Results of the NeurIPS'21 Challenge on Billion-Scale Approximate Nearest Neighbor Search

Harsha Vardhan Simhadri[1]   harshasi@microsoft.com
George Williams[2]   gwilliams@ieee.org
Martin Aumüller[3]   maau@itu.dk
Matthijs Douze[4]   matthijs@fb.com
Artem Babenko[5]   artem.babenko@phystech.edu
Dmitry Baranchuk[5]   dbaranchuk@yandex-team.ru
Qi Chen[1]   cheqi@microsoft.com
Lucas Hosseini[4]   lucas.hosseini@gmail.com
Ravishankar Krishnaswamy[1]   rakri@microsoft.com
Gopal Srinivasa[1]   gopalsr@microsoft.com
Suhas Jayaram Subramanya[6]   suhasj@cs.cmu.edu
Jingdong Wang[7]   wangjingdong@baidu.com

[1] Microsoft Research [2] GSI Technology [3] IT University of Copenhagen
[4] Meta AI Research [5] Yandex [6] Carnegie Mellon University [7] Baidu

Abstract

Despite the broad range of algorithms for Approximate Nearest Neighbor Search, most empirical evaluations of algorithms have focused on smaller datasets, typically of 1 million points (Aumüller et al., 2020). However, deploying recent advances in embedding based techniques for search, recommendation and ranking at scale require ANNS indices at billion, trillion or larger scale. Barring a few recent papers, there is limited consensus on which algorithms are effective at this scale vis-à-vis their hardware cost.

This competition[1] compares ANNS algorithms at billion-scale by hardware cost, accuracy and performance. We set up an open source evaluation framework[2] and leaderboards for both standardized and specialized hardware. The competition involves three tracks. The standard hardware track T1 evaluates algorithms on an Azure VM with limited DRAM, often the bottleneck in serving billion-scale indices, where the embedding data can be hundreds of GigaBytes in size. It uses FAISS (Johnson et al., 2017) as the baseline. The standard hardware track T2 additional allows inexpensive SSDs in addition to the limited DRAM and uses DiskANN (Subramanya et al., 2019) as the baseline. The specialized hardware track T3 allows any hardware configuration, and again uses FAISS as the baseline.

We compiled six diverse billion-scale datasets, four newly released for this competition, that span a variety of modalities, data types, dimensions, deep learning models, distance functions and sources. The outcome of the competition was ranked leaderboards of algorithms in each track based on recall at a query throughput threshold. Additionally, for track T3, separate leaderboards were created based on recall as well as cost-normalized and power-normalized query throughput.

Keywords: Approximate nearest neighbor search, large-scale search

---

1. https://big-ann-benchmarks.com
2. https://github.com/harsha-simhadri/big-ann-benchmarks/

1. Introduction

Approximate Nearest Neighbor Search or ANNS is a problem of fundamental importance to search, retrieval and recommendation. In this problem, we are given a dataset P of points along with a pairwise distance function, typically the d-dimensional Euclidean metric or inner product with d ranging from 50 to 1000. The goal is to design a data structure that, given a query q and a target k (or radius r), efficiently retrieves the k nearest neighbors of q (or all points within a distance r of q) in the dataset P according to the given distance function. In many modern-day applications of this problem, the dataset to be indexed and the queries are the output of a deep learning model (Babenko and Lempitsky, 2016; Devlin et al., 2018). The ANNS problem is widely studied in the algorithms, computer systems, databases, data mining, information theory and machine learning research communities, and numerous classes of algorithms have been developed. See, e.g., (Beygelzimer et al., 2006; Babenko and Lempitsky, 2014; Johnson et al., 2017; Weber et al., 1998; Baranchuk et al., 2018; Malkov and Yashunin, 2016; Jégou et al., 2011; Arya and Mount, 1993; Indyk and Motwani, 1998; Iwasaki and Miyazaki, 2018; Guo et al., 2020; Aumüller et al., 2019) for some recent works, and also the survey articles (Samet, 2006; Andoni and Indyk, 2008; Wang et al., 2018, 2021) comparing these techniques.

However, most of the research has focused on small to medium scale datasets of millions of vectors. For instance, the active gold-standard benchmark site (Aumüller et al., 2020) that compares almost all of the current-best ANNS algorithms *uses datasets no more than a million points each*, and design choices in the benchmarking system make it difficult to go beyond this scale. Implementing most existing state-of-art solutions for ANNS at this scale ends up being too expensive as the indices are very RAM intensive. Alternately, there are solutions such as SRS (Sun et al., 2014) and HD-Index (Arora et al., 2018) that can serve a billion-point index in a commodity machine, but these have high search latencies for achieving high search accuracy. Given the increasing relevance of search at billion+ scale, this competition aims to rigorously benchmark the *performance of billion-scale ANNS algorithms vis-à-vis their hardware cost.*

2. Tasks, Hardware Tracks, and Datasets

The main task of this challenge is to design a fast and accurate algorithm and/or system which can build and serve a billion-point index with minimal hardware cost. This scale is a good fit for comparing algorithmic ideas on a single machine. Due to the proliferation of deep-learning-based embeddings, such a system would immediately fit into a variety of application domains, including but not limited to web search, email search, document search, image search; ranking and recommendation services for images, music, video, news, etc. To test the applicability of the submitted solutions in these diverse use-cases, the competition ran evaluations on datasets representing these applications.

Query types. We distinguish the following two query types on a dataset $S \subseteq \mathbb{R}^d$:

- k-NN query: Given a query $q \in \mathbb{R}^d$ and an integer $k \geq 1$, return the k-nearest neighbors to the query point in S. A value of $k = 10$ was used for benchmarking.
- Range search: Given a query $q \in \mathbb{R}^d$ and a distance threshold R, return all points in S that are at distance at most R from q.

2.1. Hardware tracks

To provide a platform for the development of both algorithmic and systems innovation, the challenge introduces three different tracks: two standardized hardware tracks (T1 and T2) and one custom hardware track (T3).

Most current solutions to ANNS are limited in scale as they require data and indices to be stored in expensive main memory. To motivate the development of algorithms that use main memory effectively, we limit and normalize the amount of DRAM available in the standardized hardware tracks to 64GB. This is insufficient to store an uncompressed version of any of the datasets used in this competition, which range from 100GB to 800 GB in size.

Track T1. Search uses a Standard F32s_v2 Azure VM[3] with 32 vCPUs and 64GB main memory. 64GB. Index construction can use up to 4 days of time on Standard F64s_v2 machine with 64 vCPUs and 128GB main memory. Note that the main memory allowed for index construction is smaller than most billion-scale datasets. Algorithms must navigate these constraints by efficiently compressing and indexing data much larger than the main memory. We use the IVFPQ algorithm (Jégou et al., 2011) from the FAISS suite (Johnson et al., 2017) as our baseline. This track thus provides a platform for algorithmic innovation in efficient and accurate vector quantization methods, as well as compact indices.

Track T2. The second standardized hardware (T2) track incorporates an additional 1TB of inexpensive SSD to serve indices. This track allows more accuracy due to the allowance for storing uncompressed data – the SSD is larger than all datasets in the competition. This track provides a platform for algorithmic innovation for out-of-core indexing. Search uses a 8 vCPU Standard L8sv_2 Azure VM[4] which hosts a local SSD in addition to its 64GB main memory. Algorithms can use 1TB of the SSD to store the index and the data. Index constructions rules are the same as in Track T1. The baseline is an open source implementation[5] of DiskANN (Subramanya et al., 2019) which uses a hybrid DRAM-SSD index to achieve high recall and throughput.

Track T3. The specialized/custom hardware track T3 aims to encourage systems innovation to improve the cost and power-normalized performance. It allows the most flexibility in hardware and allows the use of any existing combination of hardware or even the development of custom hardware. This may include GPUs, reconfigurable hardware like FPGAs, custom accelerators available as add-on PCI board[6], and hardware not readily available on public cloud such as dedicated co-processors. Participants were required to either send the organizers an add-on PCI boards along with installation instructions, or if that is not possible, provide access to run validation scripts and docker containers on private hardware. The baseline used in this track was `IVF1048576,SQ8` from the FAISS suite (Johnson et al., 2017) running on a machine with 56-core Intel Xeon with an NVIDIA V100 GPU and 700GB of RAM.

---

3. https://docs.microsoft.com/en-us/azure/virtual-machines/fsv2-series

4. https://docs.microsoft.com/en-us/azure/virtual-machines/lsv2-series

5. https://github.com/microsoft/DiskANN/commit/4c7e9603324061916c64dee203343bdafab43a32

6. For examples, see https://www.amd.com/en/graphics/radeon-rx-graphics, https://www.apple.com/shop/product/HM8Y2VC/A/blackmagic-egpu, https://www.xilinx.com/, https://flex-logix.com/

Dataset	Type	Dim.	Distance	Query Type	Source
BIGANN	uint8	128	L2	k-NN	[Link]
Facebook SimSearchNet++	uint8	256	L2	Range	[Competition]
Microsoft Turing ANNS	float32	100	L2	k-NN	[Competition]
Microsoft SpaceV	int8	100	L2	k-NN	[Link]
Yandex DEEP	float32	96	L2	k-NN	[Link 1] [Link 2]
Yandex Text-to-Image	float32	200	Inner Product	k-NN	[Link]

Table 1: Summary of the six one billion point datasets used for benchmarking.

2.2. Datasets

The following datasets, summarized in Table 1, were released on the competition website or linked via websites where the data was originally published. The experimental framework manages working with (variants of) these datasets automatically.

- BIGANN contains SIFT image similarity descriptors applied to 1 billion images (Jegou et al., 2011) and is benchmark used by existing algorithms.

- Facebook SimSearchNet++ is a new dataset of image descriptors[7] used for copyright enforcement, content moderation, etc., released by Facebook for this competition. The original Vectors are compressed to 256 dimensions by PCA for this competition.

- Microsoft SpaceV1B is a new web relevance dataset released by Microsoft Bing for this competition. It consists of web documents and queries vectors encoded by the Microsoft SpaceV Superion model (Shan et al., 2021) to capture generic intent representation for both documents and queries.

- Microsoft Turing ANNS is a new web query similarity dataset released by the Microsoft Turing group for this competition. It consists of web search queries encoded by the universal language AGI/Spacev5 model trained to capture generic intent representation (Zhang et al., 2019) and uses the Turing-NLG architecture[8]. The query set also consists of web search queries, and the goal is to match them to the closest queries seen by the search engine in the past.

- DEEP1B consists of the outputs of the GoogleNet model for a billion images on the web, introduced in (Babenko and Lempitsky, 2016). This dataset is already used for benchmarking in the community.

- Yandex Text-to-Image is a new cross-modal dataset, where database and query vectors can potentially have different distributions in a shared representation space. The database consists of image embeddings produced by the Se-ResNext-101 model (Hu et al., 2018) and queries are textual embeddings produced by a variant of the DSSM model (Huang et al., 2013). The mapping to the shared representation space is learned via minimizing a variant of the triplet loss using click-through data.

---

7. https://ai.facebook.com/blog/using-ai-to-detect-covid-19-misinformation-and-exploitative-content

8. https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

File formats were made as uniform as possible, taking into account the fact that some datasets are in `float32` and others are in `uint8`. For each dataset, we provide:

- a set of 1 billion database vectors to index;
- a set of query vectors for validation (at least 10 000 of them);
- a set of query vectors held out for the final evaluation (the same size);
- ground truth consisting of the 100 nearest neighbors for each query in the validation set, including results at the reduced scales of 10M and 100M vectors.

3. Evaluation

3.1. Metrics

As is the general practice in benchmarking ANNS algorithms (see Aumüller et al. (2020)), we evaluate implementations based on the quality-performance tradeoff they achieve. For each dataset, participants could provide one configuration for index building and up to 10 different sets of search parameters. Each set of search parameters is intended to strike a different trade-off in terms of accuracy vs. search time. At evaluation time, the sets of search parameters were evaluated in turn, recording the search accuracy and througput achieved. To obtain a single value for the leaderboard, we defined a threshold on the throughput achieved and score the participants by the accuracy of the results.

Throughput. We report the query throughput obtained using all the threads available on the standardized machine. All queries are provided at once, and we measure the wall clock time between the ingestion of the vectors and when all the results are output. The resulting measure is the number of queries per second (QPS).

Search accuracy. We use two notions of search accuracy, defined as follows, depending on the dataset. For scenarios require k-NN search, we measure 10-recall@10, *i.e.* the number of true 10-nearest neighbors found in the $k = 10$ first results reported by the algorithm.

A range search returns a list of database items whose length is not fixed in advance (unlike k-NN search). We compute the ground truth range search results. The range search accuracy measure is the precision and recall of the algorithm's results w.r.t. the ground truth results. Accuracy is then defined as the mean average precision over recall values when clipping the result list with different values of the threshold.

Power and Cost. In the T3 track, in addition to search accuracy and throughput, we ranked participants on two additional benchmarks related to power and cost. For power, we leveraged standard power monitoring interfaces available in data-center grade systems to obtain KiloWatt-hour/query (or Joule/query) for the participant's algorithm and hardware. For cost, we used a capacity planning formula based on horizontally replicating the participant's hardware to achieve 100 000 queries/second for a period of 4 years. The cost is the product of the number of machines required to serve at 100 000 QPS and the total cost per machine, including both the manufactured suggested retail price of the hardware as well as the cost of power consumption based on a global average of \$0.10/kWh.

Synthetic performance measure. Participants obtained several tradeoffs in the QPS-accuracy space. On the challenge page and in Section 4, we report these tradeoff plots. While such plots give the most accurate overview of an implementation's performance, the leaderboard reports a synthetic scalar metric to rank the participants. For each track, we fix a given QPS target, and find the maximum accuracy an algorithm can get with at least as many QPS from the Pareto-optimal curve. The QPS targets are calibrated on the baseline methods described in Section 2.1. The thresholds for the competition are 10 000 with 32 vCPUs(T1), 1 500 with 8vCPUs (T2), and 2 000 (T3). The score for the recall leaderboard is the sum of improvements in recall over the baseline over all datasets. An algorithm is expected to submit entries for at least three algorithms to be considered for the leaderboard.

3.2. Evaluation Framework

We provide a standard benchmarking framework using and extending the techniques in the evaluation framework Aumüller et al. (2020). The framework takes care of downloading and preparing the datasets, running the experiments for a given implementation, and evaluating the result of the experiment in terms of the metrics mentioned above. Adding an implementation consist of three steps. First, it requires the implementation to provide a Dockerfile to build the code and set up the environment. Second, a Python wrapper script that calls the correct internal methods to build an index and run queries against the index must be provided. A REST-based API exists to work with a client/server-based model in case a Python wrapper is not available.[9] Lastly, authors provide a set of index build parameters (one per dataset) and query parameters (maximum of ten per dataset).

3.3. Evaluation Process

The evaluation process differed slightly among the three different tracks.

Track 1 and 2. We provided Azure compute credit to participants to work with virtual machine SKUs that were used in the final evaluation. Participants sent in their code via a pull requests to the evaluation framework. The organizers used this code to build indices on Azure cloud machines for the datasets pointed out in the pull request, and run queries the public query set. Organizers then provide these results to the participants, and iterated on fixing problems until the participants agreed with the results reported here. As a result of this process, all implementations, configurations and conversations with participants are publicly available via the pull requests linked in Tables 2 and 3.

Track 3. Since this track allowed participants to work with alternative hardware configuration, organizers were given (usually remote) access to the machine and the framework was run remotely on these machines. Some of the implementations participating in this track are not publicly available.

4. Outcome of the Challenge

A total of 30 teams expressed interest in this contest. Eventually, we received 13 submissions that participated in the final evaluation: 5 submissions for T1, 3 submissions for T2, and 5 submissions for T3. Winners mentioned below gave a short presentation of their approach during NeurIPS break-out session; these talks are shared on the competition website.

9. We thank Alex Klibisz for his work on this API.
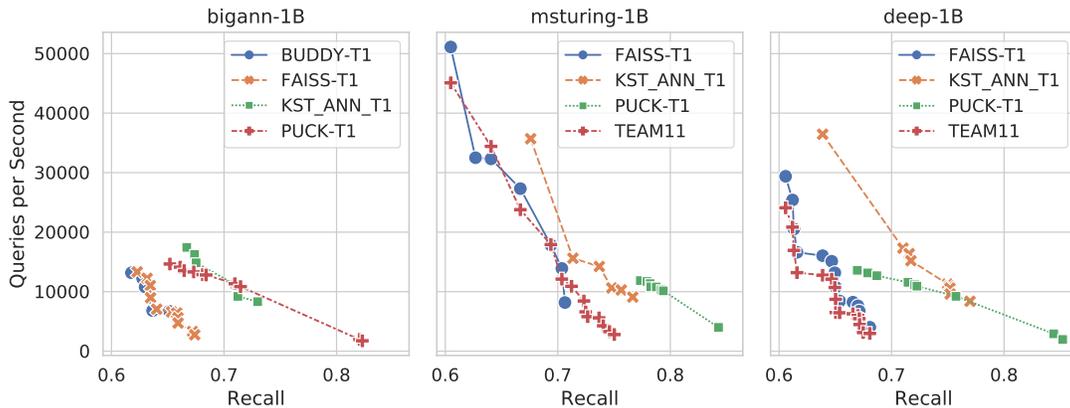
Figure 1: Selection of results for track 1. QPS-recall tradeoff; the QPS-cutoff was 10 000.

Algorithm	BIGANN	DEEP	MS SPACEV	MS Turing	SSN++	Text-to-Image
Baseline	0.6345	0.6503	0.7289	0.7036	0.7538	0.0693
team11		0.6496		0.7122		
puck-t1	0.7147	0.7226		0.7938*		0.1610*
ngt-t1						
kst_ann_t1	0.7122	0.7122	0.7645	0.7564		
buddy-t1	0.6277					

Table 2: Leaderboard for Track T1. Recall/AP achieved at 10000 QPS on Azure F32v2 VM with 32 vCPUs. * indicates entries submitted after the competition closed.

4.1. Track 1

The results of this track are summarized in Table 2. In this track the whole index had to fit into 64GB of RAM, which limits the accuracy. The challenge was to invent more efficient compression/quantization schemes. We see that most entries were able to improve on the baseline, and figure 1 gives a more detailed overview of the performance on the datasets where most solutions offered improvements. As we can see from the plot, for most implementations the cutoff at 10 000 QPS is in a region where the loss in accuracy due to quantization saturates and no big improvements seem possible. The entry PUCK-T1 provided query settings that provided better recall than other approaches, but is much below the QPS threshold, and a few of the entries were submitted after the competition deadline. However, it shows the potential of their approach and presents a promising direction to pursue. On the other hand, Text-to-Image proved to be a particularly challenging dataset for the quantization/compression-based methods in this track because the query distribution and base distribution are completely different, with one coming from text embeddings and the other from image embeddings. This again presents an interesting and important research direction to pursue. In summary, since the starred results for PUCK-T1 in Table 2 have been obtained after the competition deadline, the winner of this track is:

KST-ANN-T1, *Li Liu, Jin Yu, Guohao Dai, Wei Wu, Yu Qiao, Yu Wang, Lingzhi Liu*, Kuaishou Technology and Tsinghua University, China.

Algorithm	BIGANN	DEEP	MS SPACEV	MS Turing	SSN++	Text-to-Image
baseline	0.9491	0.9371	0.9010	0.9356	0.1627	0.4885
kota-t2	0.9509		0.9040	0.9398	0.1821	
ngt-t2						
bbann			0.7602		0.8857	0.4954

Table 3: Leaderboard for Track T2. Recall/AP achieved at 1500 QPS on Azure Ls8v2 VM.
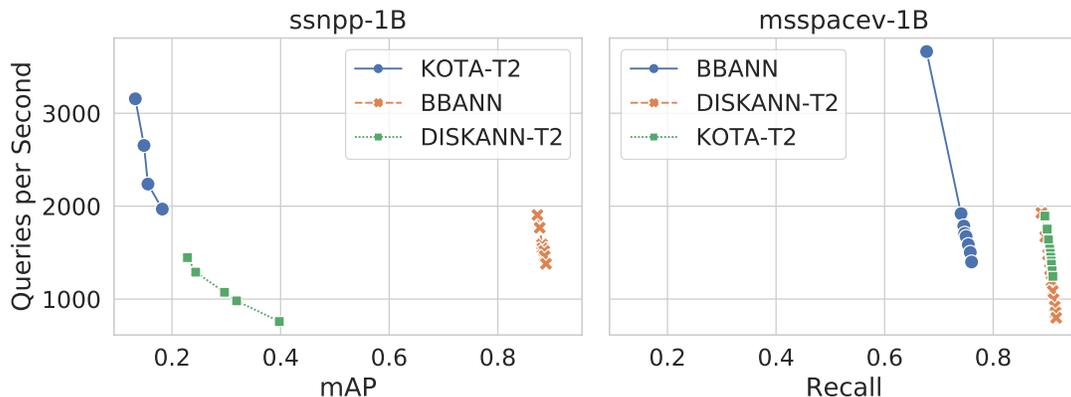
Figure 2: Selection of results for track 2. QPS-accuracy tradeoff; the QPS-cutoff was 1 500.

4.2. Track 2

The results of this track are summarized in Table 3 and more detailed plots for a couple of datasets are provided in Figure 2. This track allowed participants to use a SSD large enough to store the original vectors, thus allowing for better accuracy than in Track 1. There were only two approaches except the baseline. We suspect that this is due to the short timeframe of the challenge, since external memory implementations require more care than the in-memory approaches in Track 1. Among these two competitors, BBANN provided a huge improvement on the SSNPP dataset that required range-search queries using a hybrid graph-inverted index data structure. While this design showed recall regression on other datasets using a k-NN recall metric, the improvements outweighed the regression, leading to them being the top of the leaderboard for Track 2:

> **BBANN**, *Xiaomeng Yi, Xiaofan Luan, Weizhi Xu, Qianya Cheng, Jigao Luo, Xiangyu Wang, Jiquan Long, Xiao Yan, Zheng Bian, Jiarui Luo, Shengjun Li, Chengming Li*, Zilliz and Southern University of Science and Technology, China.

4.3. Track 3

Track 3 submissions offered by far the largest improvement over the baseline. NVidia submitted algorithms that ran on an A100 8-GPU system, while Intel submitted their algorithm, OptaNNE GraphNN, an adaptation of DiskANN leveraging Optane non-volatile

Recall achieved at a minimum of throughput 2K QPS						
Algorithm	DEEP	BIGANN	MS Turing	MS SpaceV	Text-to-image	SSN++
baseline	0.94275	0.93260	0.91322	0.90853	0.86028	0.97863
OptaNNE GraphNN	0.99882	0.99978	0.99568	0.99835	0.97340	-
CUANNS IVFPQ	0.99543	0.99881	0.988993	0.99429	0.94692	-
CUANNS MultiGPU	0.99504	0.99815	0.98399	0.98785	-	-
Querries per second at 90% recall						
Algorithm	DEEP	BIGANN	MS Turing	MS SpaceV	Text-to-image	SSN++
baseline	44464	3271	2845	3265	1789	5699
CUANNS MultiGPU	8016944	747421	584293	839749	-	-
OptaNNE GraphNN	1965446	335991	161463	157828	17063	-
CUANNS IVFPQ	91701	80109	109745	108302	19094	-
Joule/query achieved at minimum of 2K QPS and 0.9 recall						
Algorithm	DEEP	BIGANN	MS Turing	MS SpaceV	Text-to-image	SSN++
baseline	0.1117	0.1576	0.1743	0.1520	0.1128	0.0904
OptaNNE GraphNN	0.00441	0.0022	0.0048	0.0049	0.0446	-
CUANNS IVFPQ	0.0112	0.0112	0.0119	0.0090	0.0480	-
CUANNS MultiGPU	0.0029	0.0024	0.0049	0.0023	-	-
Total cost to horizontally replicate a system to serve 100 000 queries per second						
Algorithm	DEEP	BIGANN	MS Turing	MS SpaceV	Text-to-image	SSN++
baseline	545.6	737.9	853.9	735.9	1272.7	428.1
OptaNNE GraphNN	16.1	15.4	16.3	16.4	103.6	-
CUANNS IVFPQ	303.9	304.2	153.2	153.2	916.8	-
CUANNS MultiGPU	569.1	569.2	286.9	398.2	1213.8	629.4

Table 4: Recall, Throughput, Power and Cost Leaderboards for Track T3. Participant rank is preserved in the table for the non-baseline participants. Organizer submitted entries are not shown. All rankings for all submissions as well as performance on individual datasets are detailed on the competition and github site.

memory. Competition organizers also submitted entries which included the baseline submitted by Meta (formerly Facebook FAISS) for a V100 1-GPU system with 700GB RAM, GSI Technology submitted an algorithm that ran on their in-SRAM PCI accelerator in a system with 1TB RAM, and Microsoft submitted DiskANN which ran on a standard Dell server. The competition github site has more information about these systems.

Table 4 summarizes the results of the competition. The OptaNNE GraphANN implementation scored the highest on the recall and cost- and power-normalized leaderboards. The CUANNS MultiGPU implementation scored the highest on the throughput leaderboard. We did not combine the four different benchmark to determine one winner. We instead chose to name Intel and NVidia the co-winners of this track:

OptaNNE GraphNN, *Sourabh Dongaonkar (Intel Corporation), Mark Hildebrand (Intel Corporation / UC Davis), Mariano Tepper (Intel Labs), Cecilia Aguerrebere (Intel Labs), Ted Willke (Intel Labs), Jawad Khan (Intel Corporation)*

CUANNS MultiGPU, *Akira Naruse (NVIDIA), Jingrong Zhang (NVIDIA), Mahesh Doijade (NVIDIA), Yong Wang (NVIDIA), Hui Wang (Xiamen University), Harry Chiang (National Tsing Hua University)*

5. Conclusion, Lessons learned, and Outlook

With this challenge, we started a principled and reproducible research environment for the design and the evaluation of nearest neighbor implementations. In all three tracks we received implementations that were able to beat the well-designed baseline, which shows that improvements in the state-of-the-art are in reach even in a short time span.

Lessons learned. As mentioned by several participants, the design, implementation, and iterative engineering of nearest neighbor search methods usually takes time that exceeds the short time span of such a challenge. For this reason, the evaluation framework, which is now well established, will continue to accept new submissions. Moreover, the evaluation process is time consuming because of the scale of the problem. Some additional software development is required to further automate the evaluation and ensure organizers are not the bottleneck in the evaluation process. The cost related to building indices in the cloud is a limit to frequently re-running the entire evaluation, and could also be a bottleneck for non-industry affiliated teams when thinking of going beyond billion-scale datasets.

Future Directions. Possible future directions were discussed at an open forum during the NeurIPS break-out session and documented on the evaluation framework [10]. Some questions and directions for new tasks include:

- Support ANNS queries which also allow filters such as date range, author, language, image color or some combination of such attributes. See, for example, (Wei et al., 2020), a candidate algorithm for the problem.

- Design algorithms whose accuracy and performance is robust to insertions and deletions. A possible strong baseline is Fresh-DiskANN (Singh et al., 2021).

- Design algorithms that are robust to datasets with out-of-distribution queries such as those arising from cross-modal embeddings.

- Design compression with lesser information loss, perhaps at the price of more expensive decoding.

Outlook. We invite researchers and practitioners to use our framework as a starting point, and look forward to contributions of new datasets and algorithms on an ongoing basis. The website https://big-ann-benchmarks.com/ will serve as the main interface, and we plan to repeat the challenge in the future, possibly with additional directions.

Acknowledgements

---

10. https://github.com/harsha-simhadri/big-ann-benchmarks/issues/90

References

Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, January 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327494. URL http://doi.acm.org/10.1145/1327452.1327494.

Akhil Arora, Sakshi Sinha, Piyush Kumar, and Arnab Bhattacharya. Hd-index: Pushing the scalability-accuracy boundary for approximate knn search in high-dimensional spaces. *Proceedings of the VLDB Endowment*, 11, 04 2018. doi: 10.14778/3204028.3204034.

Sunil Arya and David M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '93, pages 271–280, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics. ISBN 0-89871-313-7. URL http://dl.acm.org/citation.cfm?id=313559.313768.

Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Michael Vesterli. PUFFINN: parameterless and universally fast finding of nearest neighbors. In *ESA*, volume 144 of *LIPIcs*, pages 10:1–10:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87, 2020. ISSN 0306-4379. URL http://www.sciencedirect.com/science/article/pii/S0306437918303685. http://ann-benchmarks.com.

Artem Babenko and Victor S. Lempitsky. Additive quantization for extreme vector compression. In *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*, pages 931–938. IEEE Computer Society, 2014. doi: 10.1109/CVPR.2014.124. URL https://doi.org/10.1109/CVPR.2014.124.

Artem Babenko and Victor S. Lempitsky. Efficient indexing of billion-scale datasets of deep descriptors. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2055–2063, 2016. doi: 10.1109/CVPR.2016.226. URL https://doi.org/10.1109/CVPR.2016.226.

Dmitry Baranchuk, Artem Babenko, and Yury Malkov. Revisiting the inverted indices for billion-scale approximate nearest neighbors. In *The European Conference on Computer Vision (ECCV)*, September 2018.

Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *Proceedings of the 23rd International Conference on Machine Learning*, ICML '06, page 97–104, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595933832. doi: 10.1145/1143844.1143857. URL https://doi.org/10.1145/1143844.1143857.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL http://arxiv.org/abs/1810.04805.

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

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338, 2013.

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*, STOC '98, pages 604–613, New York, NY, USA, 1998. ACM. ISBN 0-89791-962-9. doi: 10.1145/276698.276876. URL http://doi.acm.org/10.1145/276698.276876.

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

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, January 2011. doi: 10.1109/TPAMI.2010.57. URL https://hal.inria.fr/inria-00514462.

Herve Jegou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. Searching in one billion vectors: Re-rank with source coding. In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP 2011, May 22-27, 2011, Prague Congress Center, Prague, Czech Republic*, pages 861–864, 2011. doi: 10.1109/ICASSP.2011.5946540. URL https://doi.org/10.1109/ICASSP.2011.5946540.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017.

Yury A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *CoRR*, abs/1603.09320, 2016. URL http://arxiv.org/abs/1603.09320.

Hanan Samet. *Foundations of multidimensional and metric data structures*. Morgan Kaufmann, 2006.

Xuan Shan, Chuanjie Liu, Yiqian Xia, Qi Chen, Yusi Zhang, Kaize Ding, Yaobo Liang, Angen Luo, and Yuxiang Luo. Glow: Global weighted self-attention network for web search. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 519–528. IEEE, 2021.

Aditi Singh, Suhas Jayaram Subramanya, Ravishankar Krishnaswamy, and Harsha Vardhan Simhadri. Freshdiskann: A fast and accurate graph-based ANN index for streaming

similarity search. *CoRR*, abs/2105.09613, 2021. URL https://arxiv.org/abs/2105.09613.

Suhas Jayaram Subramanya, Fnu Devvrit, Rohan Kadekodi, Ravishankar Krishnawamy, and Harsha Vardhan Simhadri. Diskann: Fast accurate billion-point nearest neighbor search on a single node. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13748–13758, 2019. URL http://papers.nips.cc/paper/9527-rand-nsg-fast-accurate-billion-point-nearest-neighbor-search-on-a-single-node.

Yifang Sun, Wei Wang, Jianbin Qin, Ying Zhang, and Xuemin Lin. Srs: Solving c-approximate nearest neighbor queries in high dimensional euclidean space with a tiny index. *Proc. VLDB Endow.*, 8(1):1–12, September 2014. ISSN 2150-8097. doi: 10.14778/2735461.2735462. URL https://doi.org/10.14778/2735461.2735462.

Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. A survey on learning to hash. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):769–790, 2018. doi: 10.1109/TPAMI.2017.2699960. URL https://doi.org/10.1109/TPAMI.2017.2699960.

Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *CoRR*, abs/2101.12631, 2021. URL https://arxiv.org/abs/2101.12631.

Roger Weber, Hans-Jörg Schek, and Stephen Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, VLDB '98, pages 194–205, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1-55860-566-5. URL http://dl.acm.org/citation.cfm?id=645924.671192.

Chuangxian Wei, Bin Wu, Sheng Wang, Renjie Lou, Chaoqun Zhan, Feifei Li, and Yuanzhe Cai. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. *Proc. VLDB Endow.*, 13(12):3152–3165, aug 2020. ISSN 2150-8097. doi: 10.14778/3415478.3415541. URL https://doi.org/10.14778/3415478.3415541.

Hongfei Zhang, Xia Song, Chenyan Xiong, Corby Rosset, Paul N. Bennett, Nick Craswell, and Saurabh Tiwary. Generic intent representation in web search. *CoRR*, abs/1907.10710, 2019. URL http://arxiv.org/abs/1907.10710.