes The graph-based methods Glass and SymphonyQG are the top-performing implementations. SymphonyQG is also the most robust method with respect to query difficulty (see Figure 5). Graph-based and clustering-based indexes outperform both tree-based and hashing-based indexes.

Reduced precision All of the top-performing graph-based methods (SymphonyQG, Glass, NGT-QG) and clustering-based methods (LoRANN, ScaNN, Faiss-IVF-PQ) use different types of quantization in at least some of their computations. SymphonyQG uses RaBitQ (Gao and Long, 2024), while NGT-QG, ScaNN, and Faiss-IVF-PQ employ efficient 4-bit PQ implementations (André et al., 2017). LoRANN uses 8-bit scalar quantization to speed up vector-matrix multiplications, and Glass, an efficient HNSW implementation with scalar quantization, significantly outperforms the full-precision HNSW implementation.

OOD performance gap In the out-of-distribution (OOD) setting, two key trends emerge: (1) On the approximate attention computation MIPS datasets, where the difference between the corpus and query distributions is the largest, general-purpose ANN methods struggle to reach the highest recall levels, and methods specifically tailored for the OOD setting (MLANN, RoarGraph, LoRANN) outperform them. (2) However, on the text-retrieval and text-to-image OOD datasets, our results suggest that while the performance of the general-purpose ANN methods degrades in the OOD setting, the current OOD-specific methods do not yield significant performance improvements over general-purpose methods.

6.2 Limitations and future work

We do not record internal algorithm statistics, such as the number of corpus points accessed or distance computations performed, since most implementations do not expose such counters. Although VIBE measures index size, we do not report it here because implementations differ in whether they copy corpus vectors into the index, making the results incomparable.

All experiments in the paper are conducted on a single hardware platform. Because performance can depend on the interaction between CPU microarchitecture and index access patterns (Kuffo and Boncz, 2025), relative implementation performance may differ on other hardware. Repeating the evaluation across multiple hardware architectures would therefore be valuable for assessing the generalizability of our findings.

Due to the high dimensionality of embeddings, we focus on million-scale datasets. This scale keeps the computational requirements manageable while allowing us to compare many algorithms across a broad range of embeddings. However, evaluating the performance of ANN algorithms on larger, even billion-scale, datasets to determine whether our key findings also hold in this regime is an interesting direction for future research.

Broader Impact Statement

The article is foundational research that focuses on evaluating ANN search algorithms. ANN search is a primitive that is used as a component in machine learning pipelines. Measuring the computational cost associated with the choice of algorithm has the potential to reduce the carbon emissions of large-scale deployments. Beyond this potential positive environmental impact, our work has no direct societal impact.

Acknowledgments and Disclosure of Funding

This work has been supported by the Research Council of Finland (grant #361902 and the Flagship programme: Finnish Center for Artificial Intelligence FCAI) and the Jane and Aatos Erkko Foundation (BioDesign project, grant #7001702). Martin Aumüller received funding from the Innovation Fund Denmark for the project DIREC (9142-00001B). The authors acknowledge the research environment provided by ELLIS Institute Finland, as well as CSC – IT Center for Science, Finland, for computational resources.

References

Cecilia Aguerrebere, Ishwar Singh Bhati, Mark Hildebrand, Mariano Tepper, and Theodore L. Willke. Similarity search in the blink of an eye with compressed indices. *Proceedings of the VLDB Endowment*, 16(11):3433–3446, 2023.

Mohammad Kalim Akram, Saba Sturua, Nastia Havriushenko, Quentin Herreros, Michael Günther, Maximilian Werk, and Han Xiao. jina-embeddings-v5-text: Task-targeted embedding distillation. *arXiv preprint arXiv:2602.15547*, 2026.

Fabien André, Anne-Marie Kermarrec, and Nicolas Le Scouarnec. Accelerated nearest neighbor search with Quick ADC. In *Proceedings of the 2017 ACM International Conference on Multimedia Retrieval*, pages 159–166, 2017.

Martin Aumüller and Matteo Ceccarello. The role of local dimensionality measures in benchmarking nearest neighbor search. *Information Systems*, 101:101807, 2021.

Martin Aumüller and Matteo Ceccarello. Recent approaches and trends in approximate nearest neighbor search, with remarks on benchmarking. *IEEE Data Engineering Bulletin*, 47(3):89–105, 2023.

Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Michael Vesterli. PUFFINN: Parameterless and universally fast finding of nearest neighbors. In *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 10:1–10:16, 2019.

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

Ilias Azizi, Karima Echihabi, and Themis Palpanas. Graph-based vector search: An experimental evaluation of the state-of-the-art. *Proceedings of the ACM on Management of Data*, 3(1):1–31, 2025.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Amanda Bertsch, Uri Alon, Graham Neubig, and Matthew R. Gormley. Unlimiformer: Long-range transformers with unlimited length input. *Advances in Neural Information Processing Systems*, 36:35522–35543, 2023.

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 *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 2206–2240, 2022.

Sebastian Bruch. *Foundations of vector retrieval*. Springer, 2024.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021.

Lawrence Cayton and Sanjoy Dasgupta. A learning framework for nearest neighbor search. *Advances in Neural Information Processing Systems*, 20:233–240, 2007.

Matteo Ceccarello, Alexandra Levchenko, Ioana Ileana, and Themis Palpanas. Evaluating and generating query workloads for high dimensional vector similarity search. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2*, pages 5299–5310, 2025.

Meng Chen, Kai Zhang, Zhenying He, Yinan Jing, and X. Sean Wang. RoarGraph: A projected bipartite graph for efficient cross-modal approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 17(11):2735–2749, 2024.

Tingyang Chen, Cong Fu, Jiahua Wu, Haotian Wu, Hua Fan, Xiangyu Ke, Yunjun Gao, Yabo Ni, and Anxiang Zeng. Reveal hidden pitfalls and navigate next generation of vector similarity search from task-centric views: [experiments & analysis]. *Proceedings of the ACM on Management of Data*, 4(1):1–25, 2026.

Zhuoming Chen, Ranajoy Sadhukhan, Zihao Ye, Yang Zhou, Jianyu Zhang, Niklas Nolte, Yuandong Tian, Matthijs Douze, Leon Bottou, Zhihao Jia, and Beidi Chen. MagicPIG:

LSH sampling for efficient LLM generation. In *International Conference on Learning Representations*, 2025.

Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for YouTube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 191–198, 2016.

Sanjoy Dasgupta and Yoav Freund. Random projection trees and low dimensional manifolds. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, pages 537–546, 2008.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

Laxman Dhulipala, Majid Hadian, Rajesh Jayaram, Jason Lee, and Vahab Mirrokni. MUVERA: Multi-vector retrieval via fixed dimensional encoding. *Advances in Neural Information Processing Systems*, 37:101042–101073, 2024.

Wei Dong, Moses Charikar, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th International Conference on World Wide Web*, pages 577–586, 2011.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The Faiss library. *IEEE Transactions on Big Data*, 12(2):346–361, 2026.

D. C. Dowson and B. V. Landau. The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982.

Alex Fang, Gabriel Ilharco, Mitchell Wortsman, Yuhao Wan, Vaishaal Shankar, Achal Dave, and Ludwig Schmidt. Data determines distributional robustness in contrastive language image pre-training (CLIP). In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 6216–6234, 2022.

Jerome H. Friedman, Jon Louis Bentley, and Raphael Ari Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, 1977.

Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. *Proceedings of the VLDB Endowment*, 12 (5):461–474, 2019.

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

Jianyang Gao, Yutong Gou, Yuexuan Xu, Jifan Shi, Zhonghao Yang, and Cheng Long. The RaBitQ library. In *Proceedings of the 1st Workshop on Vector Databases at International Conference on Machine Learning*, 2025a.

Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. Practical and asymptotically optimal quantization of high-dimensional vectors in Euclidean space for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data*, 3(3):1–26, 2025b.

Yutong Gou, Jianyang Gao, Yuexuan Xu, and Cheng Long. SymphonyQG: Towards symphonious integration of quantization and graph for approximate nearest neighbor search. *Proceedings of the ACM on Management of Data*, 3(1):1–26, 2025.

Fabian Groh, Lukas Ruppert, Patrick Wieschollek, and Hendrik P. A. Lensch. GGNN: Graph-based GPU nearest neighbor search. *IEEE Transactions on Big Data*, 9(1):267–279, 2023.

Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. Accelerating large-scale inference with anisotropic vector quantization. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3887–3896, 2020.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3929–3938, 2020.

Junfeng He, Sanjiv Kumar, and Shih-Fu Chang. On the difficulty of nearest neighbor search. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1127–1134, 2012.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Sture Holm. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):65–70, 1979.

Doris Hoogeveen, Karin M. Verspoor, and Timothy Baldwin. CQADupStack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian Document Computing Symposium*, pages 3:1–3:8, 2015.

Ville Hyvönen, Teemu Pitkänen, Sotiris Tasoulis, Elias Jääsaari, Risto Tuomainen, Liang Wang, Jukka Corander, and Teemu Roos. Fast nearest neighbor search through sparse random projections and voting. In *Proceedings of the 2016 IEEE International Conference on Big Data*, pages 881–888. IEEE, 2016.

Ville Hyvönen, Elias Jääsaari, and Teemu Roos. A multilabel classification framework for approximate nearest neighbor search. *Advances in Neural Information Processing Systems*, 35:35741–35754, 2022.

Ville Hyvönen, Elias Jääsaari, and Teemu Roos. A multilabel classification framework for approximate nearest neighbor search. *Journal of Machine Learning Research*, 25(46):1–51, 2024.

Patrick Iff, Paul Brügger, Marcin Chrapek, David Kochergin, Maciej Besta, and Torsten Hoefler. Benchmarking filtered approximate nearest neighbor search algorithms on transformer-based embedding vectors. In *Proceedings of the 49th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 610–622, 2026.

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

Masajiro Iwasaki and Daisuke Miyazaki. Optimization of indexing based on k-nearest neighbor graph for proximity search in high-dimensional data. *arXiv preprint arXiv:1810.07355*, 2018.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*, 2022.

Elias Jääsaari, Ville Hyvönen, and Teemu Roos. Efficient autotuning of hyperparameters in approximate nearest neighbor search. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 590–602. Springer, 2019.

Elias Jääsaari, Ville Hyvönen, and Teemu Roos. LoRANN: Low-rank matrix factorization for approximate nearest neighbor search. *Advances in Neural Information Processing Systems*, 37:102121–102153, 2024.

Elias Jääsaari, Ville Hyvönen, and Teemu Roos. LEMUR: Learned multi-vector retrieval. In *Proceedings of the 43rd International Conference on Machine Learning*, volume 306 of *Proceedings of Machine Learning Research*, 2026.

Shikhar Jaiswal, Ravishankar Krishnaswamy, Ankit Garg, Harsha Vardhan Simhadri, and Sheshansh Agrawal. OOD-DiskANN: Efficient and scalable graph ANNS for out-of-distribution queries. *arXiv preprint arXiv:2211.12850*, 2022.

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

Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4904–4916, 2021.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

Guoxin Kang, Zhongxin Ge, Jingpei Hu, Xueya Zhang, Lei Wang, and Jianfeng Zhan. BigVectorBench: Heterogeneous data embedding and compound queries are essential in evaluating vector databases. *Proceedings of the VLDB Endowment*, 18(5):1536–1550, 2025.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. GooAQ: Open question answering with diverse answer types. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 421–433, 2021.

Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 39–48, 2020.

Soheil Kolouri, Kimia Nadjahi, Umut Simsekli, Roland Badeau, and Gustavo K. Rohde. Generalized sliced Wasserstein distances. *Advances in Neural Information Processing Systems*, 32:261–272, 2019.

Hans-Peter Kriegel, Erich Schubert, and Arthur Zimek. The (black) art of runtime evaluation: Are we comparing algorithms or implementations? *Knowledge and Information Systems*, 52:341–378, 2017.

Leonardo Kuffo and Peter Boncz. Bang for the buck: Vector search on cloud CPUs. In *Proceedings of the 21st International Workshop on Data Management on New Hardware*, DaMoN '25. Association for Computing Machinery, 2025.

Gregory M. Kurtzer, Vanessa Sochat, and Michael W. Bauer. Singularity: Scientific containers for mobility of compute. *PLOS ONE*, 12(5):e0177459, 2017.

Sean Lee, Aamir Shakir, Darius Koenig, and Julius Lipp. Open source strikes bread - new fluffy embedding model, 2024. URL `https://www.mixedbread.com/blog/mxbai-embed-large-v1`.

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. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. PAQ: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115, 2021.

Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and

improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488, 2020.

Mintaek Lim, Dogeun Kim, Minwoo Kim, and Jaeyoung Do. Revisiting filtered ANN benchmarks: A hardness-controlled benchmark generator for realistic evaluation. *arXiv preprint arXiv:2606.14193*, 2026.

Di Liu, Meng Chen, Baotong Lu, Huiqiang Jiang, Zhenhua Han, Qianxi Zhang, Qi Chen, Chengruidong Zhang, Bailu Ding, Kai Zhang, Chen Chen, Fan Yang, Yuqing Yang, and Lili Qiu. RetrievalAttention: Accelerating long-context LLM inference via vector retrieval. *Advances in Neural Information Processing Systems*, 38:54358–54385, 2025.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

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, 2020.

Pierre-Emmanuel Mazaré, Gergely Szilvasy, Maria Lomeli, Francisco Massa, Naila Murray, Hervé Jégou, and Matthijs Douze. Inference-time sparse attention with asymmetric indexing. *arXiv preprint arXiv:2502.08246*, 2025.

Leland McInnes. PyNNDescent: A Python library for approximate nearest neighbors, 2018. URL `https://github.com/lmcinnes/pynndescent/`.

Microsoft. Harrier-OSS-v1-270M, 2026. URL `https://huggingface.co/microsoft/harrier-oss-v1-270m`.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. MTEB: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037, 2023.

Blaise Munyampirwa, Vihan Lakshman, and Benjamin Coleman. Down with the hierarchy: The 'H' in HNSW stands for "hubs". In *Proceedings of the 48th European Conference on Information Retrieval (ECIR 2026), Part III*, pages 33–48, 2026.

Zach Nussbaum, Brandon Duderstadt, and Andriy Mulyar. Nomic embed vision: Expanding the latent space. *arXiv preprint arXiv:2406.18587*, 2024.

Zach Nussbaum, John Xavier Morris, Andriy Mulyar, and Brandon Duderstadt. Nomic Embed: Training a reproducible long context text embedder. *Transactions on Machine Learning Research*, 2025.

NVIDIA. cuVS: Vector search and clustering on the GPU, 2024. URL `https://github.com/rapidsai/cuvs`.

Hiroyuki Ootomo, Akira Naruse, Corey Nolet, Ray Wang, Tamas Feher, and Yong Wang. CAGRA: Highly parallel graph construction and approximate nearest neighbor search for GPUs. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 4236–4247. IEEE, 2024.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

Ninh Pham and Tao Liu. FALCONN++: A locality-sensitive filtering approach for approximate nearest neighbor search. *Advances in Neural Information Processing Systems*, 35: 31186–31198, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763, 2021.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, 2019.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Keshav Santhanam, Omar Khattab, Christopher Potts, and Matei Zaharia. PLAID: An efficient engine for late interaction retrieval. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 1747–1756, 2022a.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. ColBERTv2: Effective and efficient retrieval via lightweight late interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3715–3734, 2022b.

Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4430–4441. Association for Computational Linguistics, 2019.

Harsha Vardhan Simhadri, George Williams, Martin Aumüller, Matthijs Douze, Artem Babenko, Dmitry Baranchuk, Qi Chen, Lucas Hosseini, Ravishankar Krishnaswamny, Gopal Srinivasa, Suhas Jayaram Subramanya, and Jingdong Wang. Results of the NeurIPS'21 challenge on billion-scale approximate nearest neighbor search. In *Proceedings of the NeurIPS 2021 Competitions and Demonstrations Track*, volume 176 of *Proceedings of Machine Learning Research*, pages 177–189, 2022.

Harsha Vardhan Simhadri, Martin Aumüller, Matthijs Douze, Dmitry Baranchuk, Amir Ingber, Edo Liberty, George Williams, Ben Landrum, Magdalen Manohar, Mazin Karjikar, Laxman Dhulipala, Meng Chen, Yue Chen, Rui Ma, Kai Zhang, Yuzheng Cai, Jiayang Shi, Weiguo Zheng, Yizhuo Chen, Jie Yin, and Ben Huang. Results of the Big ANN: NeurIPS'23 competition. *Advances in Neural Information Processing Systems*, 38, 2025.

Spotify. Annoy: Approximate nearest neighbors oh yeah, 2013. URL `https://github.com/spotify/annoy`.

Robert F. Sproull. Refinements to nearest-neighbor searching in k-dimensional trees. *Algorithmica*, 6:579–589, 1991.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, 2023.

Suhas Jayaram Subramanya, Fnu Devvrit, Harsha Vardhan Simhadri, Ravishankar Krishnawamy, and Rohan Kadekodi. DiskANN: Fast accurate billion-point nearest neighbor search on a single node. *Advances in Neural Information Processing Systems*, 32:13771–13781, 2019.

Mariano Tepper, Ishwar Singh Bhati, Cecilia Aguerrebere, Mark Hildebrand, and Theodore L. Willke. LeanVec: Searching vectors faster by making them fit. *Transactions on Machine Learning Research*, 2024.

Grant Van Horn, Elijah Cole, Sara Beery, Kimberly Wilber, Serge Belongie, and Oisin Mac Aodha. Benchmarking representation learning for natural world image collections. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12884–12893, 2021.

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*, volume 235 of *Proceedings of Machine Learning Research*, pages 51255–51272, 2024.

Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment*, 14(11):1964–1978, 2021.

Mingqi Wang, Junyao Dong, Zhuoyan Wu, Jun Liu, Ruicheng Zhang, Jianjun Zhao, Ruipeng Wan, Xinyan Lei, Shuhao Zhang, Bolong Zheng, Haikun Liu, Xiaofei Liao, and Hai Jin. CANDOR-Bench: Benchmarking in-memory continuous ANNS under dynamic open-world streams [experiments & analysis]. *Proceedings of the ACM on Management of Data*, 4(1): 1–27, 2026.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788, 2020.

Zihao Wang. Graph library for approximate similarity search, 2025. URL `https://github.com/zilliztech/pyglass`.

Tobias Weyand, André Araujo, Bingyi Cao, and Jack Sim. Google landmarks dataset v2 - a large-scale benchmark for instance-level recognition and retrieval. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2575–2584, 2020.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6): 80–83, 1945.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. HuggingFace's Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2021.

Yahoo Japan. NGT: Neighborhood graph and tree for indexing high-dimensional data, 2015. URL `https://github.com/NGT-labs/NGT`.

Ji Yang, Xinyang Yi, Derek Zhiyuan Cheng, Lichan Hong, Yang Li, Simon Xiaoming Wang, Taibai Xu, and Ed H. Chi. Mixed negative sampling for learning two-tower neural networks in recommendations. In *Companion Proceedings of the Web Conference 2020*, pages 441–447, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, 2018.

Xianzhi Zeng, Zhuoyan Wu, Xinjing Hu, Xuanhua Shi, Shixuan Sun, and Shuhao Zhang. CANDY: A benchmark for continuous approximate nearest neighbor search with dynamic data ingestion. *arXiv preprint arXiv:2406.19651*, 2024.

Xiang Zhang, Chao Zhang, Ju Fan, Guoliang Li, and Xiaoyong Du. VecBench: A controllable benchmark for filtered vector search: [experiments & analysis]. *Proceedings of the ACM on Management of Data*, 4(3):1–27, 2026.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Appendix A. Benchmark and dataset description

A.1 Access

The VIBE benchmark and its associated datasets are publicly available and hosted on GitHub and Hugging Face, ensuring continued availability. The datasets can also be easily recreated from scratch using the provided framework. The companion website, the benchmark code, and the precomputed datasets can be accessed via the following links:

🌐 `https://vector-index-bench.github.io/`

`https://github.com/vector-index-bench/vibe/`

🤗 `https://huggingface.co/datasets/vector-index-bench/vibe`

A.2 Maintenance

VIBE is an ongoing effort, and we allow community contributions on GitHub for adding new methods to the benchmark. To contribute a new method, a contributor opens a pull request. After review and merging, the maintainers update the companion website with the new results. The results will be updated regularly using computing infrastructure provided by CSC – IT Center for Science, Finland. Instructions for adding new methods and datasets to the benchmark are provided in the repository README:

`https://github.com/vector-index-bench/vibe/blob/main/README.md`

A.3 Reproducibility

The benchmark code can be found in our GitHub repository. The README in the repository describes the requirements and instructions for running the benchmark. The benchmark framework automatically downloads the precomputed datasets, but the datasets can also be recreated from scratch using the provided instructions. After running the benchmark, all figures of this paper can be reproduced using the command `./plot.sh --plot-type paper`

A.4 License

The VIBE benchmarking framework is licensed under the MIT license. Most of the datasets in VIBE are created by the present authors using the embeddings and data sources detailed in Section A.9, and our embedding datasets are distributed under CC BY 4.0. However, as part of VIBE we also redistribute the following embedding datasets:

- GloVe[9] (Pennington et al., 2014) (under PDDL 1.0)
- Yandex T2I[10] (Simhadri et al., 2025) (a subset of 1 million vectors, under CC BY 4.0)
- LAION[11] (a subset of 1 million vectors, under CC BY 4.0)

A.5 Rights and responsibilities

The authors bear all responsibility in case of violation of rights associated with the datasets.

9. `https://nlp.stanford.edu/projects/glove/`
10. `https://big-ann-benchmarks.com/neurips23.html`
11. `https://laion.ai/blog/laion-400-open-dataset/`

A.6 Intended uses

The VIBE benchmark and its associated datasets are intended to be used only for benchmarking vector indexes. The datasets contain only raw embedding vectors and not the original documents or images that were used to compute the embeddings. The datasets are designed primarily for use with the VIBE framework, although they are also straightforward to use without the framework.

A.7 Framework structure

Each algorithm in the VIBE benchmark is provided as a self-contained module within the `vibe/algorithms/` directory, where each algorithm directory contains three files:

- `image.def`: Apptainer container definition that specifies the algorithm's dependencies and build instructions, inheriting from a common base image
- `config.yml`: defines algorithm variants with their hyperparameters, organized by data type and distance metric
- `module.py`: implements one or more Python classes with the required methods `fit()` for index building and `query()` for nearest neighbor search

New algorithms can be added by creating a new directory following this structure: implementing the interface with methods for fitting data and querying neighbors, defining algorithm variants with their hyperparameters in YAML format, and providing a container definition for reproducible installation and execution.

A.8 Dataset structure

Each dataset is distributed as a single HDF5 file.

File attributes The HDF5 files contain the following attributes:

`dimension` The dimensionality of the data

`distance` The distance metric to use; one of `euclidean`, `cosine`, `ip`, `hamming`, or `normalized`

`point_type` The precision of the vectors; one of `float`, `int8`, `uint8`, or `binary`

HDF5 datasets The HDF5 files contain the following HDF5 datasets:

`train` Array of size $(m_{\text{corpus}}, \text{dim})$: the embeddings used to build the vector index

`test` Array of size $(m_{\text{test}}, \text{dim})$: the test query embeddings

`neighbors` Array of size $(m_{\text{test}}, 100)$: the IDs of the true 100 nearest neighbors of each query

`distances` Array of size $(m_{\text{test}}, 100)$: the distances to the `neighbors`

`avg_distances` Array of size m_{test}: the average distance from each test query to the corpus

Additionally, the HDF5 files of OOD datasets contain the following HDF5 datasets:

`learn` Array of size $(m_{\text{learn}}, \text{dim})$: a larger sample from the query distribution

`learn_neighbors` Array of size $(m_{\text{learn}}, 100)$: the IDs of the true 100 nearest neighbors for each point in `learn`

A.9 Models and data sources

The embedding models used to produce the new datasets (see Tables 1 and 2) in VIBE are shown in Table 4. For discussion, we refer to Sections 4.1 and 4.2. The used data sources are shown in Table 5. Most of the data sources are under custom non-commercial licenses; for details, we refer to the provided references for each data source.

Approximate attention To construct our approximate attention computation datasets, we use Hugging Face Transformers (Wolf et al., 2019) with two long-context LLMs:

- `llama-128-ip`: Extracted from Layer 13, Head 16 of the gradientai/Llama-3-8B-Instruct-262k[12] model using a copy of *The Picture of Dorian Gray*[13] as the prompt
- `yi-128-ip`: Extracted from Layer 32, Head 14 of the 01-ai/Yi-6B-200K[14] model using a copy of the book *Pride and Prejudice*[15] as the prompt

Both models use grouped-query attention and have 32 query heads, but Yi has 4 key/value heads and Llama has 8. Thus, each Yi key head is shared by 8 consecutive query heads, whereas each Llama key head is shared by 4. Within each layer, at each token position, we therefore split the query projection into 32 query-head vectors and the key projection into 4 vectors for Yi or 8 vectors for Llama. All of these head vectors are 128-dimensional. This produces a separate query matrix $Q \in \mathbb{R}^{L\times 128}$ and key matrix $K \in \mathbb{R}^{L\times 128}$ for every layer–query-head pair, where L is the tokenized prompt length.

For a selected layer and head, the rows of K form the indexed corpus and the rows of Q form the queries. Following Mazaré et al. (2025), we additionally discard the keys and queries corresponding to the first token and the last 2047 tokens, as typically they have higher inner products with the queries and should be computed exactly.

To select the final layer–head pair, we estimate the sliced 2-Wasserstein distance (Kolouri et al., 2019) between K and Q for all layer and head combinations, and choose the pair with median distance as the final dataset. The benchmark results were similar across different prompts and candidate attention datasets with varying Wasserstein distances, so the selected pairs are representative of the approximate attention workload. The `create_dataset.sh` script in the code repository can be used to download other layer–head datasets.

Multi-vector reduction For the multi-vector reduction datasets `cqadupstack-muvera` and `cqadupstack-lemur`, we first use the ColBERTv2 (Santhanam et al., 2022b) model[16] to compute the token embeddings from CQADupStack (Hoogeveen et al., 2015). For each corpus document, we use a maximum of 220 token embeddings, and for each query document, we use exactly 32 token embeddings. To construct the single-vector reductions, for MUVERA[17] (Dhulipala et al., 2024), we set $R_{\text{reps}} = 40$, $k_{\text{sim}} = 6$, and $d_{\text{proj}} = 128$, and then apply a final projection to generate 5120-dimensional fixed-dimensional encodings (FDEs). For LEMUR[18] (Jääsaari et al., 2026), we set the hidden-layer size to $d' = 2048$.

12. `https://huggingface.co/gradientai/Llama-3-8B-Instruct-262k`
13. `https://www.gutenberg.org/ebooks/174`
14. `https://huggingface.co/01-ai/Yi-6B-200K`
15. `https://www.gutenberg.org/ebooks/1342`
16. `https://huggingface.co/colbert-ir/colbertv2.0`
17. `https://github.com/google/graph-mining/tree/main/sketching/point_cloud`
18. `https://github.com/ejaasaari/lemur`

Table 4: Embedding models used to produce the VIBE datasets. The license for each embedding applies to the model; none of the models claim rights over their outputs.

Name	Model	Citation	License
ALIGN	align-base	Jia et al. (2021)	Custom
CLIP	clip-vit-base-patch32	Radford et al. (2021)	MIT
DINO	vit_base_patch16_224.dino	Caron et al. (2021)	Apache 2.0
DistilRoBERTa	all-distilroberta-v1	Sanh et al. (2019)	Apache 2.0
Harrier	harrier-oss-v1-270m	Microsoft (2026)	MIT
Jina	jina-embeddings-v5-text-nano	Akram et al. (2026)	CC-BY-NC-4.0
MiniLM	all-MiniLM-L6-v2	Wang et al. (2020)	Apache 2.0
MXBAI	mxbai-embed-large-v1	Lee et al. (2024)	Apache 2.0
Nomic Text	nomic-embed-text-v1.5	Nussbaum et al. (2025)	Apache 2.0
Nomic Vision	nomic-embed-vision-v1.5	Nussbaum et al. (2024)	Apache 2.0
Qwen	Qwen3-Embedding-0.6B	Zhang et al. (2025)	Apache 2.0
ResNet	resnet50	He et al. (2016)	BSD 3

Table 5: Data sources used to produce the VIBE embedding datasets.

Name	Type	n	Reference	Citation
AGNews	Text	770 382	[1]	N/A
arXiv abstracts	Text	1 345 643	[2]	N/A
CQADupStack	Text	457 149	[3]	Hoogeveen et al. (2015)
DPR	Text	20 970 760	[4]	Karpukhin et al. (2020)
GooAQ	Text	1 476 024	[5]	Khashabi et al. (2021)
HotpotQA	Text	5 233 329	[6]	Yang et al. (2018)
ImageNet	Image	1 281 167	[7]	Deng et al. (2009)
ImageNet-Captions	Text	316 648	[8]	Fang et al. (2022)
iNaturalist Mini	Image	500 000	[9]	Van Horn et al. (2021)
Landmark	Image	761 757	[10]	Weyand et al. (2020)
MS MARCO	Text	8 841 823	[11]	Bajaj et al. (2016)
Yahoo Answers	Text	678 305	[12]	N/A

[1] http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
[2] https://www.kaggle.com/datasets/Cornell-University/arxiv
[3] http://nlp.cis.unimelb.edu.au/resources/cqadupstack/
[4] https://github.com/facebookresearch/DPR
[5] https://github.com/allenai/gooaq
[6] https://hotpotqa.github.io/
[7] https://www.image-net.org/
[8] https://github.com/mlfoundations/imagenet-captions
[9] https://github.com/visipedia/inat_comp/tree/master/2021
[10] https://github.com/cvdfoundation/google-landmark
[11] https://microsoft.github.io/msmarco/
[12] https://www.kaggle.com/datasets/soumikrakshit/yahoo-answers-dataset

Appendix B. Interactive website

Our interactive website is available at `https://vector-index-bench.github.io/`. The website can also be run locally by following the instructions at `https://github.com/vector-index-bench/vector-index-bench.github.io`. It provides pages that allow users to (1) compare all algorithms at a given recall level; (2) explore the distribution of queries and data of different datasets; (3) investigate the trade-offs of different implementations (Figure 12); (4) assess the performance of all configurations for each algorithm (Figure 13); (5) evaluate the robustness of algorithms on easy and difficult queries.

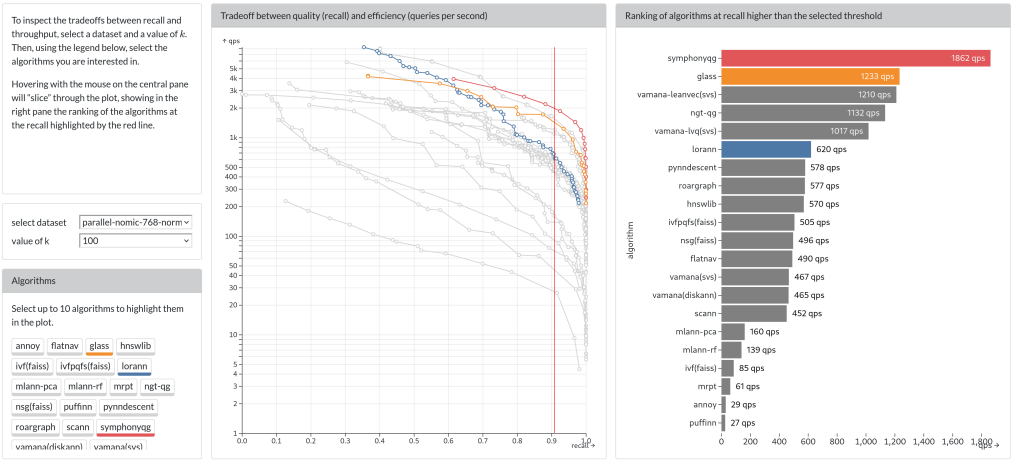


Figure 12: Screenshot of the webpage that allows users to explore the recall/throughput trade-off interactively.

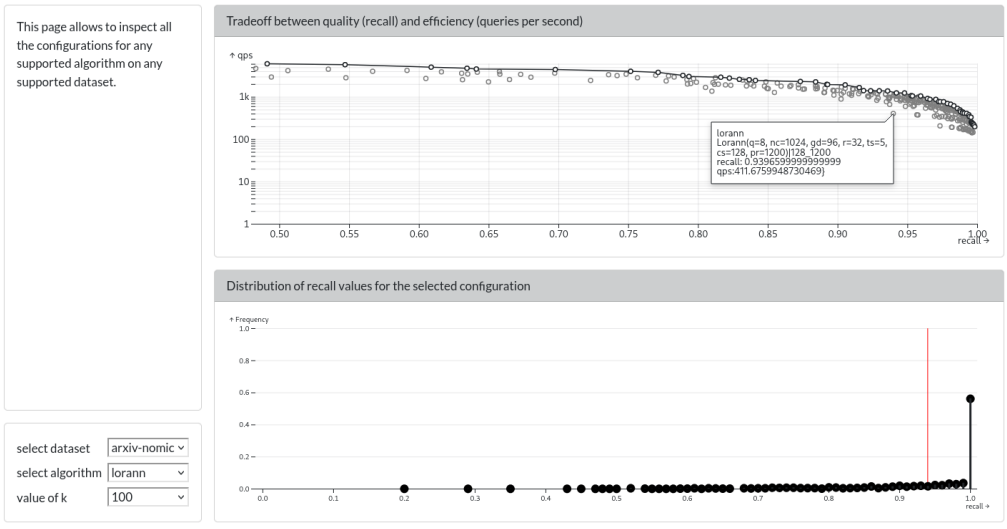


Figure 13: Screenshot of the webpage that allows users to explore the behavior of all the tested configurations of a given algorithm interactively.

Appendix C. Distribution shift in OOD datasets

We quantify the distribution shift in the OOD datasets using three metrics: Multiscale maximum mean discrepancy (MMD) captures general differences between distributions across multiple kernel length scales. The Fréchet distance measures differences in the means and covariance structures of the embedding distributions (Dowson and Landau, 1982), while sliced 2-Wasserstein measures discrepancies between their one-dimensional projections and therefore captures broad geometric changes in their mass distributions (Kolouri et al., 2019).

In Table 6, we show these three metrics for each OOD dataset using data-versus-data ($D \leftrightarrow D$) (within-distribution reference) and data-versus-query ($D \leftrightarrow Q$) metrics. The query distribution is represented by the larger query sample provided with each OOD dataset rather than by the 1000 held-out test queries. All evaluated datasets are clearly out of distribution, as their data-versus-query distances consistently exceed the corresponding data-versus-data reference distances across all three metrics. The `llama` and `yi` datasets exhibit the most severe overall shifts, while `hotpotqa-harrier` and `yandex` show the weakest shifts.

Table 6: Distribution distances for the OOD datasets. For each metric, $D \leftrightarrow D$ measures the within-distribution distance between two corpus samples, while $D \leftrightarrow Q$ measures the distance between the corpus and query distributions. Larger $D \leftrightarrow Q$ values relative to $D \leftrightarrow D$ indicate a stronger distribution shift.

dataset	multiscale MMD		Fréchet distance		sliced 2-Wasserstein	
	$D \leftrightarrow D$	$D \leftrightarrow Q$	$D \leftrightarrow D$	$D \leftrightarrow Q$	$D \leftrightarrow D$	$D \leftrightarrow Q$
hotpotqa-harrier	0.0022	0.2084	0.0380	0.5322	0.0004	0.0176
laion-clip	0.0009	0.3298	0.0309	0.8818	0.0004	0.0328
imagenet-align	0.0013	0.4269	0.0371	1.0477	0.0004	0.0374
yandex	0.0010	0.0850	0.0277	0.5073	0.0008	0.0150
cqadupstack-muvera	0.0011	0.2335	6.1337	47.3362	0.0075	0.5478
cqadupstack-lemur	0.0014	0.5180	0.7450	9.6105	0.0024	0.1802
yi	0.0018	0.6612	0.2156	26.4129	0.0113	2.2717
llama	0.0014	0.8017	0.1931	32.7253	0.0107	2.7928

Appendix D. Algorithms

In Table 7, we give a detailed list of versions of each algorithm used in the experiments. The name of each method links to the corresponding library used in the benchmark. Clustering-based methods are highlighted in green, tree-based in purple, hashing-based in orange, and graph-based in blue.

Table 7: Details of the algorithm implementations included in the benchmark.

Method	Reference	Version
ANNOY	Spotify (2013)	1.17.3
FALCONN++	Pham and Liu (2022)	git+5fd3f17
FlatNav	Munyampirwa et al. (2026)	0.1.2
Glass	Wang (2025)	git+d2296ec
HNSW	Malkov and Yashunin (2020)	0.8.0
HNSW-RaBitQ	Gao and Long (2024); Gao et al. (2025b,a)	git+5ea4df0
IVF	Jégou et al. (2011); Douze et al. (2026)	1.14.3
IVF-PQ	Jégou et al. (2011); Douze et al. (2026)	1.14.3
IVF-RaBitQ	Gao and Long (2024); Gao et al. (2025a,b)	git+5ea4df0
LVQ	Aguerrebere et al. (2023)	0.4.0
LeanVec	Tepper et al. (2024)	0.4.0
LoRANN	Jääsaari et al. (2024)	0.4.5
MLANN	Hyvönen et al. (2022)	git+ba141b4
MRPT	Hyvönen et al. (2016); Jääsaari et al. (2019)	2.0.4
NGT-ONNG	Iwasaki and Miyazaki (2018); Yahoo Japan (2015)	2.7.4
NGT-QG	Yahoo Japan (2015)	2.7.4
NSG	Fu et al. (2019); Douze et al. (2026)	1.14.3
PUFFINN	Aumüller et al. (2019)	git+fd86b0d
PyNNDescent	Dong et al. (2011); McInnes (2018)	0.6.0
RoarGraph	Chen et al. (2024)	git+f2b49b6
ScaNN	Guo et al. (2020)	1.4.2
SymphonyQG	Gou et al. (2025)	git+32a0019
Vamana	Subramanya et al. (2019)	0.7.0

The list of algorithms used in the GPU experiments is given in Table 8.

Table 8: Details of the GPU algorithm implementations included in the benchmark.

Method	Reference	Version
CAGRA (cuVS)	Ootomo et al. (2024); NVIDIA (2024)	26.04.00
GGNN	Groh et al. (2023)	0.9
IVF-PQ (cuVS)	NVIDIA (2024)	26.04.00
IVF (cuVS)	NVIDIA (2024)	26.04.00
IVF (Faiss)	Johnson et al. (2021); Douze et al. (2026)	1.14.3

Appendix E. Index construction time

We measure the index construction time on a single CPU core. Table 9 shows index construction times for all algorithms on 10 datasets at 95% average recall for $k = 100$ at optimal hyperparameters (a dash means that the algorithm did not reach 95% average recall).

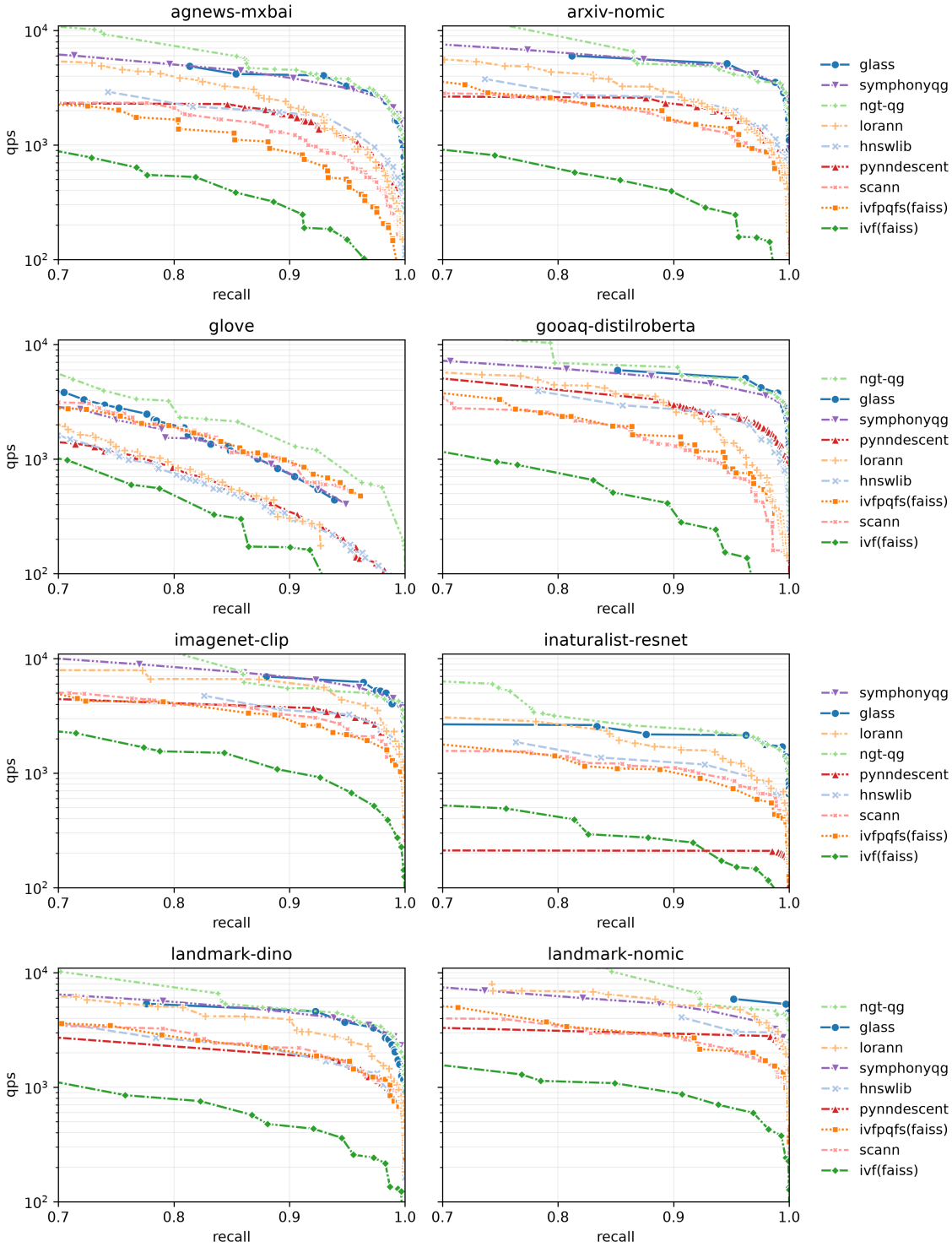
Table 9: Index construction times (seconds) with throughput-optimized hyperparameters at 95% average recall. The final column reports the mean ± standard deviation of construction time relative to the fastest construction time on each dataset. Algorithms are ordered by mean relative build time. Clustering methods are highlighted in green, tree in purple, hashing in orange, and graph in blue.

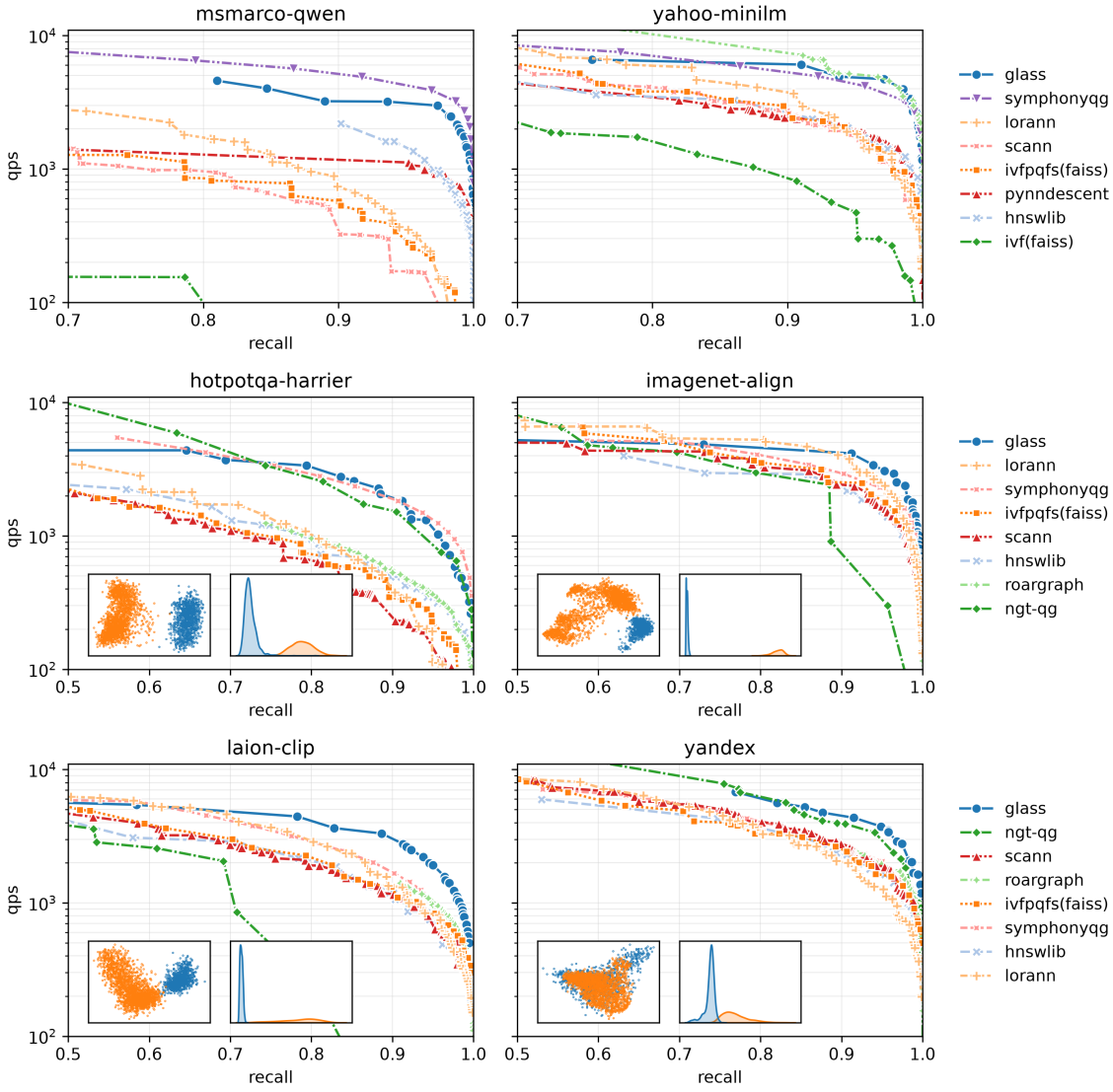
algorithm	agnews-mxbai	arxiv-nomic	glove	gooaq-distilroberta	imagenet-clip	inaturalist-resnet	landmark-dino	landmark-nomic	msmarco-qwen	yahoo-minilm	relative build time (mean ± sd)
ivfpqfs(faiss)	92	81	71	82	35	145	69	35	561	38	1.4 ±0.6
rabitq-ivf	78	77	235	79	56	47	70	58	395	23	1.4 ±0.8
ivf(faiss)	214	571	63	563	432	108	50	443	394	246	5.8 ±4.5
scann	689	761	412	764	479	1004	468	248	5497	241	11.1 ±4.1
glass	987	929	-	1045	658	535	797	375	11674	562	16.5 ±6.1
pynndescent	327	427	2524	965	288	2136	551	230	11404	384	17.9 ±14.2
vamana-leanvec(svs)	1052	1499	-	1698	912	501	596	427	10369	774	19.4 ±7.5
mrpt	1263	1872	870	1966	1424	1037	1023	988	-	623	24.1 ±7.3
puffinn	-	2106	676	2363	1147	-	1258	1231	-	689	27.2 ±7.3
vamana-lvq(svs)	1823	2219	-	2323	1070	-	1133	1017	14797	832	29.6 ±4.8
annoy	446	2797	2125	3072	487	2197	2341	354	22873	948	33.1 ±16.6
lorann	1149	1656	-	3526	2706	1086	1045	1773	19055	1539	40.7 ±20.7
flatnav	1585	3021	5509	3231	1362	1890	2075	909	16878	1542	44.3 ±18.3
nsg(faiss)	2256	2863	-	2796	2485	1626	2069	1801	20204	1465	46.0 ±13.3
symphonyqg	2207	3152	-	4085	2013	-	1760	1589	23779	1437	47.6 ±11.4
falconnpp	-	2843	1906	1573	3152	1762	927	2994	28190	1343	49.6 ±25.7
hnswlib	2211	3090	7591	3521	1627	2069	2005	863	30935	1165	51.8 ±26.8
vamana(diskann)	2393	4407	4237	3597	2378	1644	1546	2178	31148	1205	52.7 ±16.0
rabitq-hnsw	4610	4592	11013	7820	3138	3350	3375	2218	33884	2814	89.1 ±34.2
ngt-qg	14735	7352	40140	12047	3898	7977	5188	3486	-	7665	209.7 ±166.3

Comparing index construction times, it is worth noting that the hyperparameters in VIBE have been tuned for optimal query throughput, and index construction times may vary significantly depending on hyperparameter choices. We also note that, in general, the index construction of clustering-based and tree-based methods can be parallelized more easily than the index construction of graph-based methods.

Appendix F. Additional results

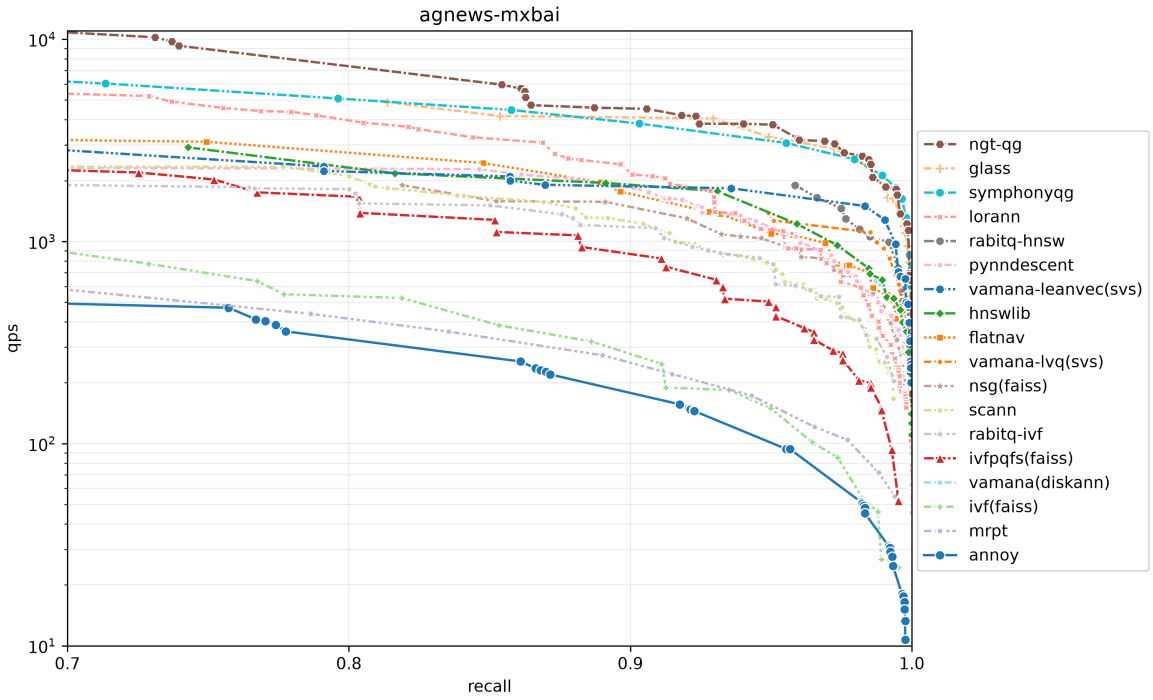
F.1 Selected algorithms

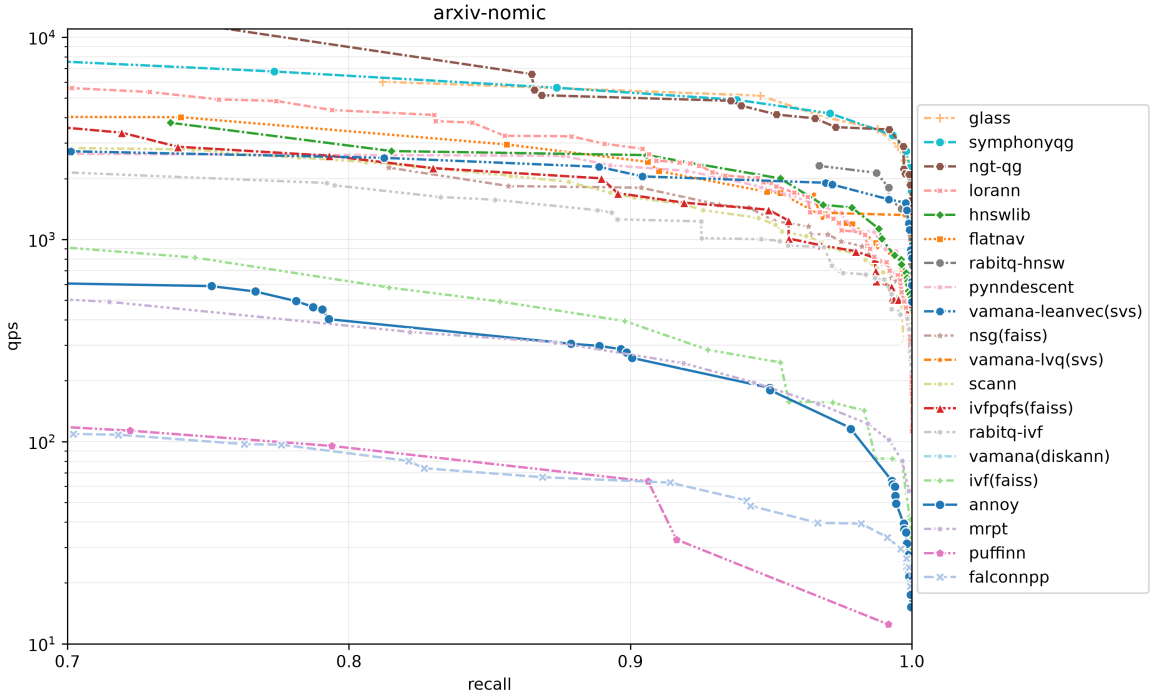



msmarco-qwen
yahoo-minilm
hotpotqa-harrier
imagenet-align
laion-clip
yandex
qps
recall
glass
symphonyqg
lorann
scann
ivfpqfs(faiss)
pynndescent
hnswlib
ivf(faiss)
roargraph
ngt-qg

F.2 All algorithms

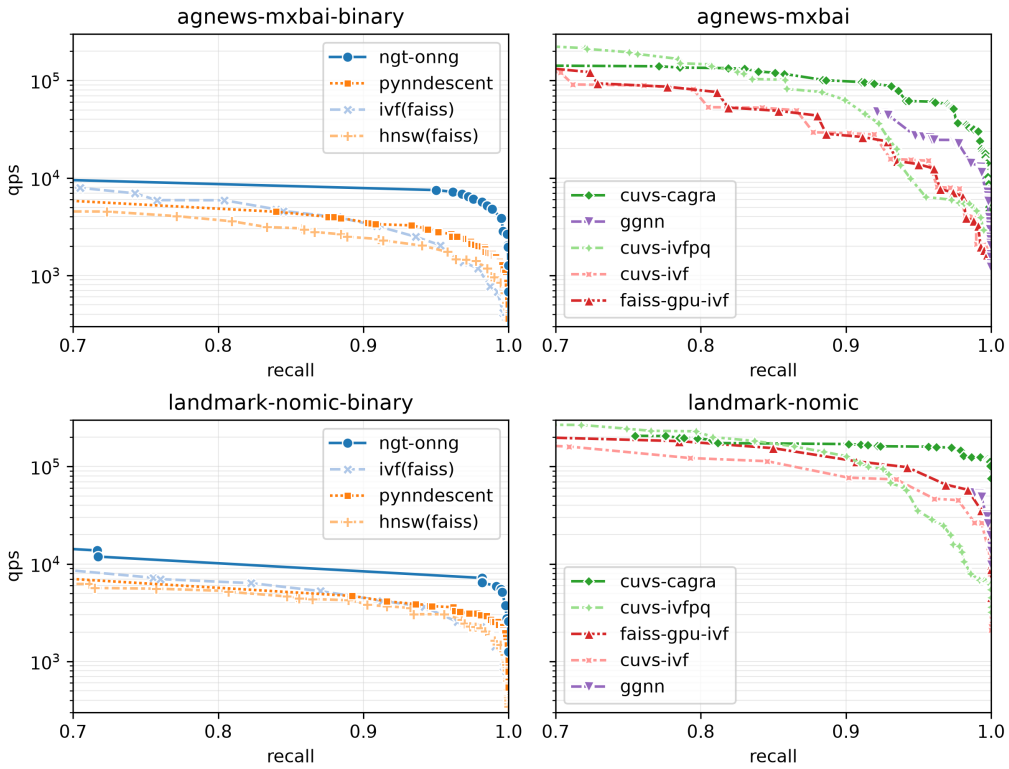
In this section, we present benchmarking results for all algorithms on two datasets. For full results on all datasets, we refer to our website: `https://vector-index-bench.github.io/`





F.3 Binary embeddings and GPU algorithms

In this section, we present additional results on achieving a higher throughput through binary embeddings and GPU algorithms. For details on the GPU algorithms, see Appendix D. For discussion, we refer to Section 5.2.



Appendix G. Hypervolume

In Table 10, we summarize the recall–throughput trade-off results on 10 datasets using normalized hypervolume over the average recall interval $[0.9, 1]$. Let $\mathcal{C}_{a,d} = \{(r_{a,d,i}, q_{a,d,i})\}_i$ denote the (average recall, QPS) pairs for algorithm a on dataset d. For a threshold τ, define

$$Q_{a,d}(\tau) := \max\left(\{q : (r,q) \in \mathcal{C}_{a,d},\, r \geq \tau\} \cup \{0\}\right), \qquad \tilde{Q}_{a,d}(\tau) := \frac{Q_{a,d}(\tau)}{\max_{a'} Q_{a',d}(0.9)}.$$

That is, $\tilde{Q}_{a,d}(\tau)$ is the normalized best throughput among configurations achieving average recall at least τ. The normalized hypervolume is given by

$$\mathrm{HV}_{a,d} = \mathbb{E}_{\tau \sim \mathrm{Unif}[0.9,1]}\left[\tilde{Q}_{a,d}(\tau)\right].$$

Table 10: Normalized hypervolume of recall–throughput curves over average recall $[0.9, 1]$. The final column gives mean ± standard deviation across datasets with rows sorted by decreasing mean. Colors denote clustering, tree, hashing, and graph methods.

algorithm	agnews-mxbai	arxiv-nomic	glove	gooaq-distilroberta	imagenet-clip	inaturalist-resnet	landmark-dino	landmark-nomic	msmarco-qwen	yahoo-minilm	hypervolume (mean ± sd)
glass	0.67	0.83	0.16	0.71	0.86	0.83	0.72	0.84	0.57	0.63	0.68 ±0.20
ngt-qg	0.73	0.80	0.52	0.69	0.75	0.85	0.79	0.77	0.00	0.71	0.66 ±0.24
symphonyqg	0.60	0.81	0.18	0.59	0.81	0.00	0.72	0.57	0.76	0.54	0.56 ±0.26
rabitq-hnsw	0.35	0.42	0.09	0.30	0.42	0.67	0.36	0.49	0.38	0.27	0.38 ±0.14
lorann	0.25	0.33	0.06	0.22	0.61	0.50	0.47	0.63	0.07	0.25	0.34 ±0.20
vamana-leanvec(svs)	0.34	0.35	0.06	0.32	0.30	0.48	0.32	0.37	0.37	0.25	0.32 ±0.10
hnswlib	0.26	0.33	0.13	0.30	0.42	0.38	0.28	0.45	0.25	0.24	0.30 ±0.09
flatnav	0.23	0.29	0.10	0.30	0.43	0.39	0.26	0.45	0.23	0.22	0.29 ±0.10
pynndescent	0.24	0.31	0.14	0.35	0.45	0.09	0.30	0.41	0.20	0.26	0.27 ±0.11
vamana-lvq(svs)	0.26	0.30	0.09	0.29	0.28	0.00	0.27	0.28	0.35	0.23	0.23 ±0.10
scann	0.15	0.21	0.26	0.11	0.35	0.33	0.32	0.29	0.04	0.22	0.23 ±0.10
nsg(faiss)	0.20	0.23	0.00	0.22	0.30	0.47	0.20	0.31	0.20	0.15	0.23 ±0.11
ivfpqfs(faiss)	0.10	0.22	0.32	0.14	0.30	0.28	0.30	0.29	0.06	0.23	0.22 ±0.09
rabitq-ivf	0.15	0.18	0.13	0.10	0.25	0.32	0.24	0.30	0.07	0.23	0.20 ±0.08
vamana(diskann)	0.14	0.15	0.13	0.14	0.21	0.20	0.15	0.16	0.16	0.14	0.16 ±0.03
ivf(faiss)	0.03	0.04	0.05	0.02	0.09	0.06	0.06	0.09	0.00	0.05	0.05 ±0.03
mrpt	0.03	0.03	0.03	0.03	0.03	0.03	0.03	0.07	0.00	0.04	0.03 ±0.02
annoy	0.02	0.03	0.02	0.01	0.06	0.04	0.03	0.05	0.01	0.02	0.03 ±0.02
falconnpp	0.00	0.01	0.05	0.02	0.02	0.04	0.02	0.01	0.01	0.03	0.02 ±0.01
puffinn	0.00	0.00	0.02	0.02	0.01	0.03	0.02	0.01	0.00	0.02	0.01 ±0.01

Appendix H. OOD performance gap

In this section, we present the gap between the performance of in-distribution and out-of-distribution queries on the text-retrieval and text-to-image OOD datasets. For discussion, we refer to Section 5.3.

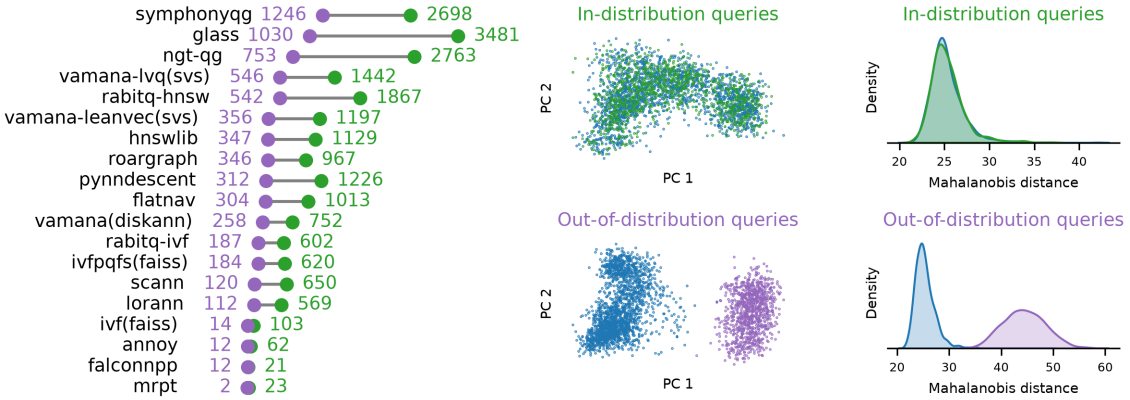


Figure 14: OOD performance gap on `hotpotqa-harrier`

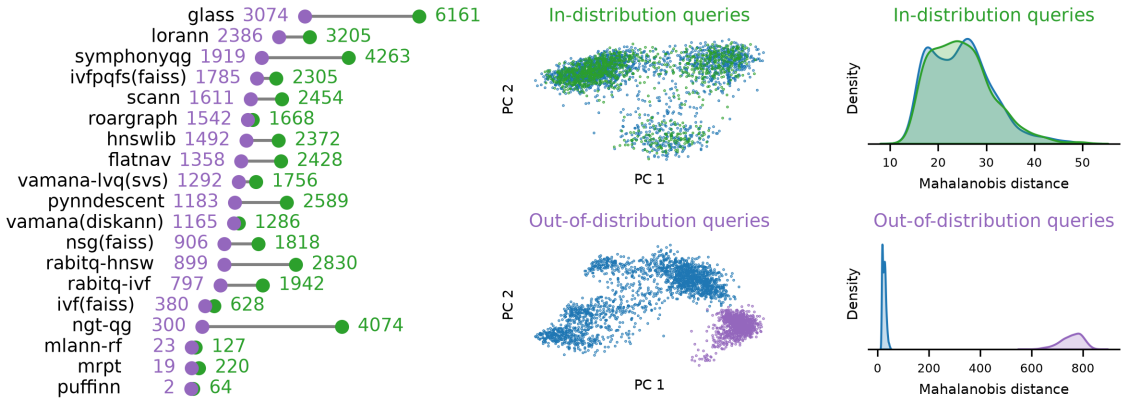


Figure 15: OOD performance gap on `imagenet-align`

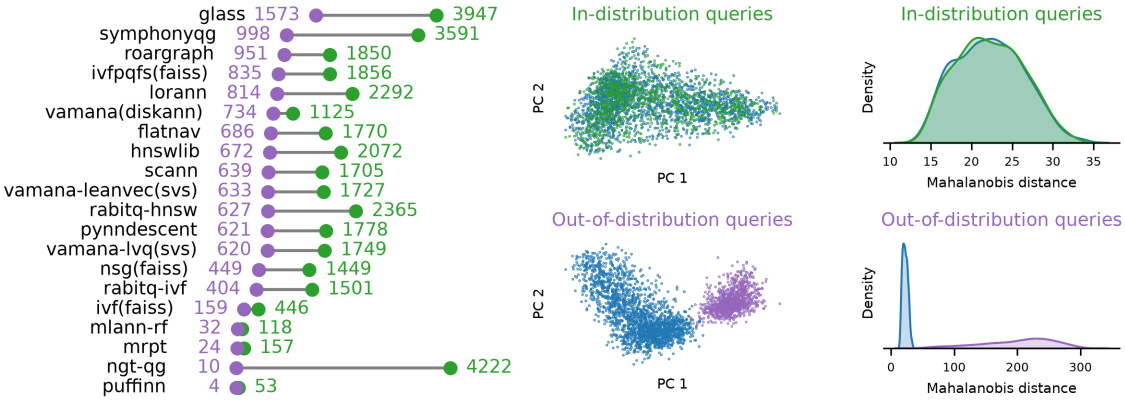


Figure 16: OOD performance gap on `laion-clip`

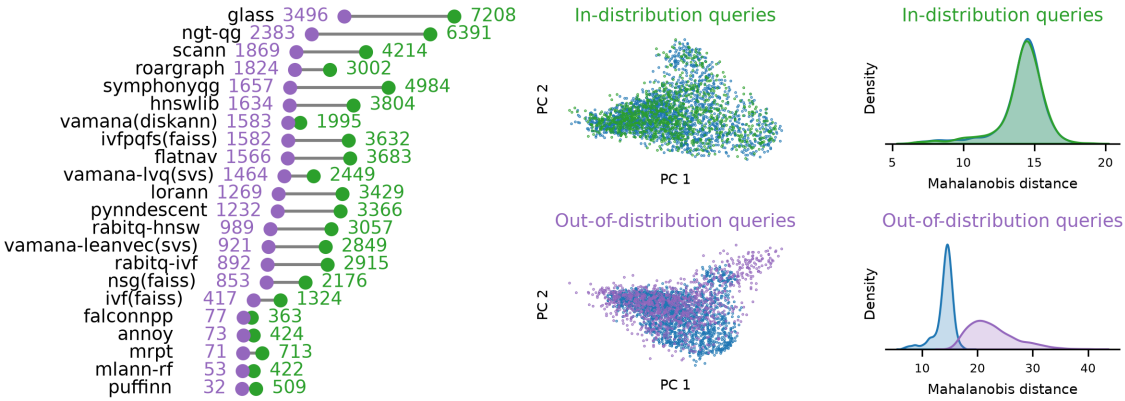


Figure 17: OOD performance gap on `yandex`