HANNIS: A Hybrid Approximate Nearest Neighbor Indexing and Search for Large Descriptor Databases

M M Mahabubur Rahman
toufik@txstate.edu

Jelena Tešić
jtesic@txstate.edu

Data Lab @Texas State University

Computer Science

DataLab12.github.io

TEXAS STATE UNIVERSITY

MEMBER THE TEXAS STATE UNIVERSITY SYSTEM

Outline

- Introduction
- Methodology
- Dataset
- System Specification
- Experimental Analysis
- Conclusion

Introduction

- k-Nearest Neighbor (k-NN) retrieval is suitable for retrieving exact solutions in smaller datasets with lower dimensions.
- ANN search is the solution to sluggish exact k-NN search for large and high-dimensional databases.
- Approximate Nearest Neighbor (ANN) methods speed up the search by compromising some accuracy.
- In this paper, we propose an ANN search method, Hybrid Approximate Nearest Neighbor Indexing and Search (HANNIS) for large descriptor databases.
- HANNIS offers high Recall, Precision, and F1-score, fast index loading into the memory and compatible retrieval time with the state-of-the-art libraries.

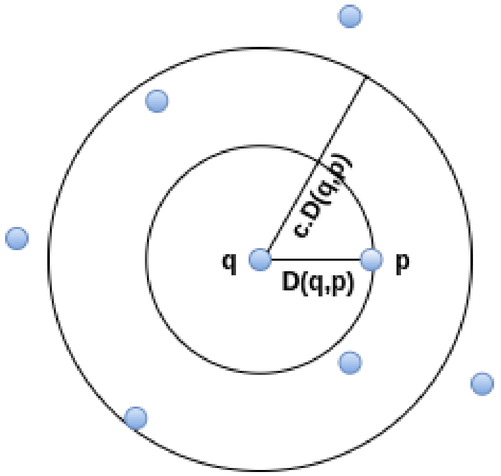

Fig. 1: ANN search

Methodology

❖Index Building:

- HANNIS first clusters all the datapoints using k-means++ and pass each cluster with the centroid information to the next phase.
- Each cluster is then arranged into a hierarchical layer of proximity graphs for building indexes with adapted Hierarchical Navigable Small World (HNSW) algorithm.
- All the indexes are then stored into the memory along with their centroid information.

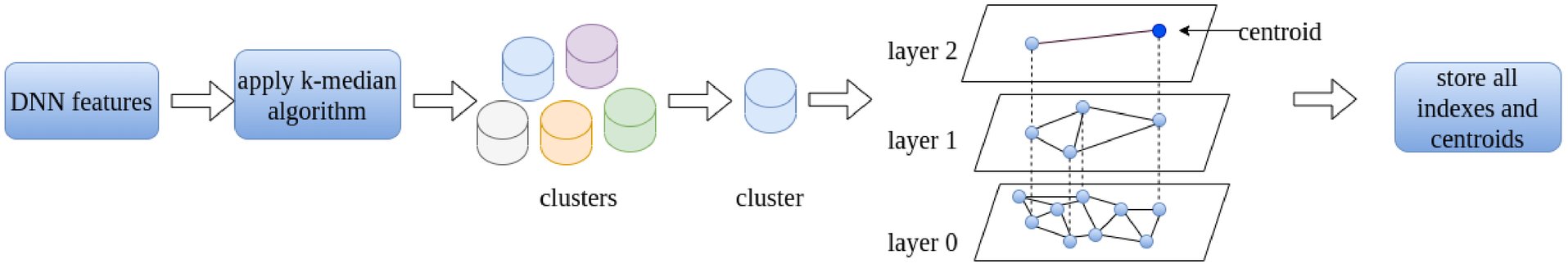

Fig. 2: HANNIS: proposed indexing approach

Methodology

❖ **Improving effectiveness of retrieval:**

- During retrieval HANNIS only loads the index that has smallest query to centroid distance.
- The search starts at the top layer and applies greedy approach to reach local optimum then moves to the lower layer.
- Searching in the lower layer starts with the previous local optimum, and this process continues until the query is reached.

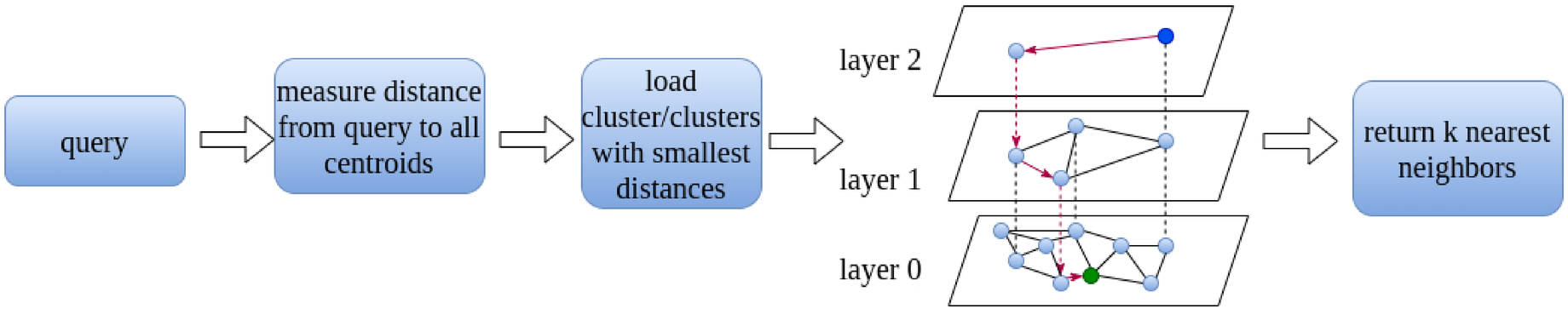

Fig. 3: Proposed HANNIS retrieval pipeline

Dataset

For all our experiments we have used two different dataset-

- ❖DOTA 2.0 data set with 2,7 million instances and 1024 dimension.
- ❖SIFT10M with 10 million instances and 128 dimension.

Dataset Name	Dimension	Number of instances
DOTA 2.0	1024	2.7 million
SIFT10M	128	10 million

Table 1: Dataset used for experimental analysis.

System Specification

System	Configuration
Operating system	Ubuntu 20.04.3
CPU	11th Gen Intel® Core™ i9-11900K @ 3.50GHz × 16
GPU	NVIDIA GeForce RTX 3070
GPU memory	8GB
RAM	128GB

Table 2: System specifications.

Experimental analysis

- We compared HANNIS performance with four other state-of-the-art library built on HNSW algorithm-
  - Lightweight approximate Nearest Neighbor library (N2)
  - Non-Metric Space Library (NMSLIB)
  - Hierarchical Navigable Small World library (HNSW lib)
  - Facebook AI Similarity Search library (FaissHNSW)
- Performance metric: Recall, Precision, F1-score, Index loading time and Retrieval time.

Experimental analysis

- HANNIS outperforms all state-of-the-art methods in terms of Recall, Precision and F1-score.
- HNSW lib gives similar performance to HANNIS in terms of Recall but HANNIS is superior in Precision and F1-score.
- N2, NMSlib and FaissHNSW has moderate Recall but all them performs poorly in terms Precision and F1-score for higher retrieval results.

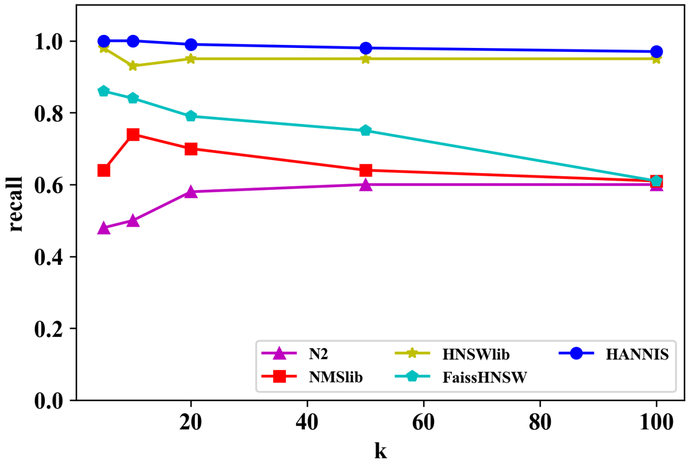

Fig. 4: Recall@k for DOTA 2.0

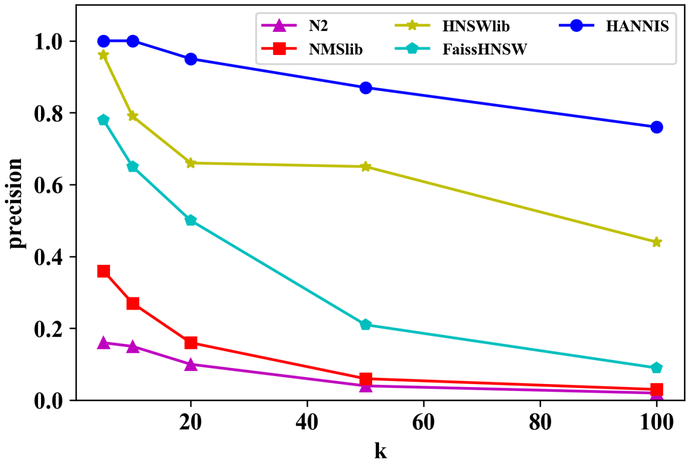

Fig. 5: Precision@k for DOTA 2.0

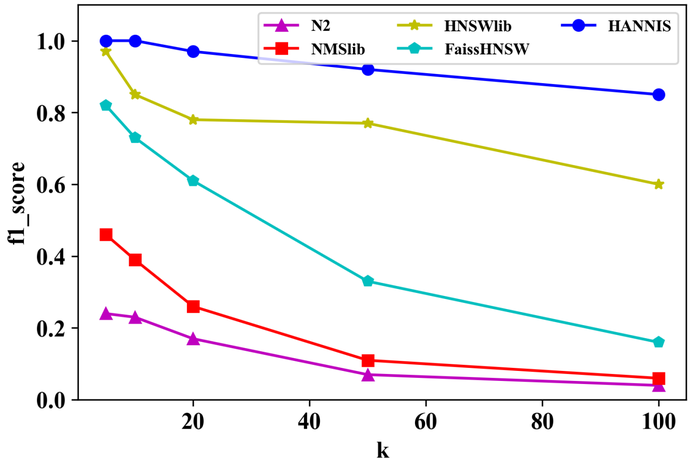

Fig. 6: F1-score@k for DOTA 2.0

Experimental analysis

- HANNIS has the fastest index loading time other N2, whereas N2 has the poorest retrieval performance.
- The best retrieval performance of HANNIS comes with the price of higher retrieval time.

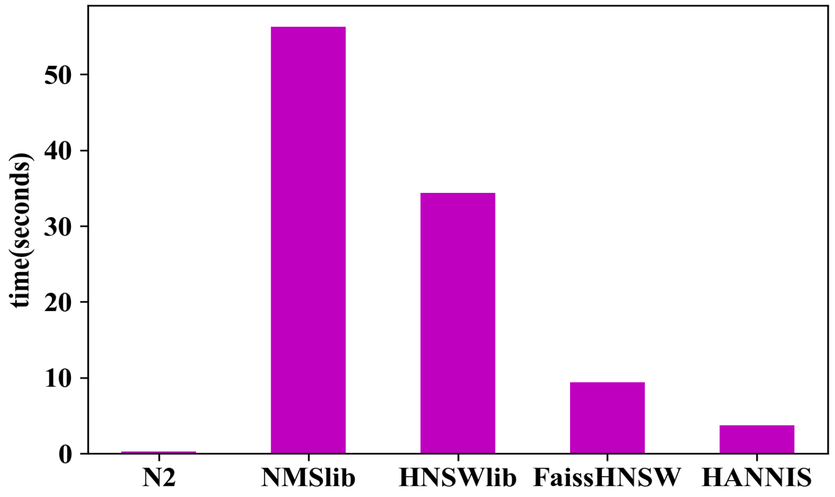

Fig. 7: Index loading times for DOTA 2.0

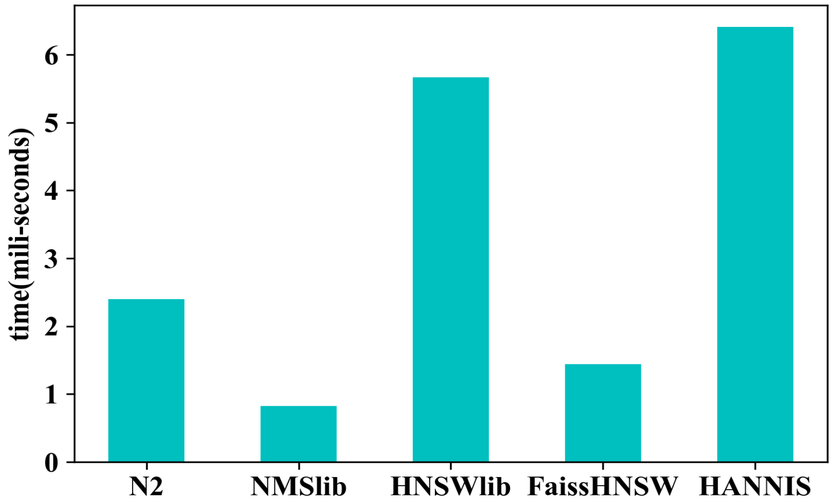

Fig. 8: Retrieval times for DOTA 2.0

Experimental analysis

- HANNIS has best Recall for k>5 over all state-of-the-art methods.
- HNSW lib Precision and F1-score drops for k>10 but performs similar to HANNIS at higher retrieval results.
- N2, NMSlib and FaissHNSW has moderate Recall and consistently low precision and F1-score for higher retrieval results.

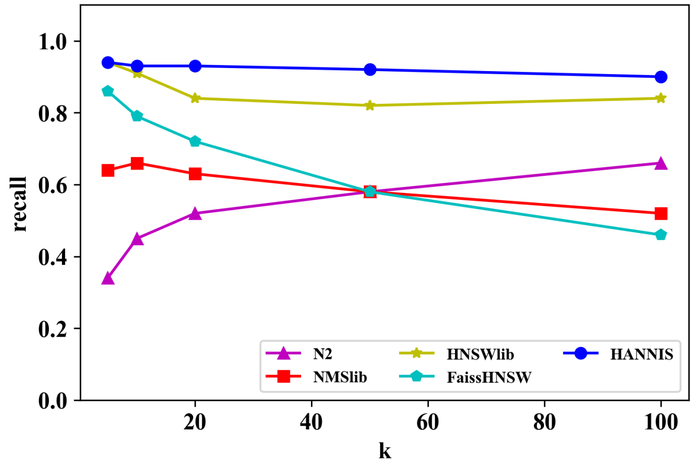

Fig. 9: Recall@k for SIFT10M

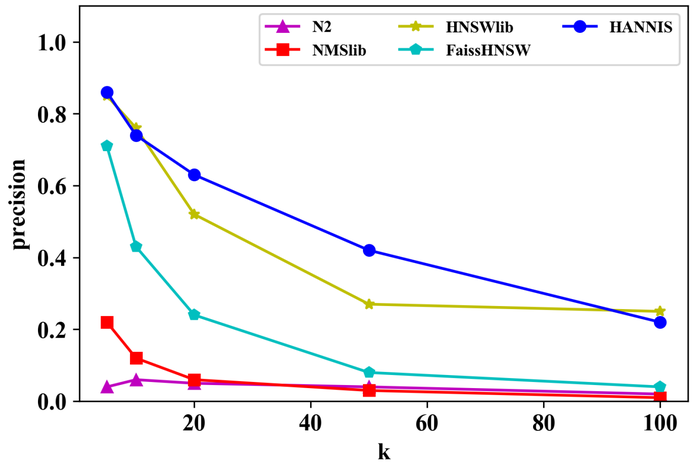

Fig. 10: Precision@k for SIFT10M

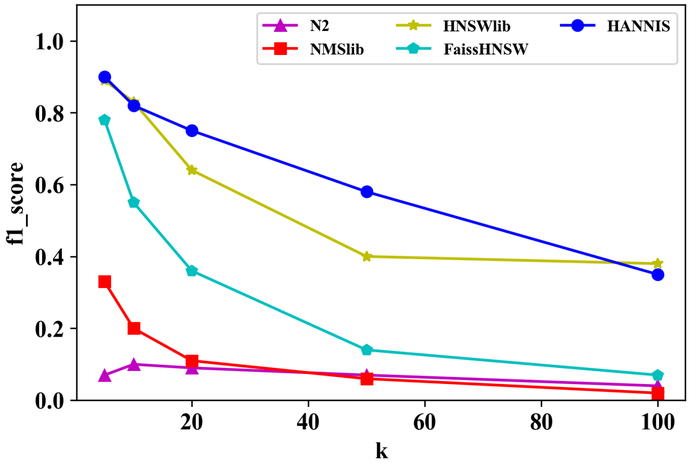

Fig. 11: F1-score@k for SIFT10M

Experimental analysis

- HANNIS has the second fastest index loading time with much better retrieval results.
- HANNIS similar retrieval time as FaissHNSW with overall best Recall, Precision and F1-score.

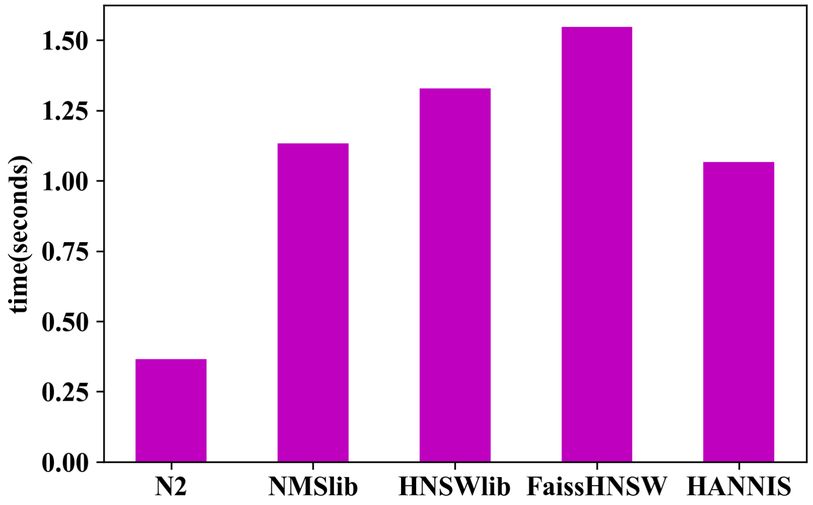

Fig. 12: Index loading times for SIFT10M

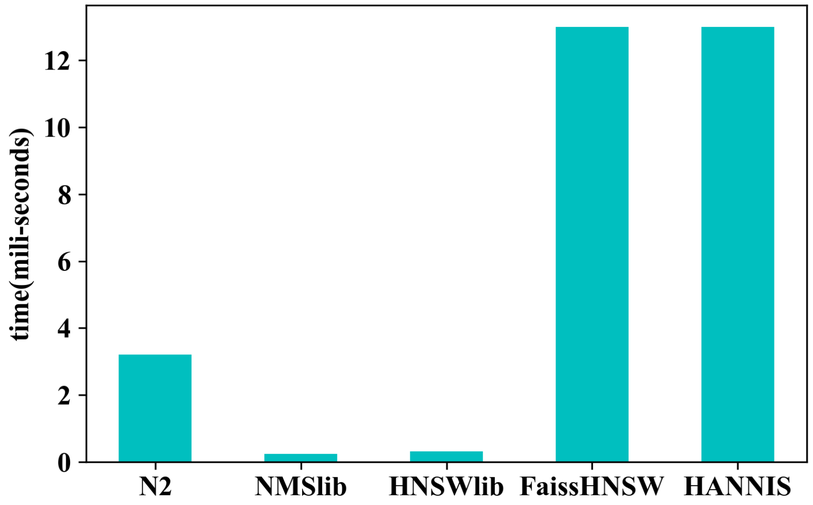

Fig. 13: Retrieval times for SIFT10M

Experimental analysis

- NMSlib, HNSW lib, FaissHNSW , and HANNIS index load time is proportional to the size of the dataset.
- N2 has the lowest index loading time, and it does not correlate to data set size.
- N2, FaissHNSW , and HANNIS methods, retrieval time corresponds to the number of instances in the dataset.

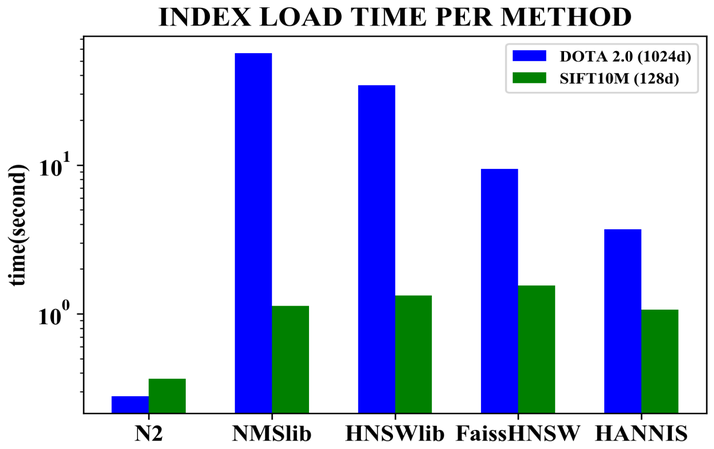

Fig. 14: Index loading for 5 approaches

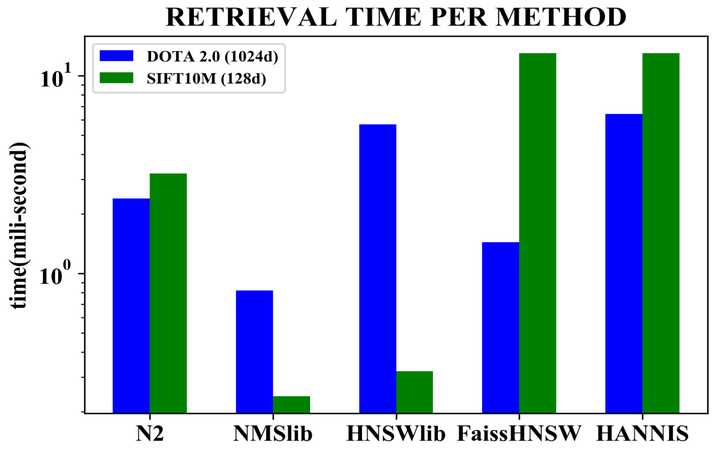

Fig. 15: Retrieval times for 5 approaches

Summary and Next Steps

- HANNIS only loads the indexes that are most similar to the query.
  - Recall exceeds that of all existing state-of-the-art libraries built on the HNSW algorithm up to 100.
  - HANNIS is up to 18 times faster than state-of-the-art libraries in terms of index loading time.
  - Retrieval times are similar to those of state-of-the-art libraries for searching in vector space.
- HANNIS outperforms all the state-of-the-art library built on HNSW algorithm, in terms of Recall, Precision, F1-score and offers fast index loading and compatible retrieval time.
- **Next Steps: Optimize HANNIS code and heuristics to handle over 2.76 billion floats**

THANK YOU! Question? More details: DataLab12.github.io