Evaluating Hybrid Approximate Nearest Neighbor Indexing and Search (HANNIS) for High-dimensional Image Feature Search

M M Mahabubur Rahman
Computer Science
Texas State University
San Marcos,TX,USA
toufik@txstate.edu

Jelena Tešić
Computer Science
Texas State University
San Marcos,TX,USA
jtesic@txstate.edu

2022 IEEE International Conference on Big Data (Big Data) | 978-1-6654-8045-1/22/$31.00 ©2022 IEEE | DOI: 10.1109/BigData55660.2022.10021048

***Abstract*—In this paper, we evaluate the performance of a novel method for efficient and effective retrieval of similar high-dimensional image features. The proposed solution —- hybrid approximate nearest neighbor indexing and search (HANNIS) —- retrieves truly similar items in the database, even if the retrieval set is large. This approach enables us to load items that are truly close to the incoming query at retrieval time. HANNIS outperforms all state-of-the-art methods in terms of recall, precision, and F1 score at depths of up to 100 and offers the fastest index loading and consistent retrieval performance.**

***Index Terms*—large set retrieval; high-dimensional indexing and search; deep descriptors; big data**

I. Introduction

The k-nearest neighbors (k-NN) search identifies the top k nearest neighbors to the query and performs very well for retrieving exact solutions in smaller data sets with a lower dimension. However, the k-NN search can be sluggish in large datasets and higher dimensions because of the "curse of dimensionality".

To address this problem, several approximate nearest-neighbor (ANN) methods were introduced. ANN methods work very fast but sacrifice some accuracy by loosening the condition of exact nearest-neighbor retrieval. ANN methods handle the "curse of dimensionality" well and thus have superior performance over the k-NN search in large datasets with higher dimensions. Many effective ANN solutions have been proposed in the past decade and can be grouped as graph-based, hashing-based, and partition-based methods.

Some libraries [1]–[4] have used the graph-based Hierarchical Navigable Small World (HNSW) algorithm. Among all ANN methods, the HNSW algorithm [1] shows promising results in terms of retrieval time [5]. However, there is still room for improvement. Moreover, all of the above libraries build a large index that often takes a long time to load the index from memory during nearest neighbor retrieval.

In this paper, we show that the hybrid approximate nearest neighbor indexing and search (HANNIS) approach outperforms existing state-of-the-art ANN method libraries built on the HNSW algorithm. In Section II we discuss the indexing and searching procedure in detail. In Section III, we discuss our experimental results in detail.

II. Methodology

In this section, we present our proposed method, which is divided into two sections.

a) ***Index building****:* To reduce the search space and accelerate the index loading time, we first subdivide the entire data space into n different partitions of $C_1, C_2, C_3, ..., C_n$ 1. For this work, we have used the k-means++ algorithm, and the centroid information for each cluster is stored and passed to the next phase to build the HNSW index. Once the groups have been calculated, each group is arranged into a hierarchical layer of proximity graphs (Fig. 2). The starting point for building an index is chosen randomly, and a heuristic is used to select the neighbors. Finally, all indexes are stored with their centroid information so that only a certain index can be loaded based on the query, which in turn saves the index loading time.

b) ***Improving effectiveness of retrieval****:* Fig. 3 shows the k -NN retrieval approach. During the retrieval, the index with the smallest distance from the query to the centroid is loaded for searching. The search within an index starts with the centroid in the upper layer where the edges are the longest, and then a greedy search is used within that layer until it reaches a local minimum (Fig. 3). Then the search switches to the lower layer starting with the previous local minimum, and this process continues until the query is reached and the top k-NN to the top k is returned. For multi-index search, each index returns its top k-NN to the query. Then, all the retrieval results are sorted on the basis of their distance to the query. Finally, k -NN are chosen as the final retrieval result.

III. Experiments

For all of our experiments, we have used the DEEP10M benchmark data set with 10 million instances and 96 dimensions [6], SIFT10M with 10 million instances and 128 dimensions [7], and the DOTA 2.0 data set with 1,024 dimensions and 2,7 million instances [8].

Performance analysis We measured the performance of our HANNIS method with four different state-of-the-art methods built on the HNSW algorithm [1]–[4] and are evaluating the method most suitable for similar image feature retrieval. Our criteria is that recall@k,precision@k, and f1-score@k need to

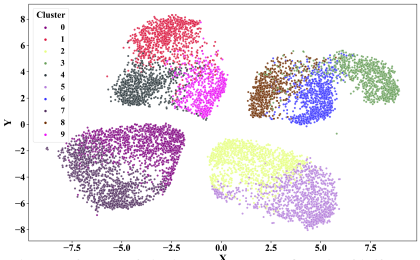

Fig. 1: Clustering with kmeans++ for building indexes for each cluster

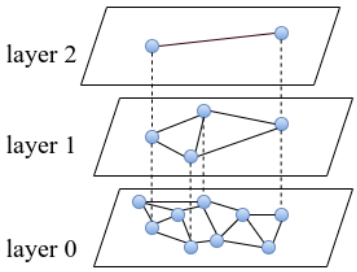

Fig. 2: Indexing with adapted HNSW approach

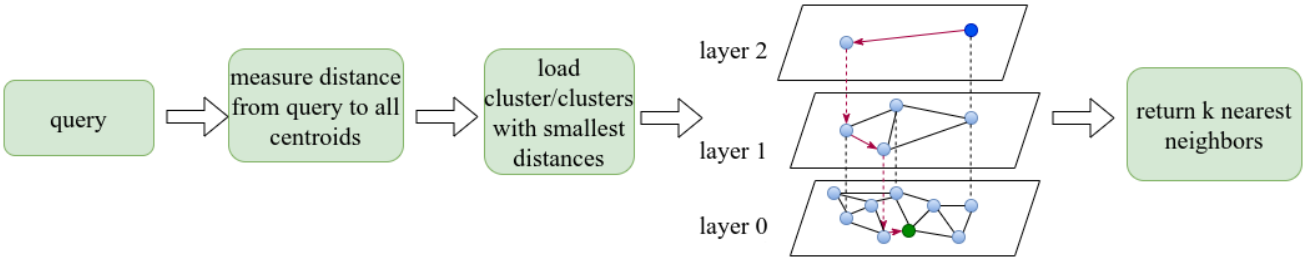

Fig. 3: Proposed HANNIS retrieval pipeline.

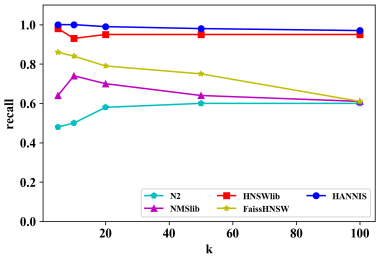

Fig. 4: Recall@k for DOTA 2.0

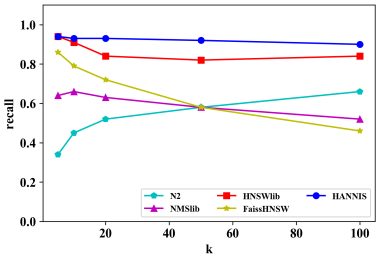

Fig. 5: Recall@k for SIFT

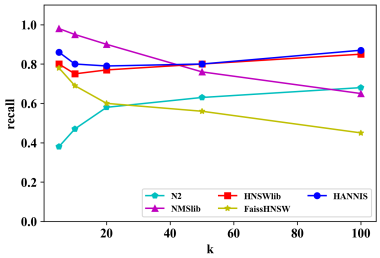

Fig. 6: Recall@k for Deep10M

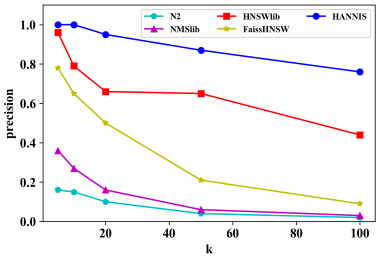

Fig. 7: Precision@k for DOTA 2.0

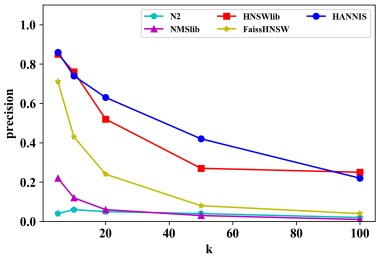

Fig. 8: Precision@k for SIFT

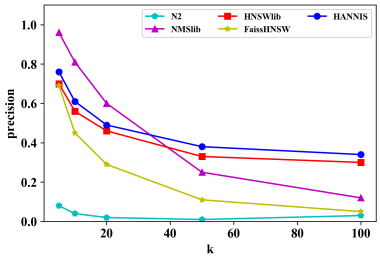

Fig. 9: Precision@k for Deep10M

stay consistent as k increases while keeping the index loading time and retrieval time comparable to the state of the art.

Recall@k-retrievals demonstrates HANNIS is consistently dominating in recall at **all** $k \in [5, 10, 20, 50, 100]$ for DOTA 2.0 (Fig. 4) and the SIFT data set (Fig. 5), and consistently better than NMSlib for $k > 40$ in (Fig. 6) with smaller index loading times (Fig. 13) at the price of much higher retrieval time (Fig. 14). We also observe interesting behavior of N2 for all three datasets: the recall of the indexing method is *improving* with larger k at a low index loading time (Fig. 13) and low retrieval time (Fig. 14). However, it has the poorest performance amongst all methods at lower k (Fig. 4, 5, 6). FaissHNSW performance quickly degrades for $k > 5$ for all three data sets (Fig. 4, 5, 6), and we do not consider the method appropriate for this scenario.

Precision@k-retrievals shows HANNIS out-performs all the methods at **all** $k \in [5, 10, 20, 50, 100]$ for DOTA 2.0 in Fig. 7. We also observe HNSWlib in Fig. 8: the effectiveness of the indexing method is *improving* with a larger k and exceeded the effectiveness of HANNIS for $k = 100$ with a longer index loading time (Fig. 13) and a low retrieval time (Fig. 14). NMSlib shows an interesting trend for the DEEP10M dataset in Fig. 9: the indexing method has high precision among all the methods at lower k, but the performance degrades quickly as k increases. HANNIS performs marginally better than HNSWlib for the DEEP10M dataset at **all** k in Fig. 9 with a faster index loading time (Fig. 13) and a longer retrieval time (Fig. 14). N2 has consistently low precision for all three datasets in Fig.. 7, 8 and 9 with a lower index loading time (Fig. 13) and a lower retrieval time (Fig. 14).

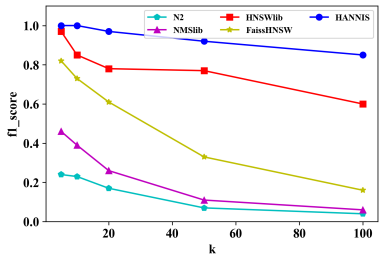

Fig. 10: F1-score@k for DOTA 2.0

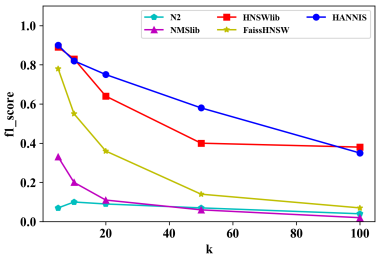

Fig. 11: F1-score@k for SIFT

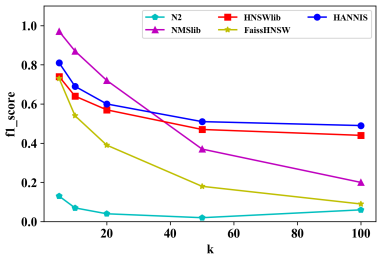

Fig. 12: F1-score@k for Deep10M

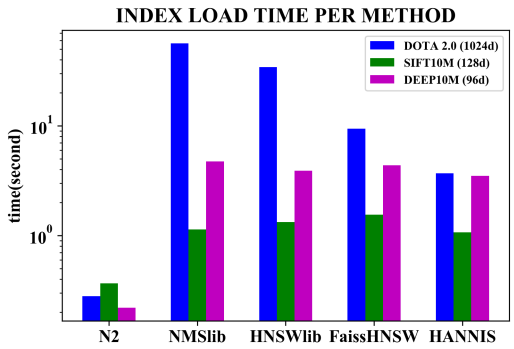

Fig. 13: Index loading time for 5 approaches.

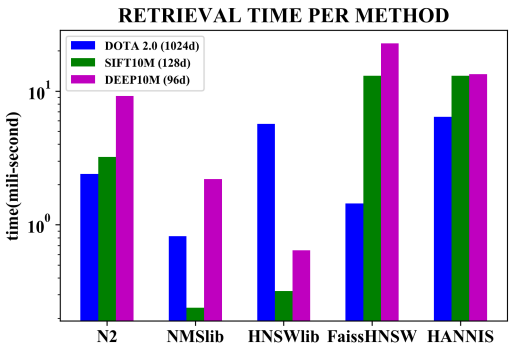

Fig. 14: Retrieval time for 5 approaches.

F1-score@k-retrievals demonstrates the dominating performance of HANNIS for all the three datasets in Fig. 10, 11 and 12 at higher k. We see an exception in Fig. 12 for NMSlib where the indexing method has the highest F1 score among all methods for $k < 40$, but its performance degrades at higher k. N2 and FaissHNSW have a consistently lower F1 score than HANNIS for all three datasets in Fig. 10, 11 and 12. HNSWlib performs marginally better than HANNIS at $k = 100$ in Fig. 11, in all other cases (Fig. 10, 11 and 12), HANNIS performs better.

Index load and Retrieval time We analyze the five methods index load timing and retrieval timing on a log scale *per method* to uncover any common trends. DOTA2.0 is approximately 3 times the size of Deep10M and nine times the size of SIFT10M. Fig. 13 compares the index loading time, and NMSlib, HNSWlib, FaissHNSW, and HANNIS index load time is proportional to the size of the dataset, and HANNIS has the best overall index load timing. N2 has the lowest index loading time, and it does not correlate to data set size in instances and in feature dimension, and type. Fig. 14 compares retrieval time per method for $k = 100$. For N2, FaisHNSW, and HANNIS methods, retrieval time corresponds to the number of instances in the dataset, and HANNIS seems to do better on high-dimensional descriptors than the comparable methods. For HNSWlib, the retrieval time is directly proportional to the dataset size, and it is hard to interpret the rule for NMSlib methods.

IV. Conclusion

The brute-force k-Nearest-Neighbor Search in large and high-dimensional data is computationally expensive. In this paper, we have demonstrated the efficiency of our approximate nearest-neighbor indexing and search method (HANNIS) to index and search for similar image features from a high-dimensional deep-descriptor data set. HANNIS outperforms the state-of-the-art libraries built on the HNSW algorithm in terms of recall, precision, and F1 score up to 100. HANNIS offers up to 18 times faster index loading into the memory, and the retrieval times are compatible with state-of-the-art libraries.

Acknowledgement

This work is partially supported by the NAVAIR SBIR N68335-18-C-0199. The views, opinions, and/or findings contained in this article are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government paper references.

References

[1] 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, 2018.

[2] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.

[3] GeonHee Lee. *TOROS N2 - lightweight approximate Nearest Neighbor library which runs fast even with large datasets*, 2017. Python package version 0.1.7.

[4] Leonid Boytsov and Bilegsaikhan Naidan. Engineering efficient and effective non-metric space library. In *International Conference on Similarity Search and Applications*, pages 280–293. Springer, 2013.

[5] Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. *Information Systems*, 87:101374, 2020.

[6] Dmitry Baranchuk, Artem Babenko, and Yury Malkov. Revisiting the inverted indices for billion-scale approximate nearest neighbors. *CoRR*, abs/1802.02422, 2018.

[7] Xiping Fu, Brendan McCane, Steven Mills, Michael Albert, and Lech Szymanski. Auto-jacobin: Auto-encoder jacobian binary hashing. *CoRR*, abs/1602.08127, 2016.

[8] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3974–3983, 2018.