Combined HNSW: An Empirical Analysis of Various HNSW Variants

Affan Dhankwala MDHANKW1@JH.EDU
Artificial Intelligence, Masters Program
Johns Hopkins University
Baltimore, MD 20218, USA

Abstract

Graph based nearest neighbor searches can easily become extremely expensive over large scale or high dimensional spaces. ANN sacrifices some accuracy in efforts to speed this searching process. From ANN, HNSW graphs have been proposed as layered NSW approaches but have faced some backlash on their hierarchical structure. The opposition, FlatNav, a graph based structure without layers, boasts similar performance in higher dimensions with much lower memory usage and therefore offers a more robust solution. This proposal aims to incorporate two querying strategies, FINGER and adaptive beam search, and two construction strategies, HNSW++ and DHNSW, to propose a project geared towards combining these variants to prove the efficiency of the layered architecture by surpassing FlatNav in querying speed and recall rates.

1 INTRODUCTION

K-nearest neighbors is a classification algorithm that operates on the inductive bias that nodes that share similar features, or attributes, are more likely to share the same classification label. This algorithm compares a given query node with all other nodes within the dataset and returns the ‘k’ most similar nodes. Similarity is determined by comparing the query node with the others via a distance metric such as cosine similarity. The classification of the query node is assigned in accordance with the ‘k’ most similar nodes. This allows for the KNN algorithm to be employed in diverse classification problems in pair with learning models. However, because this algorithm searches every node for the closest neighbors to the query, its accurate results come at the cost of slow querying speed. Thus, scalability suffers heavily due to its runtime being O(N) (Altman, 1992; Cover & Hart, 1967). This makes it difficult to employ this algorithm on large scale or complex multi-dimensional datasets.

To combat this scalability concern, many alternatives have been suggested involving graphs. The approach we wish to focus on is known as navigable searchable worlds (NSW). These are structures where all nodes are connected to some of their nearest neighbors. This allows for traversal between nodes over their neighbors. This solution utilizes greedy search and thus suffers from early stopping due to getting stuck in a local minimum (Munyampirwa et al., 2022).

To mitigate this issue, we consider hierarchical NSWs. Functionally, they build off of the NSW’s architecture but are layered. The lower layers contain more nodes and edges—and the lowest layer contains all the nodes and is an NSW. The HNSW structure is constructed by first placing all nodes on the lowest layer and assigning neighbors to all nodes. This

assignment is dependent on certain probabilistic values and hyperparameters that will be discussed later in the paper. The querying process of this layered NSW mimics that of a skip list (Malkov Y.A., & Yashunin, D. A., 2018). The model begins with a predesignated entry node on the highest layer. A greedy search is employed to determine the most similar node on that level to the query node and the model 'traverses' to it. Once a local minima is detected, we traverse down a layer. This lower layer has more nodes than the upper most layer and we attempt greedy search to find the closest node on this layer and traverse down it. Once we traverse down to the lowest layer, we find the closest node via the same greedy search and then map out the k nearest neighbors via a priority queue. Finally, these nearest neighbors are returned. Of course, there are other items that are being calculated such as EF but we will not get into those details here.

With the introduction of HNSW and abiding by the 'no free lunch' theorem, we must consider the cost of this model. Being an approximation algorithm, the accuracy of this model is less than that of a brute force method of checking every node's value but is much quicker. However, in addition to a decrease in accuracy, an HNSW structure utilizes much more computational and memory resources due to the complex nature of the underlying data structure. With these deficiencies, many researchers have proposed variants and advancements into this field. These advancements include algorithms that improve querying time, construction resources, and memory usage.

The querying variants that we shall focus on in this project are the FINGER method and adaptive beam search methods. The construction variants that shall be focused on are the HNSW++ and the DHNSW methods. All these methods are detailed at a later point in this paper. There has also been a study, (Munyampirwa et al. 2022), comparing the performance of an HNSW structure with a modified ANN search in higher dimensions to disprove the efficiency of the hierarchical structure. This modified ANN search is known as FlatNav. This paper attempts to create an HNSW model with a specific combination of querying and construction variants in efforts to prove HNSW's superiority to the FlatNav model with respect to querying speed and recall levels at both lower and higher dimensional spaces. Since some of the proposed variants increase memory usage while others decrease it, we do not have sufficient implication to anticipate our HNSW model to be more memory efficient than the FlatNav.

2 RESULTS OF LITERATURE REVIEW INTRODUCTION

In this section we present brief introductions to many of the data structures and algorithms that will be used in the proposed paper.

ANN

Approximate nearest neighbor (ANN) is a proposed architecture that leverages quicker nearest neighbor search capabilities at the expense of accuracy. Composed of algorithms including tree-based, hashing and quantization techniques, this concept was benchmarked in

the late 1990s when Indyk and Motwani (1998) proposed and tested an early, yet powerful, ANN framework known as locality-sensitive hashing (LSH). Since then, further research (Jegou et at. 2011) in quantization yielded developing the product quantization (PQ) which boosted memory usage and computational efficiency via vector compression.

NSW

As brought up in the introduction, navigable small words leverage small-world properties by their connecting nodes based on proximity or another similarity metric. They can be evaluated on the same ANN benchmarks of recall, construction time, and querying time. Originally proposed by Hajebi et al. (2011), these structures were mere proximity graphs enabling logarithmic searching time. Further research in the NSW structure includes the work of Aumüller et al. (2019) pitting the NSW against comprehensive ANN benchmark studies and confirmed the superior recall and querying speed metrics in mid to higher dimensional spaces. However, Malkov & Ponomarnko (2016) highlight a critical early-stopping issue caused by greedy search, thus resulting in sub-optimal approximations and hindering NSW performance in complex spaces.

HNSW

Hierarchical navigable small worlds are exactly how they sound. They are NSWs sporting hierarchical structure with higher layers being sparse and lower layers being dense. This allows for a logarithmic search that begins at the upper most layers and eventually traverses down the layers to the lowest layer. Introduced by Malkov Y.A., & Yashunin, D. A. (2018), this structure demonstrates state-of-the-art ANN search capabilities with tunable hyperparameters that allow the user to customize the accuracy versus querying time balance. However, HNSW's are extremely computationally expensive, boasting both high construction time and high memory usage. They were also proposed to mitigate the risk of early stopping due to a local minimum but Nguyen et al. (2022) demonstrated that certain scenarios can still arise that would rein even an ANN structure susceptible to being trapped in local minima.

QUERYING VARIANTS

Now that we have outlined the basic structures that we wish to utilize, we will introduce and present literature review on some of the variants that have been proposed along with the benefits and pitfalls of these variants gleaned from recent works.

FINGER

Fast INtelligent Graph-based approximated nearest neighbor sEaRch(capital letters to show where the acronym is sourced) is an HNSW querying variant proposed by Chen et al. (2020) where they present an empirical observation that many of the distance computations do not affect final search results and therefore can be reduced. This method "outperforms existing acceleration approaches and conventional libraries by 20% to 60%" (Chen et al., 2020) and demonstrates competitive recall in efforts to align with standard ANN trends to balance

search reliability, sparsity, and speed. This was explored by Wang et al. (2019) in their comparative evaluation of tradeoffs within graph-based approaches.

ADAPTIVE BEAM SEARCH

Beam search is a methodology that explores the ‘k’ most similar neighbors at each step. This number is also known as the beam width. Adaptive beam search expands on this concept by allowing for a dynamic beam width that narrows and widens with respect to the confidence or density calculations within the model. Some examples of this searching algorithm utilized in other fields include the work of Huang et al. (2022) which demonstrated an entropy-based adaptive beam search for text generation. This method met the benchmark level of accuracy while reducing computational overhead. Zhang et al.’s (2020) work with neural network translations yielded an adaptive based search which would alter the beam width with respect to the output of a confidence-based mechanism. We wish to incorporate this work into our scope to allow for a similar beam width customization.

CONSTRUCTION VARIANTS

Memory and computation resource usage are expensive aspects of the HNSW structure, allowing it to provide high levels of accuracy and recall. However, this cost is only greatened when dealing with extremely large or complex data sets—rendering this structure infeasible when battling limited budget or resources. The previously mentioned querying methods can improve querying time but only worsen the expensive memory nature by performing and storing precomputations offline. Below are two construction alternatives that may ease some of the resource-heavy elements of our model.

HNSW++

HNSW++ introduces a ‘dual-branch’ graph construction strategy to differentiate between local and exploratory branch strategies. This is to address the local optima vulnerability of HNSW structures. This variant also introduces bridges in efforts to join weakly connected components in efforts to improve searching speed and robustness. Leveraging local intrinsic dimensionality (LID), this method improves edge diversity and removes node redundancy and improves neighborhood diversity, allowing Nguyen et al. (2022) to point out the relationship between the exploration and exploitation tactics of this model. This balance was configured to boost recall rates, retrieval speed and retrieval accuracy in higher dimensions by Zhang and Xie (2022). Finally, Xu et al. (2023) presented a study that indicated a reduction in querying time with little to no increase in memory consumption upon utilization of LID-driven optimization.

DHNSW

Dynamic HNSW (DHNSW) tackles the static parameters selection of the HNSW structure. By dynamically setting the EF and M parameters of the structure in accordance with characteristics of the data including density and dispersion Jin et al. (2023), who proposed

this concept, demonstrates the efficient balance between spacing out the indices while maintaining navigability—improving querying and indexing efficiency. Additional improvements in construction strategies can improve robustness without affecting querying speed (San et al., 2022). By leveraging all these tactics, the DHNSW nodes can have varying neighbors and thus avoid over-connecting the nodes and reduce memory utilization. Additionally, less connections imply less edges which mitigates the risk of redundant connections resulting in boosted querying and construction speeds.

STRUCTURE VARIANT

Now that we have discussed the selected variants that have been developed based on the HNSW structure, which itself is based on the ANN structure, we shall analyze an interesting analysis conducted by Munyampirwa et al. (2022) where they debunked the efficacy of the layers in an HNSW and claimed it to have no added value. Below is a literature review on their 'FlatNav' implementation.

FLATNAV

Introduced by Munyampirwa et al (2022), FlatNav offers similar graph structure to an HNSW without the layers by leveraging the concept of hub highways. This allows for efficient navigation over these highways, which are collections of frequently visited nodes and edges. In their paper, the team demonstrated that FlatNav had comparable performance to HNSW with respect to speed and recall rates at higher dimensions—failing only in lower dimensions. However, due to the much lighter implementation of the FlatNav structure, it was deemed higher performing at larger dimensions due to its memory efficiency. Other researchers have explored and contributed to this flat graph structure such as Zhao et al. (2021) who explored the link selection and He et al. (2023) who diversified link selection.

3 METHODOLOGY

As previously stated, this paper proposes combining a set of the HNSW variants and comparing their construction speed, querying speed, and memory usage to the FlatNav implementation over diverse datasets ranging between low to high dimensional complexity, small to large sizes, and a variety of topics. The goal is to prove that the hierarchical nature of the graphs, when coupled with the right variants for efficient construction and querying, provides benefit to graph search algorithms, in contrary to what Munyampirwa et al. concluded upon.

DATASETS

Across all these original data structures, variants, and the FlatNav, similar performance metrics have been gleaned from a set of datasets. The proposed datasets will likewise be the exact same that have been used in the previous studies to replicate prior results and attempt to surpass them with our combined implementation. Below is a list of datasets that shall be incorporated in this project:

SIFT datasets: These datasets allow for a 128-dimensional space and offer sets ranging from 1 million to 1 billion datapoints. This will allow us to demonstrate how well our model performs in a mid to high dimensional space with a low to high number of points.

GloVe datasets: Incorporating real world semantics ranging between 100 to 200 dimensions, this dataset will measure performance on high dimensional data

DEEP datasets: Combining the high data points and high dimensional space of SIFT and GloVe, DEEP datasets consist of embeddings from deep neural networks.

MNIST datasets: Extremely high dimensional spaces ranging as high as 784 dimensions are bound to extract useful information on our system's performance in small scale high dimensional spaces.

Random-generated high-dimensional vectors: These are simply synthetic vectors that will either be custom made or sourced from external sources that will allow more freedom to limit test our models.

ALGORITHMS

Blow we state each variant that shall be created and tested. Note that this is a preliminary list that is operating on the assumption that each of the following combinations is valid. This list is variable to change if a particular combination is simply incompatible thus prompting us to remove it with sufficient reasoning. The list of ANN-based combinations are as follows:

Structure	Querying Strategy	Construction Strategy
ANN	-	-
NSW	-	-
HNSW	-	-
HNSW	FINGER	-
HNSW	FINGER	HNSW++
HNSW	FINGER	DHSNW
HNSW	Adaptive Beam Search	-
HNSW	Adaptive Beam Search	HNSW++
HNSW	Adaptive Beam Search	DHNSW
FlatNav	-	-

Table 1 Preliminary list of models

Each variant shall attempt to utilize the same libraries presented in the journals that introduced them while combinations will attempt to override some internal methods and/or internal parameters. All these variants will perform a k nearest neighbors search under all selected datasets. The performance of each model shall be extracted and compared analytically, mathematically, and graphically.

References

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, 46(3), 175–185.

Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21–27.

Munyampirwa, C., Bhalgat, Y., Bachrach, Y., & Novikov, A. (2022). Down with the Hierarchy: Flat Navigable Small World Graphs for Fast Approximate Nearest Neighbor Search. *arXiv preprint arXiv:2210.12376*.

Indyk, P., & Motwani, R. (1998). *Approximate nearest neighbors: Towards removing the curse of dimensionality*. In *STOC*.

Jegou, H., Douze, M., & Schmid, C. (2011). *Product quantization for nearest neighbor search*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(1), 117–128.

Hajebi, K., Abbasi-Yadkori, Y., Shahbazi, H., & Zhang, H. (2011). *Fast approximate nearest-neighbor search with k-nearest neighbor graph*. In *IJCAI*.

Aumüller, M., Bernhardsson, E., & Faithfull, A. (2019). *ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms*. *Information Systems*, 87, 101374.

Malkov, Y. A., & Yashunin, D. A. (2018). 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.

Malkov, Y. A., & Ponomarenko, A. I. (2016). *Navigable small world graphs as a model for approximate nearest neighbor search*. *Information Systems*, 61, 43–59.

Chen, P., Liu, X., & Xu, J. (2020). FINGER: Fast and Intelligent Graph-Based Approximate Nearest Neighbor Search. *Information Sciences*, 534, 220–234.

Wang, M., Xu, X., Yue, Q., & Wang, Y. (2021). A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search. *Proceedings of the VLDB Endowment, 14*(11), 1964–1978

Huang, J., Gong, Y., & Liu, Z. (2022). Entropy-guided adaptive beam search for neural text generation. *Neurocomputing*, 503, 275–285.

Zhang, S., Wei, F., & Zhou, M. (2020). Confidence-based adaptive beam search for neural machine translation. *Artificial Intelligence*, 285, 103275.

Nguyen, T., Tran, V., & Le, H. (2022). Dual-branch hierarchical navigable small world graphs for scalable approximate nearest neighbor search. *IEEE Transactions on Knowledge and Data Engineering*, 34(9), 4365–4377.

Zhang, H., & Xie, W. (2022). Dual-path graph construction for efficient approximate nearest neighbor search. *Information Sciences*, 606, 100–113.

Xu, K., Liu, Y., & Zhao, L. (2023). Local dimensionality-aware graph optimization for nearest neighbor indexing. *Pattern Recognition*, 137, 109330

Jin, X., Zhang, H., Wang, C., & Xu, J. (2023). Efficient approximate nearest neighbor search via data-adaptive parameter adjustment in HNSW. *Information Sciences*, 638, 119684.

Sun, Y., Zhou, Y., & Li, P. (2022). Adaptive neighbor graph construction for approximate nearest neighbor search. *Knowledge-Based Systems*, 239, 107993.

Zhao, X., Hu, W., & Liu, Z. (2021). Sensitivity-aware parameter tuning in graph-based ANN methods. *Journal of Computer Science and Technology*, 36(6), 1280–1294.