Overview of the SISAP 2024 Indexing Challenge

Eric S. Tellez[1], Martin Aumüller[2], and Vladimir Mic[3]

[1] eric.tellez@ieee.org INFOTEC; IxM CONACyT, México.
[2] maau@itu.dk ITU Copenhagen, Denmark.
[3] v.mic@cs.au.dk Aarhus University, Denmark.

Abstract. The SISAP 2024 Indexing Challenge invited replicable and competitive approximate similarity search solutions for datasets of up to 100 million real-valued vectors. Participants are evaluated on the search performance of their implementations under quality constraints. Using a subset of the deep features of a neural network model provided by the LAION-5B dataset, the challenge posed three tasks, each with its unique focus:

- **Task 1, Unrestricted indexing**: Conduct a classical approximate nearest neighbors search, ensuring an average recall of at least 0.8 for 30-NN queries.
- **Task 2, Memory-constrained indexing with reranking**: Conduct nearest neighbors search in a low-memory setting where the dataset collection is only accessible on disk, ensuring the same quality as in Task 1.
- **Task 3, Memory-constrained indexing without reranking**: Conduct nearest neighbor search in a setting where the dataset cannot be accessed at search stage, ensuring an average recall of at least 0.4 for 30-NN queries.

The present paper describes the details of the challenge, the evaluation system that was developed with it, and gives an overview of the submitted solutions.

Keywords: Approximate nearest neighbor search · Indexing and searching pipelines · Experimental comparison of search methods

1 Introduction

Large Language Models (LLMs) led to a fundamental change in how computers can help humans to solve complex tasks. However, once model training has finished, the corpus of knowledge of the model remains static. A crucial component for the applicability of these systems is thus that knowledge can be added post-hoc. To do so, systems can retrieve semantically matching, additional knowledge from large knowledge corpora [11]. This semantic search is made possible through deep-learning based vector embedding techniques such as CLIP [7], in which a piece of text or an image, among others, can be represented as a real-valued vector in a high-dimensional space.

Given a large collection of high-dimensional objects in a metric space, the most common search operation is to find the nearest neighbors of a given point in the space. An *exact* metric search structure will either detorate to a linear scan of the whole data collection, or use too much space in these high-dimensional spaces [3]. This means that the search algorithms should become approximate, i.e., *inexact*, to let users trade between speed and quality. Furthermore, there is a compromise with other practical constraints like preprocessing and construction time, as well as the whole memory usage, leading to various indexing solutions, each with merits and drawbacks.

The SISAP Indexing Challenge[4] seeks to identify efficient similarity search algorithms that strike a balance between accuracy and practical constraints like construction time, search time, and memory consumption. To facilitate this, we devised a 100M vectors test using the LAION deep features English subset [12]. Additionally, there are two query sets: public and private, each comprising 10K vectors. The public queries were made available during the call for papers, while the private ones were revealed post-submission and were used for the evaluation.

Dataset. This challenge is the second iteration of the SISAP indexing challenge; see [18] for an overview of the first iteration. As in this previous iteration, we used the CLIP image embeddings from the LAION dataset [12] as basis for the challenge. We selected a 100-million vector subset excluding *Not Safe for Work* (NSFW) elements. Note that the LAION dataset has been removed from several mirrors due to illegal content access on their metadata; therefore, we remove metadata access and take synthetic decisions to construct queries that will be detailed in §2.2.[5] The dataset contains 768-dimension vectors in the unit sphere such that the cosine can be used as a similarity function without needing vector normalization. More details on the dataset can be found on the challenge's companion site and the overview of the previous challenge [18].

Related Challenges and Benchmarks. Given its wide application-range, numerous methods for approximate nearest neighbor search have been developed in the last decade. The ann-benchmarks project [1] provides a standard benchmarking environment for million-scale approximate nearest neighbor search with a large collection of implementations. Alternatively, Li et al. [8] provide an alternative experimental survey and Matsui [9] provides an alternative benchmarking environment. For larger data collections, the 2021 NeurIPS challenge by Simhadri et al. [14] focused on billion-scale approximate nearest neighbor search, and the 2023 NeurIPS challenge by Simhadri et al. [13] focused on nearest neighbor search under different constraints such as the presence of metadata filters, or in a streaming setting.

[4] Official site of the challenge `https://sisap-challenges.github.io/`.
[5] Our datasets are made of image embeddings avoiding any access to metadata.

2 Challenge Details, Evaluation Setup, and Task Description

The Indexing Challenge focuses on approximate k nearest neighbor search. Given a collection of d-dimensional vectors $S \subseteq \mathbb{R}^d$, each task comes in two phases: First, given S, the index construction phase asks to layout the data to answer nearest neighbor searches quickly. Traditionally, layouts can be based on building a graph, a tree, or partitioning the space using a quantizer such as clustering-based or hashing-based approaches, see [2] for a recent survey. In the second phase, the query workload $Y \subseteq \mathbb{R}^d$ is provided, and the task is to find the 30 nearest neighbors for each query in Y. The challenge tasks simulate scenarios with different needs in terms of quality, speed, and memory.

The reproducibility is crucial; therefore, submissions must have an operational Github Action (GHA) workflow.[6] Teams crafted their solutions by meticulously setting and benchmarking hyperparameters for each task and detailing their choices in their GHA entry point.

2.1 Indexing Tasks

All tasks revolved around retrieving the approximate $k = 30$ nearest neighbors. We expected teams to construct indexes that efficiently solve queries and excel under each task's specific conditions and metrics. We first review the evaluation hardware, and then describe the three different tasks.

Hardware Similarly to the previous SISAP Challenge 2023, we limit the index construction time in each task, making the hardware parameters important for participants. The evaluation was going to be conducted on 2x Intel(R) Xeon(R) CPU E5-2690 V4 CPUs (28 cores, 56 hyperthreads) workstation with 512GiB of RAM. The original dataset resided on a 1TB SSD. Despite the same hardware used a year ago, current tasks focus more on a search with limited hardware. These hardware limits were enforced through the use of Docker containers.

Task 1: Unrestricted Indexing Task 1 provides continuity to the SISAP Challenge 2023. Inspired by Task A from the previous year, it allows teams to use all resources on the evaluation computer to build an index for the respective datasets and carry out search operations as quickly as possible under quality thresholds. Unlike in the previous iteration, we (1) limit the index building time to 12 hours instead of 24 hours, (2) specify that the index does not have to be serialized on a disk, and (3) set a threshold for an average recall of at least 0.8. The solutions meeting these requirements using a private query set are sorted by their search time.

[6] Github Actions is a continuous integration platform that enables continuous testing of repositories within virtual machines.

Task 2: Memory-Constrained Indexing with Reranking Task 2 is motivated by limited hardware situations, which is expected, for instance, when sharing machines in cloud services or whenever very large datasets are handled. This task considers the main memory to be the main bottleneck. Participants are given just eight virtual CPUs and 32 GiB of RAM. The wall clock time for index construction is 24 hours. The search quality still targets an average recall of at least 0.8, as in Task 1. For 100 million vectors of 768 dimensions using 4 bytes per coordinate, storing the vectors requires more than 300 GiB of RAM. Therefore, the dataset cannot be stored uncompressed/unembedded, resulting in an accuracy loss. To reach the target quality, participants are expected to refine the candidates in a final phase, as the search dataset is available on SSD.

Task 3: Memory-Constrained Indexing without Reranking Task 3 considers persistent storage to be the main bottleneck. Participants are asked to develop a memory-efficient index that does not use re-ranking. Participants must build an index in up to 12 hours using all available CPU cores and with at most 64 GiB of RAM. In the search phase, the original vectors cannot be used. The minimum recall to be considered in the final ranking is 0.4.

2.2 Public and Private Query Workloads

The indexing challenge provides two query workloads, each with 10 thousand query vectors; both query sets were extracted from different subsets not seen in any of the datasets given for indexing. The public query set was provided with the call for participation; it is intended to be used by participating teams to fine-tune their submissions and provide a thorough analysis of their system in their report. The private one is used to evaluate all systems and produce this manuscript's analysis.

This 2024 edition uses the private query set of the previous year (2023) as the public query set. We computed gold standards for k nearest neighbor queries, which are the foundation for calculating the recall score in the final results. Both the query set and the gold standard are accessible from the SISAP Indexing Challenge site.[7]

To make the evaluation more interesting, this year's private query set was chosen slightly differently than last year: Both query sets were selected from different LAION bundles[8]. For the public query set, we removed near-duplicate objects/queries, considering as *near-duplicate* any object inside a radius of 0.15;[9] for the private query set, we remove near-duplicates with a radius of 0.2. This slight change is reflected in variations on the expected quality that show the robustness of each system solution; note that the challenge rules allow up to 30 search hyperparameter probes that are expected to help overcome this query distribution change. Additionally, we removed any query retrieving a zero-distant

[7] Gold standards incorporate results up to $k = 1000$.

[8] The LAION dataset was available in bundles of close to one million vectors

[9] Using the cosine distance, e.g., $1 - cosine(q_i, q_j)$; also note that we used an approximate algorithm to remove near duplicates.

Table 1. Quantile values of the 30th neighborhood, i.e., $1-cosine$; note that Q_0 means for minimum and Q_1 for maximum values.

query set	Q_0	$Q_{0.1}$	$Q_{0.2}$	$Q_{0.3}$	$Q_{0.4}$	$Q_{0.5}$	$Q_{0.6}$	$Q_{0.7}$	$Q_{0.8}$	$Q_{0.9}$	Q_1
public	0.012	0.08	0.105	0.127	0.149	0.171	0.196	0.224	0.26	0.313	0.604
private	0.009	0.146	0.168	0.187	0.205	0.224	0.246	0.271	0.303	0.348	0.593

nearest neighbor and those with distance values tying at 29th and 30th neighbors. The private query set was made available after the evaluation results were presented to the participants.

Table 1 shows the distance distribution of the 30th nearest neighbor for the two different query sets. We can notice the effect of the near duplicate radius by observing larger distances in the private query set for all quantiles (0.1 to 0.9) except for the extremes.

3 Solutions Overview

This section describes the set of solutions to the SISAP 2024 Indexing Challenge. The solutions were implemented in C++ and Rust programming languages using Python as a wrapper. Our baselines used the Julia programming language. As detailed below, system solutions used different index structures, graph-based indexes, combinatorial and numerical optimization, and neural networks to learn how to project the dataset. Their submissions were carefully crafted to exploit the dataset characteristics and running setups.

3.1 Baselines

We present two baselines: **BL-SearchGraph** and **BL-Bruteforce**, based on on the Julia's package `SimilaritySearch.jl`. The BL-SearchGraph uses the SearchGraph index; see [17,16]. This index is a graph-based index similar to the HNSW, but instead of a hierarchy, it uses a small sample of disjoint neighbors to get fast navigation. In contrast to HNSW, it uses variable-size neighborhoods with an upper bound size defined as $M = O(\log i)$. It uses Beam Search (BS) as a search algorithm; that is, it is based on storing candidates in a priority list of maximum size (beam size) and also controls what is inserted into the beam using a parameter $0 < \Delta < 2$; the result set is populated during the navigation, and the search finishes when the result set does not improve and the beam is empty. It supports single-pass automatic index optimization for a given quality score. As a baseline, it was constructed with 0.95 as objective recall and a neighborhood size of $M = \log_{1.2} i$ for Task 1. On the other hand, we use $M = \log_2 i$ for memory-limited tasks, i.e., 2 and 3. The database was also projected to reduce its memory footprint, using 96-dimensional PCA for Task 2 and 128-dimensional PCA for Task 3. Note that for tasks 2 and 3, the objective recall is achieved in the projected space and not in the original metric database. During the search

stage, we varied the self-optimized Δ parameter in the range $\Delta/1.05^2 \leq \Delta' < 2$ growing exponentially in a 1.05 factor.

The BL-Bruteforce is a parallelized exhaustive evaluation of all objects with queries. Task 1 is approached as a direct search over the 768-dimensional vectors using the cosine distance. In contrast, Task 3 uses the Euclidean distance over a 128-dimensional PCA projection of the dataset to fulfill the requirement of not touching the original vectors during the search stage. Task 2 is not tackled with this baseline.

3.2 Teams Solutions

Four teams submitted a candidate for evaluation; one team (HIOB) targeted all three tasks, two teams (DEGLIB and LMI) solved two tasks (Task 1 and 3),[10] and the remaining team (HSP) focused on efficient retrieval in the standard setting (Task 1). Teams used their own implementations or tuned well-known approximate nearest neighbor search libraries. We encouraged participants to use multithreading or multiprocessing in the construction and searching stages to take advantage of the hardware. We give a short overview over the approaches next and refer to the individual papers for more details.

DEGLIB [6]. The solution uses the Continuous Refining Exploratory Graph [5], which is a graph index using monotone metric approximation properties but also ensuring three properties: i) no redundant paths, ii) good starting points, iii) compression of the metric database (vectors under the cosine similarity). The submission is tuned to take advantage of low-recall requirements and work with low-memory limits. Task A projects the LAION 768-dimensional vectors into 512-dimensional vectors using their compression technique, while Task C compresses to dimension 64, ensuring working in a low-memory regime.

HIOB [20]. This approach continues the research on HIOB binary sketching presented at SISAP 2023 [19]. The current approach addresses the theoretical limits of Hamming spaces used as a proxy for the memory-effective search in complex spaces. The proposed index uses the HIOB sketches and groups their bits into integers to efficiently find solution candidates. Considering the available main memory size, the authors identify proper parameters for the sketching and grouping: the length of sketches is 384 bits, and sketches are indexed using 57 (overlapping) groups of bits, each covering 12 randomly selected bits of sketches. The authors observed a speed-up of about 17 times in comparison with the brute force search in Hamming space.

LMI [10]. The Learned Metric Index (LMI) is an updated version of the 2023 submission [15] to the indexing challenge. LMI clusters the dataset using spherical k-means and learns the cluster labels using an MLP. To cope with larger datasets that do not fit into memory, TruncatedSVD is employed to embed

[10] LMI submitted a solution for Task 2, but was wall-clocked after 36hrs of construction.

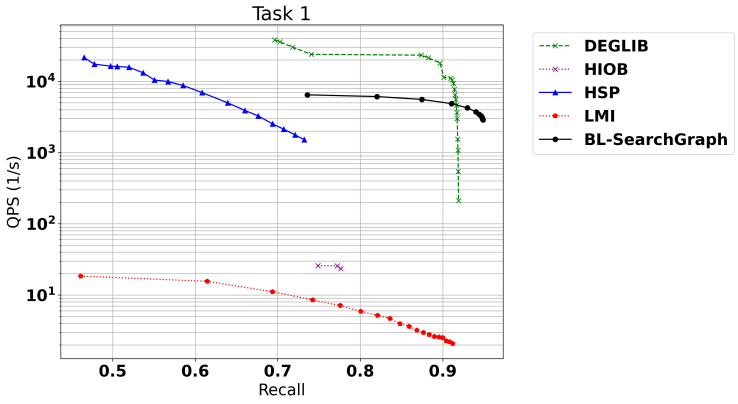

Fig. 1. Recall vs query throughput in our unrestricted indexing task under our private query set using 30 nearest neighbor queries. Up and to the right is better.

the vectors into a lower-dimensional space. Further improvements to the previous pipeline are query parallelism, improved sampling schemes during index creation, and better memory management.

HSP [4]. This submission proposes the construction of an approximate half-space proximal graph (HSP) with local monotonicity properties. Instead of applying the HSP test to all possible neighbors of a node v, which results in a construction time of $O(N^2)$ on a dataset collection with N vectors, it instead first finds an approximate neighborhood $R(v)$ and applies the test only locally. A hierarchy of HSP graphs is built recursively through sampling to enable fast construction time.

4 Results and Discussions

Figure 1 presents the results of the evaluation of Task 1 for our 10K private query set. For fixed search hyperparameters, it related the achieved quality (in terms of the average recall over all 10K queries) to the observed throughput measuring queries per second. As mentioned in §2, all teams built a single index and provided a list of at most 30 different search hyperparameters. Disregarding the baseline, and focusing on the quality in terms of the achieved recall, teams DEGLIB and LMI surpass the 0.8 recall limit specified by the task. At the same time, HIOB kept just behind the lower line, presumably due to the differences between public and private query sets and a possible overfitting of the hyperparameter setting. Team HSP did not provide search parameters that would provide a quality that exceeds the recall threshold. As clear from the plot, team DEGLIB's solution provides the best recall-throughput trade-off. It provides a speed-up of almost a factor 3 over the baseline SearchGraph at around .8 recall.

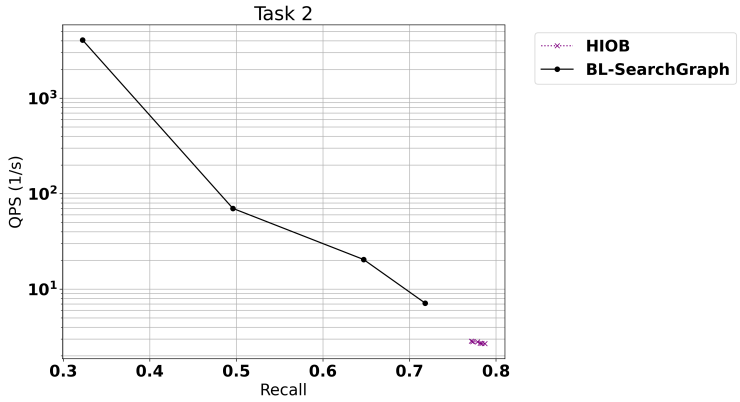

Fig. 2. Recall vs query throughput for the memory constrained task with reranking.

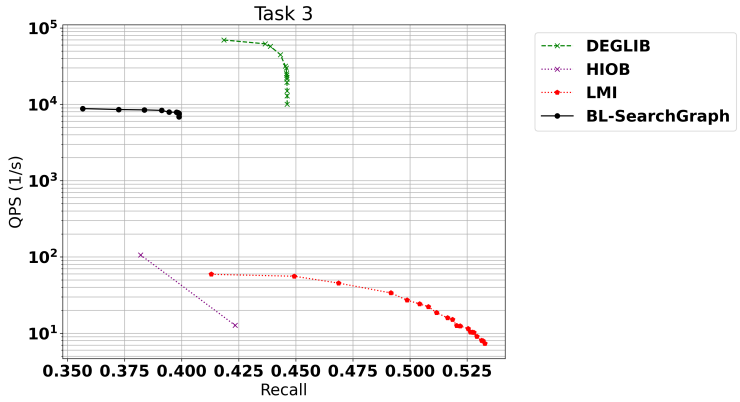

Fig. 3. Recall vs query throughput for our memory-restricted task without reranking, i.e., Task 3.

Figure 2 compares search indexes in the memory-constrained task with reranking using the private query set. Only HIOB and LMI teams submitted system solutions for this task, but LMI's run was terminated due to exceeding 24 hours of wall-clock time. The figure shows that neither HIOB nor our baselines achieved the requested average recall of 0.8. Even when undesired, this is a consequence of how private queries differ from public ones.

Finally, our memory-constrained task without reranking is evaluated in Table 3. Please remember that the challenge asks for fastest searching system solutions achieving a recall of at least 0.4. Here, we observe that DEGLIB, HIOB, and LMI improves the recall requirement; all of them improving on the baseline consisting of 128-dimensional PCA projection with Search Graph. Note that highest average recall is achieved by LMI while the best search performance is again achieved by DEGLIB; as in Task 1, the performance gain is remarkable.

Table 2. Summary of the evaluation process on the 100M database and the 10K private query set for Task 1, 2, and 3. Bold marks indicate best performing solutions under the challenge specifications; the strikethrough mark indicates that the recall limit was achieved with numerical round by the left, and ignored from the ranking. Build times are given in hours and minutes and query time in seconds.

	Task 1				Task 2				Task 3			
Team	Build time	Query time	Recall	Rank	Build time	Query time	Recall	Rank	Build time	Query time	Recall	Rank
DEGLIB	2h 44m	**0.4s**	0.87	1	–	–	–	–	1h 33m	**0.14s**	0.42	1
HIOB	13m	425.9s	0.78	–	33m	3,712.3s	0.79	–	33m	789.31s	0.42	3
HSP	11h 00m	6.6s	0.73	–	–	–	–	–	–	–	–	–
LMI	2h 07m	1,691.6s	0.80	3	>24h	–	–	–	3h 08m	169.04s	0.41	2
BL-SearchGraph	9h 20m	1.6s	0.82	2	4h 17m	1,401.7s	0.72	–	3h 21m	1.46s	~~0.40~~	–
BL-Bruteforce	–	$\sim 3\times10^4$s	1.0	4	–	–	–	–	–	3638.1s	0.41	4

Summary and Final Ranking. Table 2 summarizes the challenge results, showing the index construction time, the shortest search time that exceeded the recall requirement, and the actual recall achieved within that time. We also include the best runs for solutions that do not meet the requirements yet cannot get a rank position. We can observe that DEGLIB stands out in both Task 1 and Task 3, achieving the best performances. Task 2 is officially declared unmatched, but in practice, HIOB is better in this modality. HIOB also has a pretty low construction time, which is remarkable for the database size (100 million vectors). On the other hand, the LMI achieves high recall values in low-memory environments, but it holds high search times yet uses moderate building times. The HSP team also achieves high query throughput but does not satisfy the recall requirements.

5 Conclusions

The SISAP 2024 Indexing Challenge offered three tasks to assess indexing methods' performance under various operational constraints. Participants proposed approaches that effectively balanced search speed, result quality, memory usage, and indexing time when dealing with high-dimensional and large-scale metric datasets. The challenge inspired innovative methods, and the solutions exemplify carefully engineered data structures, efficient search algorithms, and feature projection techniques.

We are confident that the challenge contributes to future research and development efforts in approximate nearest neighbor searches. While the methods submitted and tested represent a significant step forward, there is still room for improvement in the fields of faster indexing and searching algorithms, more robust algorithms, and tackle the secondary memory challenge.

Acknowledgments. We thank Edgar Chavez for useful discussion during the planning phase of the challenge. The second co-author received funding from the Innovation Fund Denmark for the project DIREC (9142-00001B).

References

1. Aumüller, M., Bernhardsson, E., Faithfull, A.J.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Inf. Syst. **87** (2020)
2. Aumüller, M., Ceccarello, M.: Recent approaches and trends in approximate nearest neighbor search, with remarks on benchmarking. IEEE Data Eng. Bull. **46**(3), 89–105 (2023)
3. Chávez, E., Navarro, G., Baeza-Yates, R., Marroquín, J.L.: Searching in metric spaces. ACM Comput. Surv. **33**(3), 273–321 (sep 2001). https://doi.org/10.1145/502807.502808, https://doi.org/10.1145/502807.502808
4. Foster, C., Chavez, E., Kimia, B.: Top-down construction of locally monotonic graphs for similarity search. In: SISAP. Lecture Notes in Computer Science, Springer (2024)
5. Hezel, N., Barthel, K.U., Schall, K., Jung, K.: An exploration graph with continuous refinement for efficient multimedia retrieval. In: Proceedings of the 2024 International Conference on Multimedia Retrieval. p. 657–665. ICMR '24, Association for Computing Machinery, New York, NY, USA (2024). https://doi.org/10.1145/3652583.3658117, https://doi.org/10.1145/3652583.3658117
6. Hezel, N., Schilling, B., Barthel, K., Schall, K., Jung, K.: Adapting the exploration graph for high throughput in low recall regimes. In: SISAP. Lecture Notes in Computer Science, Springer (2024)
7. Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., Lewis, M.: Generalization through memorization: Nearest neighbor language models. In: ICLR. OpenReview.net (2020)
8. Li, W., Zhang, Y., Sun, Y., Wang, W., Li, M., Zhang, W., Lin, X.: Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement. IEEE Trans. Knowl. Data Eng. **32**(8), 1475–1488 (2020)
9. Matsui, Y.: https://github.com/matsui528/annbench
10. Procházka, D., Slanináková, T., Čerňanský, J., Olĺha, J., Antol, M., Dohnal, V.: Scaling learned metric index to 100m datasets. In: SISAP. Lecture Notes in Computer Science, Springer (2024)
11. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., Sutskever, I.: Learning transferable visual models from natural language supervision. In: ICML. Proceedings of Machine Learning Research, vol. 139, pp. 8748–8763. PMLR (2021)
12. Schuhmann, C., Beaumont, R., Vencu, R., Gordon, C., Wightman, R., Cherti, M., Coombes, T., Katta, A., Mullis, C., Wortsman, M., et al.: Laion-5b: An open large-scale dataset for training next generation image-text models. Advances in Neural Information Processing Systems **35**, 25278–25294 (2022)
13. Simhadri, H.V., Aumüller, M., Ingber, A., Douze, M., Williams, G., Manohar, M.D., Baranchuk, D., Liberty, E., Liu, F., Landrum, B., Karjikar, M., Dhulipala, L., Chen, M., Chen, Y., Ma, R., Zhang, K., Cai, Y., Shi, J., Chen, Y., Zheng, W., Wan, Z., Yin, J., Huang, B.: Results of the big ANN: neurips'23 competition. CoRR **abs/2409.17424** (2024)
14. Simhadri, H.V., Williams, G., Aumüller, M., Douze, M., Babenko, A., Baranchuk, D., Chen, Q., Hosseini, L., Krishnaswamy, R., Srinivasa, G., Subramanya, S.J., Wang, J.: Results of the neurips'21 challenge on billion-scale approximate nearest neighbor search. In: NeurIPS (Competition and Demos). Proceedings of Machine Learning Research, vol. 176, pp. 177–189. PMLR (2021)

15. Slaninákoová, T., Procházka, D., Antol, M., Olha, J., Dohnal, V.: Sisap 2023 indexing challenge – learned metric index. In: Proceedings of the 16th International Conference on Similarity Search and Applications (2023)
16. Tellez, E.S., Ruiz, G.: Similarity search on neighbor's graphs with automatic pareto optimal performance and minimum expected quality setups based on hyperparameter optimization. CoRR **abs/2201.07917** (2022), `https://arxiv.org/abs/2201.07917`
17. Tellez, E.S., Ruiz, G.: Similaritysearch.jl: Autotuned nearest neighbor indexes for julia. Journal of Open Source Software **7**(75), 4442 (2022)
18. Tellez, E.S., Aumüller, M., Chávez, E.: Overview of the SISAP 2023 indexing challenge. In: SISAP. Lecture Notes in Computer Science, vol. 14289, pp. 255–264. Springer (2023)
19. Thordsen, E., Schubert, E.: An alternating optimization scheme for binary sketches for cosine similarity search. In: Similarity Search and Applications: 16th International Conference, SISAP 2023, A Coruña Spain, October 9-11, Proceedings. Springer (2023)
20. Thordsen, E., Schubert, E.: Grouping sketches to index high-dimensional data in a resource limited setting. In: SISAP. Lecture Notes in Computer Science, Springer (2024)