Efficient Approximate Nearest Neighbor Search via Data-Adaptive Parameter Adjustment in Hierarchical Navigable Small Graphs

연세대학교 컴퓨터과학과 Huijun Jin

2024년 12월

과제명: IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

Efficient Approximate Nearest Neighbor Search via Data-Adaptive Parameter Adjustment in Hierarchical Navigable Small Graphs

Huijun Jin, Jieun Lee, Shengmin Piao, Sangmin Seo, Sein Kwon, Sanghyun Park

Data Engineering Lab, Yonsei University, South Korea

Problem Definition

- **Introduction to the Problem**
  - Approximate Nearest Neighbor (ANN) search is crucial for various high-dimensional data applications, such as image retrieval, recommendation systems, and natural language processing

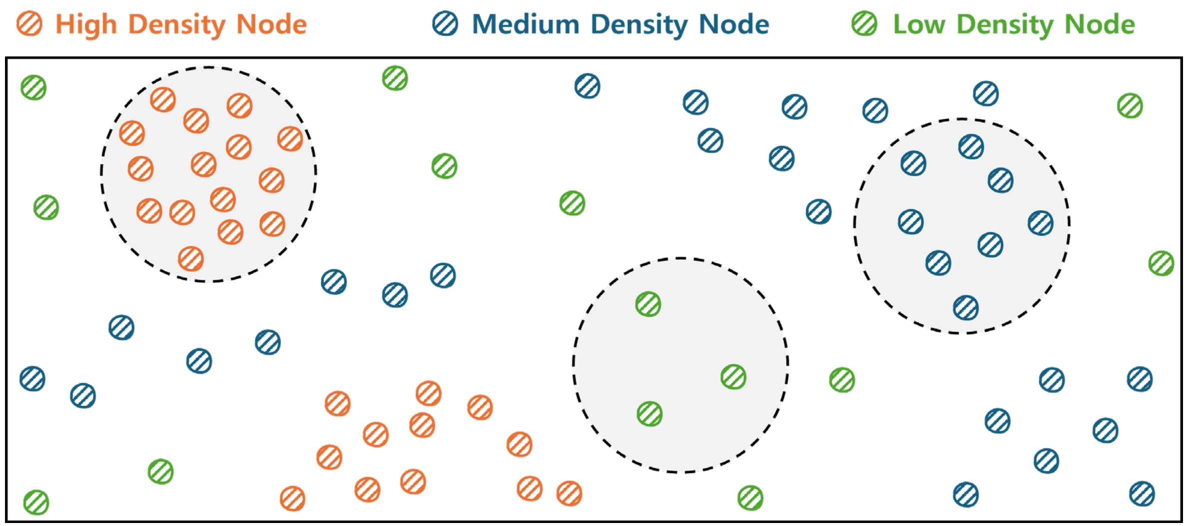


Fig. 1. Example of data density distribution in 2D space.

- **Limitation of Existing Solutions**
  - Hierarchical Navigable Small World (HNSW) is a popular ANN search algorithm due to its efficiency and scalability. However, its static parameters M (connections per node) and ef (candidate list size) cannot adapt to local data density variations
    → **High-density regions**: Insufficient connections reduce recall
    → **Low-density regions**: Excessive connections waste memory and computational resources

- **Objective**
  - To overcome these inefficiencies, we propose **Dynamic HNSW (DHNSW)**, which introduces **data-adaptive parameter adjustment** to optimize performance across varying data densities

Contribution

- **Dynamic Parameter Optimization**
  - Introduces a novel method to adaptively adjust HNSW parameters based on local data characteristics. To the best of our knowledge, this approach is the first to provide localized, adaptive control at the node level, enabling more efficient handling of varying data densities

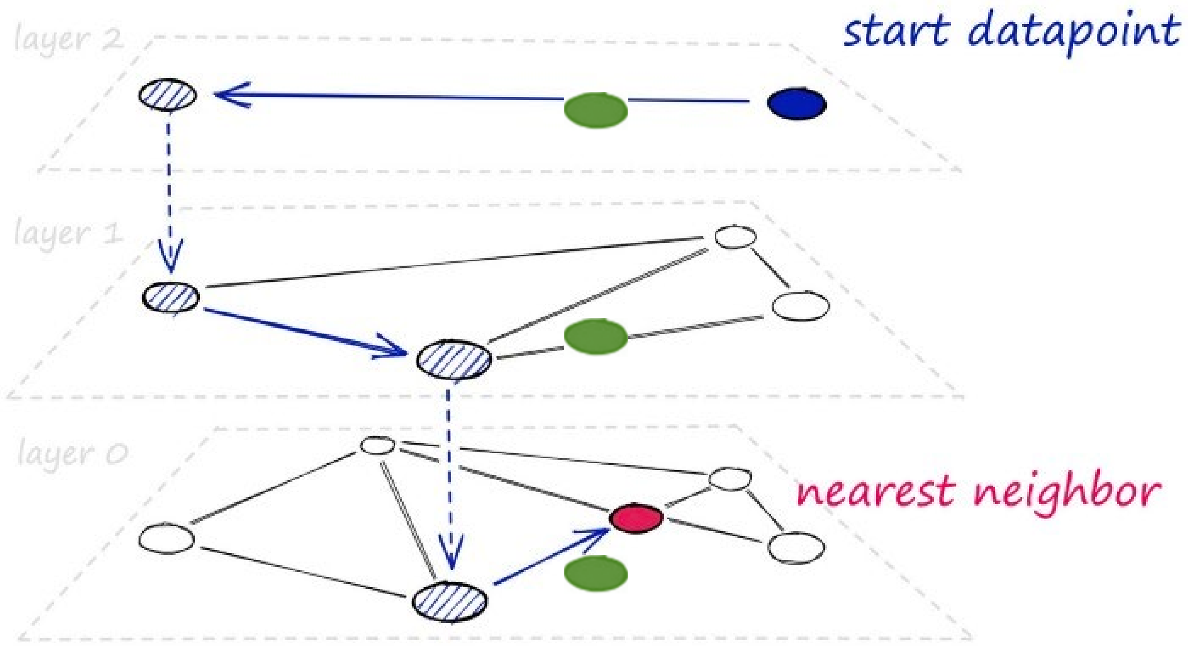
- **Data-Adaptive Algorithmic Framework**
  - Develops an algorithmic approach that dynamically fine-tunes key parameters as the graph is constructed, ensuring improved efficiency across diverse data distributions

- **Comprehensive Evaluation and Analysis**
  - Demonstrates DHNSW's effectiveness through extensive experiments, showing reductions in build time and memory usage while maintaining competitive recall, confirming its adaptability and scalability for large-scale, high-dimensional ANN search

How Vanilla HNSW Works

- **Overview of HNSW**
  - ➢ Constructs a **hierarchical, multi-layer** graph for ANN search
  - ➢ Each layer refines data representation
    → **Top layer**: Sparse and global connections
    → **Lower layers**: Dense, local connections
  - ➢ Traversal: Start from the top layer and refine the search as you descend

- **Static Parameters in Vanilla HNSW (fixed globally)**
  - ➢ $\boldsymbol{M}$: Maximum connections per node
  - ➢ $\boldsymbol{ef}$: Candidate list size during search



https://www.pinecone.io/learn/series/faiss/vector-indexes/

DHNSW: Overview and Key Concepts

- **What is RP-KNN?**
  - **Random Projection - K Nearest Neighbors (RP-KNN)**
    → A technique used to estimate local data density efficiently, even in high-dimensional spaces
    → Projects data into a lower-dimensional space using random projections
    → Computes density as the inverse of the average distance to the k-nearest neighbors in the reduced space

- **Why Use ef_{ref}?**
  - ef_{ref} serves as a global reference point for search depth
  - Provides a baseline to scale ef_{low} and ef_{high} dynamically to reduce the effect of the curse of dimensionality
  - **α**: Adjusts the magnitude of ef_{ref} scaling

$$ef_{ref} = \left\lfloor ef_{init} + \left(\frac{dim(D)}{\alpha}\right)^2 \right\rfloor \qquad (1)$$

Algorithm 1: DHNSW Graph Construction

Input: Initial parameters M_{init}, ef_{init}, Dataset D, # of data n, Data point x
Output: Updated HNSW with all elements inserted

Calculate local data density, $\rho \leftarrow$ RP-KNN (D)
Set a reference point, $ef_{ref} \leftarrow ef_{init} + \left(\frac{dim(D)}{\alpha}\right)^2$
Calculate boundaries $M_{low}, M_{high}, ef_{low}, ef_{high}$ based on M_{init} and ef_{ref}
for $q = 1$ to n **do**
    $M_q \leftarrow M_{low} + \left(\frac{\rho_q - \rho_{min}}{\rho_{max} - \rho_{min}}\right) \times (M_{high} - M_{low})$
    $ef_q \leftarrow ef_{low} + \left(\frac{\rho_q - \rho_{min}}{\rho_{max} - \rho_{min}}\right) \times (ef_{high} - ef_{low})$
    Insert x_q into HNSW based on M_q and ef_q
end for
return Updated HNSW

DHNSW: Dynamic Parameter Adjustment

- **Setting Bounds for M and ef**
  - **λ**: Controls sensitivity to density variations

$$M_{low} = max\left(2, \left\lfloor M_{init} - M_{init} \times \left(\frac{\rho_\sigma}{\rho_\mu}\right) \times \lambda \right\rfloor\right) \tag{2}$$

$$M_{high} = \left\lfloor M_{init} + M_{init} \times \left(\frac{\rho_\sigma}{\rho_\mu}\right) \times \lambda \right\rfloor \tag{3}$$

$$ef_{low} = max\left(10, \left\lfloor ef_{ref} - ef_{ref} \times \left(\frac{\rho_\sigma}{\rho_\mu}\right) \times \lambda \right\rfloor\right) \tag{4}$$

$$ef_{high} = \left\lfloor ef_{ref} + ef_{ref} \times \left(\frac{\rho_\sigma}{\rho_\mu}\right) \times \lambda \right\rfloor \tag{5}$$

Algorithm 1: DHNSW Graph Construction

Input: Initial parameters M_{init}, ef_{init}, Dataset D, # of data n, Data point x

Output: Updated HNSW with all elements inserted

```
Calculate local data density, ρ ← RP-KNN (D)
Set a reference point, ef_ref ← ef_init + (dim(D)/α)^2
Calculate boundaries M_low, M_high, ef_low, ef_high based on M_init and ef_ref
for q = 1 to n do
    M_q ← M_low + ((ρ_q − ρ_min)/(ρ_max − ρ_min)) × (M_high − M_low)
    ef_q ← ef_low + ((ρ_q − ρ_min)/(ρ_max − ρ_min)) × (ef_high − ef_low)
    Insert x_q into HNSW based on M_q and ef_q
end for
return Updated HNSW
```

DHNSW: Dynamic Parameter Adjustment

- **Dynamic Scaling for Each Node**

$$M_q = M_{low} + \left\lfloor \left(\frac{\rho_q - \rho_{min}}{\rho_{max} - \rho_{min}}\right) \times \left(M_{high} - M_{low}\right) \right\rfloor \quad (6)$$

$$ef_q = ef_{low} + \left\lfloor \left(\frac{\rho_q - \rho_{min}}{\rho_{max} - \rho_{min}}\right) \times \left(ef_{high} - ef_{low}\right) \right\rfloor \quad (7)$$

Algorithm 1: DHNSW Graph Construction

Input: Initial parameters M_{init}, ef_{init}, Dataset D, # of data n, Data point x

Output: Updated HNSW with all elements inserted

```
Calculate local data density, ρ ← RP-KNN (D)
Set a reference point, ef_ref ← ef_init + (dim(D)/α)^2
Calculate boundaries M_low, M_high, ef_low, ef_high based on M_init and ef_ref
for q = 1 to n do
    M_q ← M_low + ((ρ_q − ρ_min)/(ρ_max − ρ_min)) × (M_high − M_low)
    ef_q ← ef_low + ((ρ_q − ρ_min)/(ρ_max − ρ_min)) × (ef_high − ef_low)
    Insert x_q into HNSW based on M_q and ef_q
end for
return Updated HNSW
```

Experimental Results

- **Datasets and Metrics**
  - Datasets: MNIST, GloVe100K, SIFT1M, GIST1M
  - Metrics: Build Time, Memory Usage, Recall
- **Baseline Performance Comparison Across Datasets**
  - Build Time: Reduced by up to **33.11%**
  - Memory Usage: Reduced by up to **32.44%**
  - Recall: Maintaining comparable, better in GIST1M (**79.26%**)

TABLE I. DATASET DETAIL

Dataset	Dimension	Size	# of Samples
MNIST	784	52 MB	60,000
GloVe100K	300	990 MB	100,000
SIFT1M	128	550 MB	1,000,000
GIST1M	960	5.37 GB	1,000,000

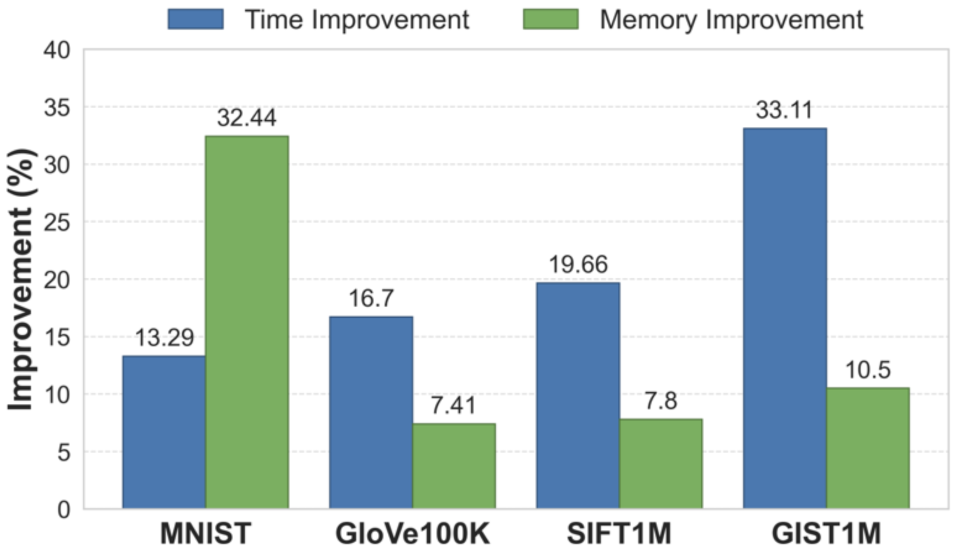


Fig. 2. Build time and memory usage improvements across datasets.

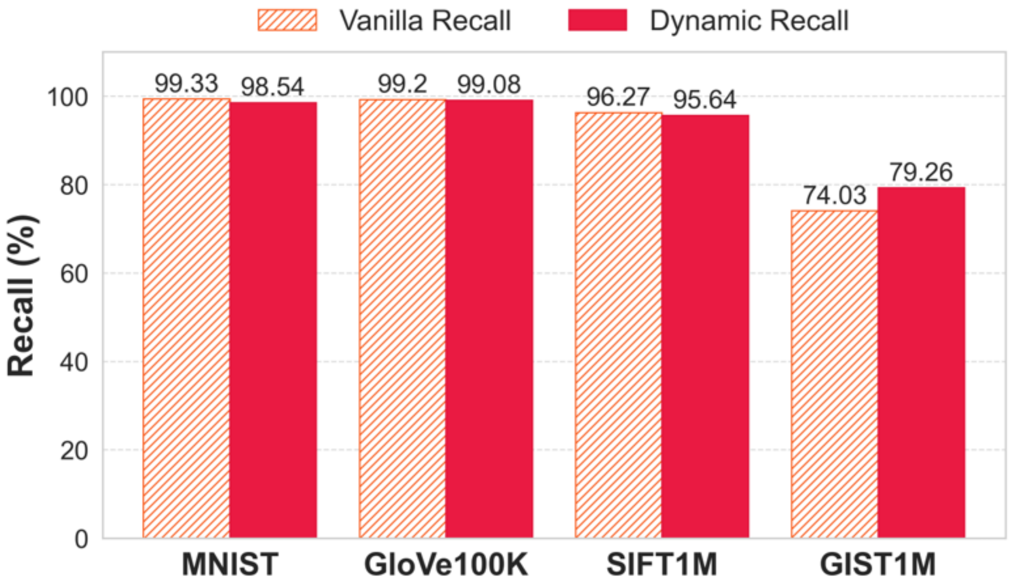
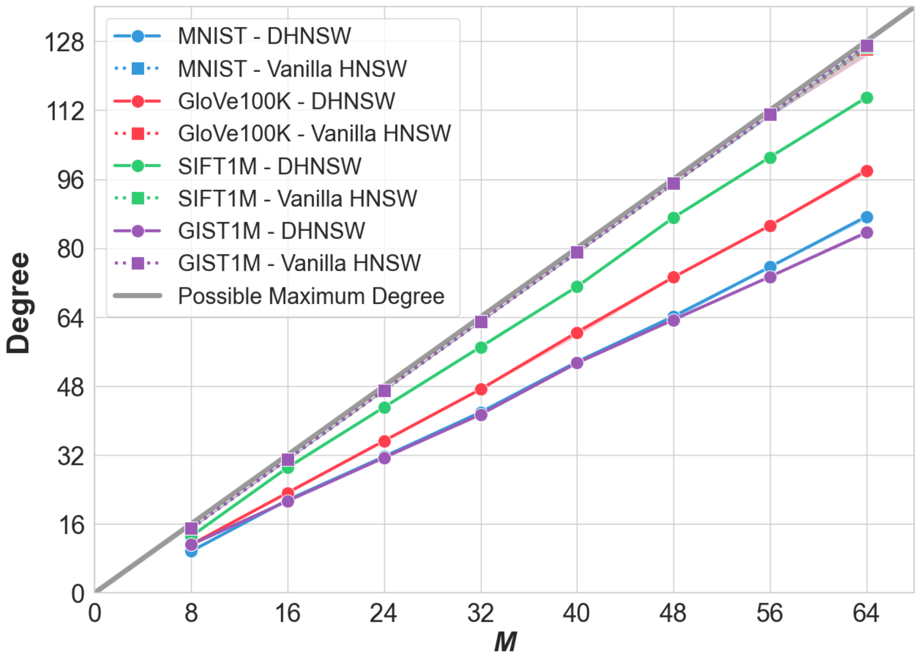
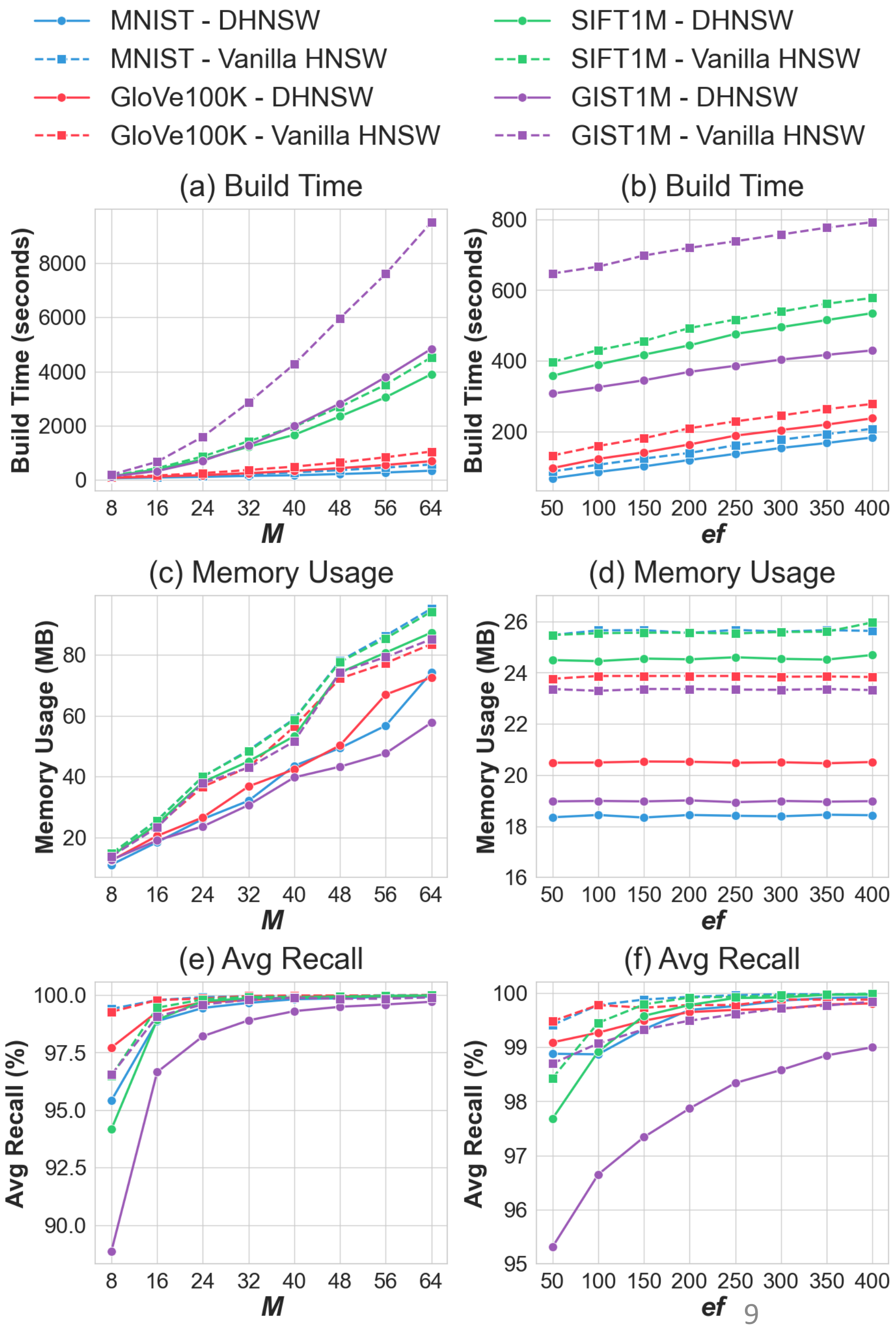


Fig. 3. Recall comparison between vanilla HNSW and DHNSW.

Experimental Results

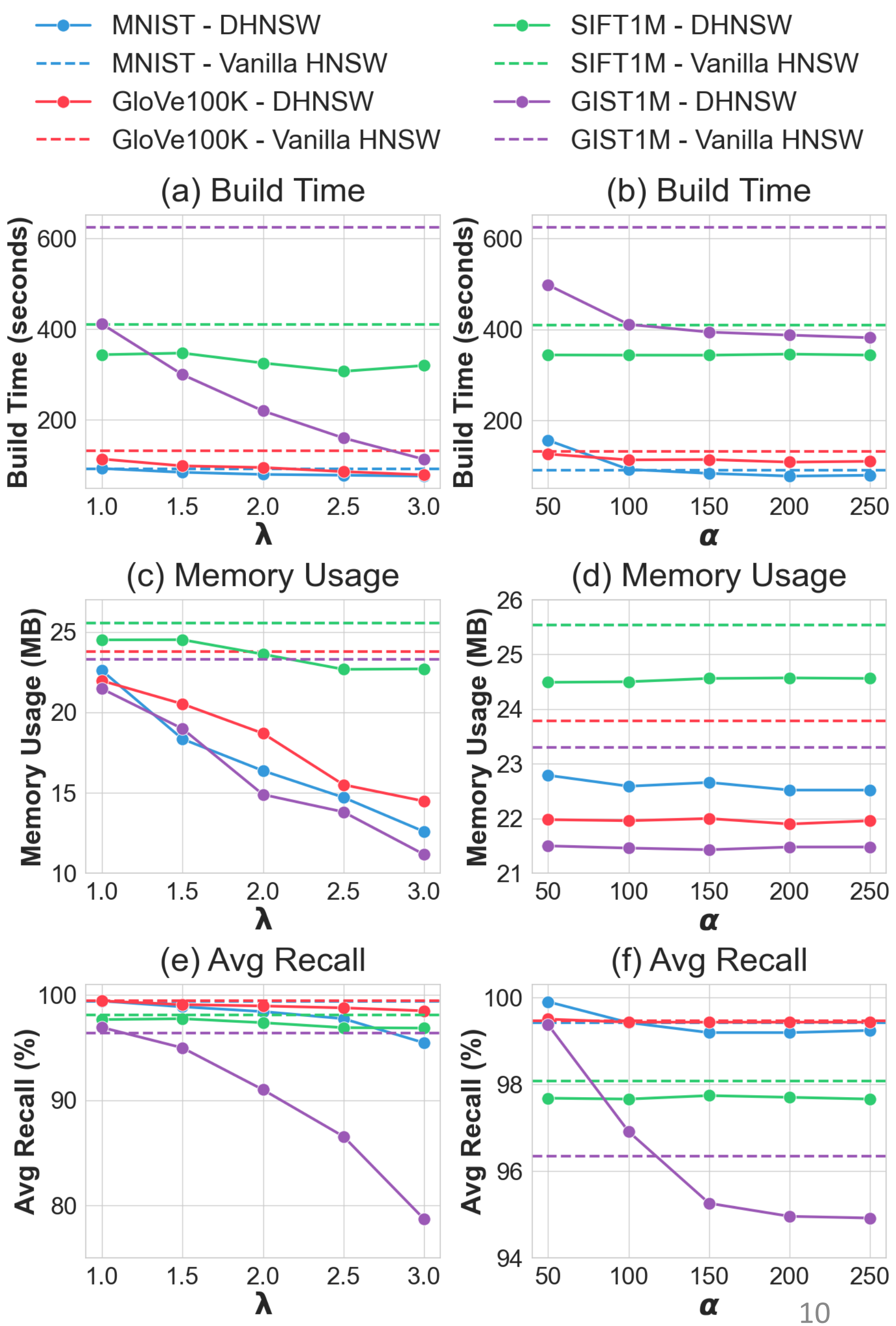
- **Robustness Analysis Across Different Initial Parameter Settings**
  - ➢ Build Time: Reduced by up to **29.04%**
  - ➢ Memory Usage: Reduced by up to **20.70%**
  - ➢ Recall: Competitive





Experimental Results

- **Effect of Different Hyperparameter Settings**
  - ➢ **λ**: consistent reductions in build time and memory usage some decrease in recall for GIST1M
  - ➢ **α**: relatively stable impact on build time and memory usage lower values of α yield higher recall, especially in GIST1M



Summary and Future Directions

- **Proposed Method**
  - Introduced DHNSW, an enhanced HNSW algorithm with **dynamic parameter tuning** for M and ef at the node level
  - First approach to incorporate **node-specific adjustments** in the HNSW framework

- **Key Results**
  - Improved **graph build time** and **memory usage**
  - Maintained **competitive recall rates** across benchmark datasets

- **Future Work**
  - Develop strategies for **automatic tuning** of hyperparameters λ and α
  - Explore **adaptive mechanisms** and advanced techniques for density estimation or learning-based methods
  - Aim for improved adaptability and robustness in **diverse real-world applications**

Q & A

Jinhuijun@yonsei.ac.kr