A Cluster-Based Approach to kNN Join over Batch-Dynamic High-Dimensional Data

Advanced Data Mining and Applications 2024

Never Stand Still | Faculty of Engineering | Computer Science and Engineering

The University of New South Wales, Sydney

Nimish Ukey, Guangjian Zhang, Zhengyi Yang,
Xiaoyang Wang, Binghao Li, Serkan Saydam, and Wenjie Zhang

Introduction

- **What is kNN-join?**
  - Finds the *k* nearest neighbors for each point in the **query dataset R** from an **answer dataset S**.

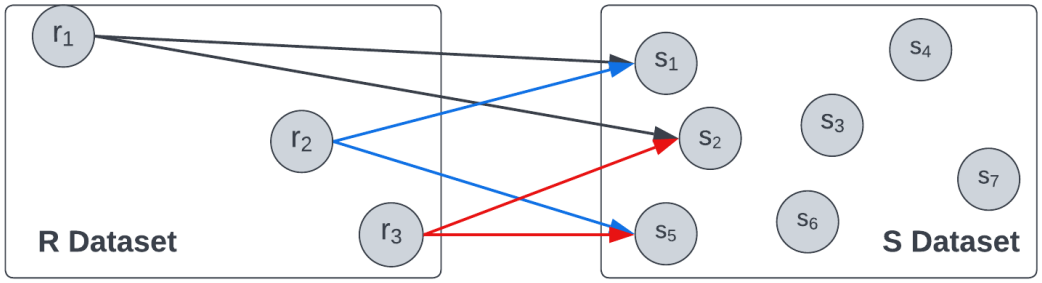


Fig. 1. An example of kNN Join with k = 2.

- **Applications**
  - kNN classification, k-means clustering, outlier detection, similarity search, etc.

Introduction

- **What is Dynamic kNN join?**
  - For any item updates, the affected user list must be updated efficiently

- **Applications of Dynamic kNN Join**
  - Recommendation system
  - Feature extraction
  - Video on-demand
  - Social network services, etc

Introduction

- **What is Batch-Dynamic kNN join?**
  - Where updates occur in batches rather than individually
  - It means maintaining the results of kNNJ (R, S) for every batch update in R or S.

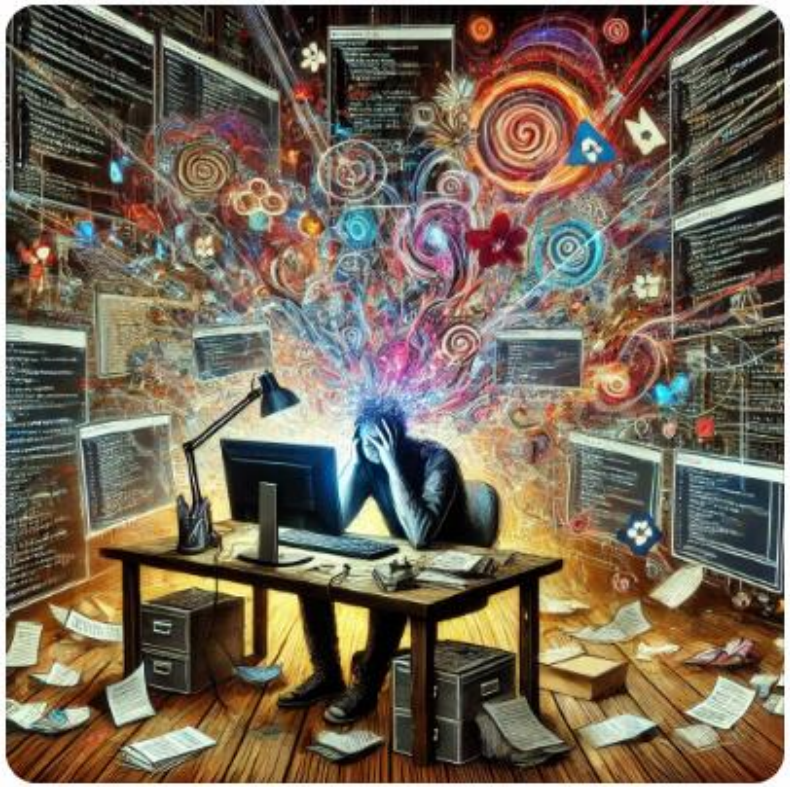
- **Example:**
  - R = $[r_1, \ldots, r_i, \ldots, r_N]$, S = $[s_1, \ldots, s_i, \ldots, s_M]$, a series of new data points $[s_{M+1}, \ldots, s_{M+I}]$ are inserted into S, update kNNJ(R, S) every batch of inserted data points instead of every inserted data point.

Importance of Batch Dynamic kNNJ

Batch kNN join can offer advantages in some specific scenarios:

1. Prevent information overload

Too frequent recommendations from e-commerce platforms can cause users to experience information overload, making it difficult for them to locate their preferred items.



Importance of Batch Dynamic kNNJ

Batch kNN join can offer advantages in some specific scenarios:

2. **Saving of computation resources**

Implementing real-time recommendation algorithms requires substantial computational resources, which may not be feasible for organizations with limited resources. Batch processing offers a more resource-efficient solution.

Research Problem

- **Key Challenges:**
  - Reducing redundant computation in batch updates.
  - Efficient processing with scalable index structures.

- **Research Questions:**
  - *RQ1:* How to identify shared computation in batch updates with minimal overhead?
  - *RQ2:* How to process shared computation efficiently?

Structure of HDR Tree

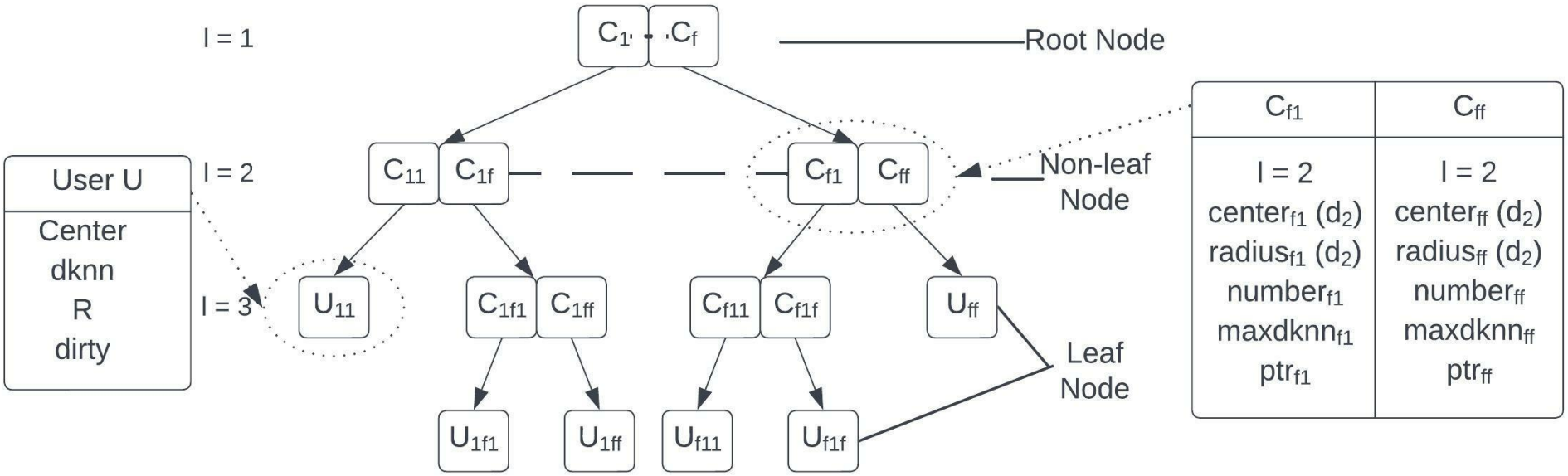


Fig. 2. HDR-tree structure

Proposed Methodology

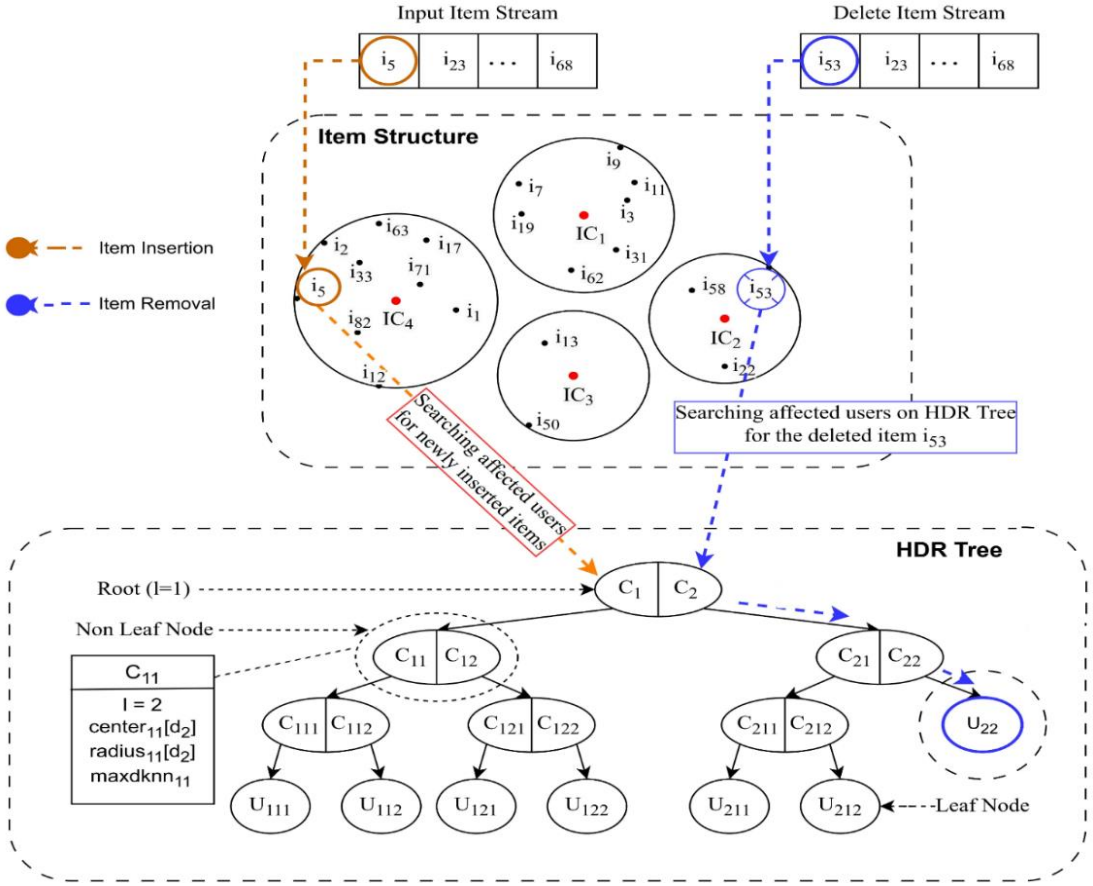
- Proposed a **Cluster-Based Batch Update** technique:
  - Groups similar updates into clusters
  - Reduces computation via shared processing.

- Introduced **Cluster-Based Pruning** using HDR-Tree:
  - Prunes irrelevant data efficiently, i.e. it avoids unnecessary computations.

- Demonstrated superior performance:
  - 19x to 55x faster than state-of-the-art methods.

What techniques do we use?

if $dist_{PCA}(ct(C_i), ct(C)) - r_{PCA}(C_i) - r_{PCA'}(C) < C.maxdknn$ **then**
`RkNN_Search_Update_by_Cluster`$(C, S', C_i.child, M^{O}_{HDR}, Mode)$;



	$C_{1,1}$	…	C_{1,N_1}	…	$C_{i,1}$	…	C_{i,N_i}	…	$C_{L-1,1}$	…	$C_{L-1,N_{L-1}}$
C_1	$d_1^{1,1}$	…	d_1^{1,N_1}	…	$d_1^{i,1}$	…	d_1^{i,N_i}	…	$d_1^{L-1,1}$	…	$d_1^{L-1,N_{L-1}}$
…	…	…	…	…	…	…	…	…	…	…	…
C_m	$d_m^{1,1}$	…	d_m^{1,N_1}	…	$d_m^{i,1}$	…	d_m^{i,N_i}	…	$d_m^{L-1,1}$	…	$d_m^{L-1,N_{L-1}}$

Table 1. Matrix M^{O}_{HDR}

Fig. 4. Process of cluster-based efficient updating of *kNNJ*(*U, I*).

Experimental Setup

- **Language:** C++
- **RAM:** 16GB
- **Processor:** Intel Core i7-10700 2.90 GHz
- **OS:** Windows 10 Pro
- **Real-world Dataset:** NUS-WIDE Image Dataset, Cifar, Trevi, etc
- **Dimensions:** 128D to 4096D.

Experimental Results

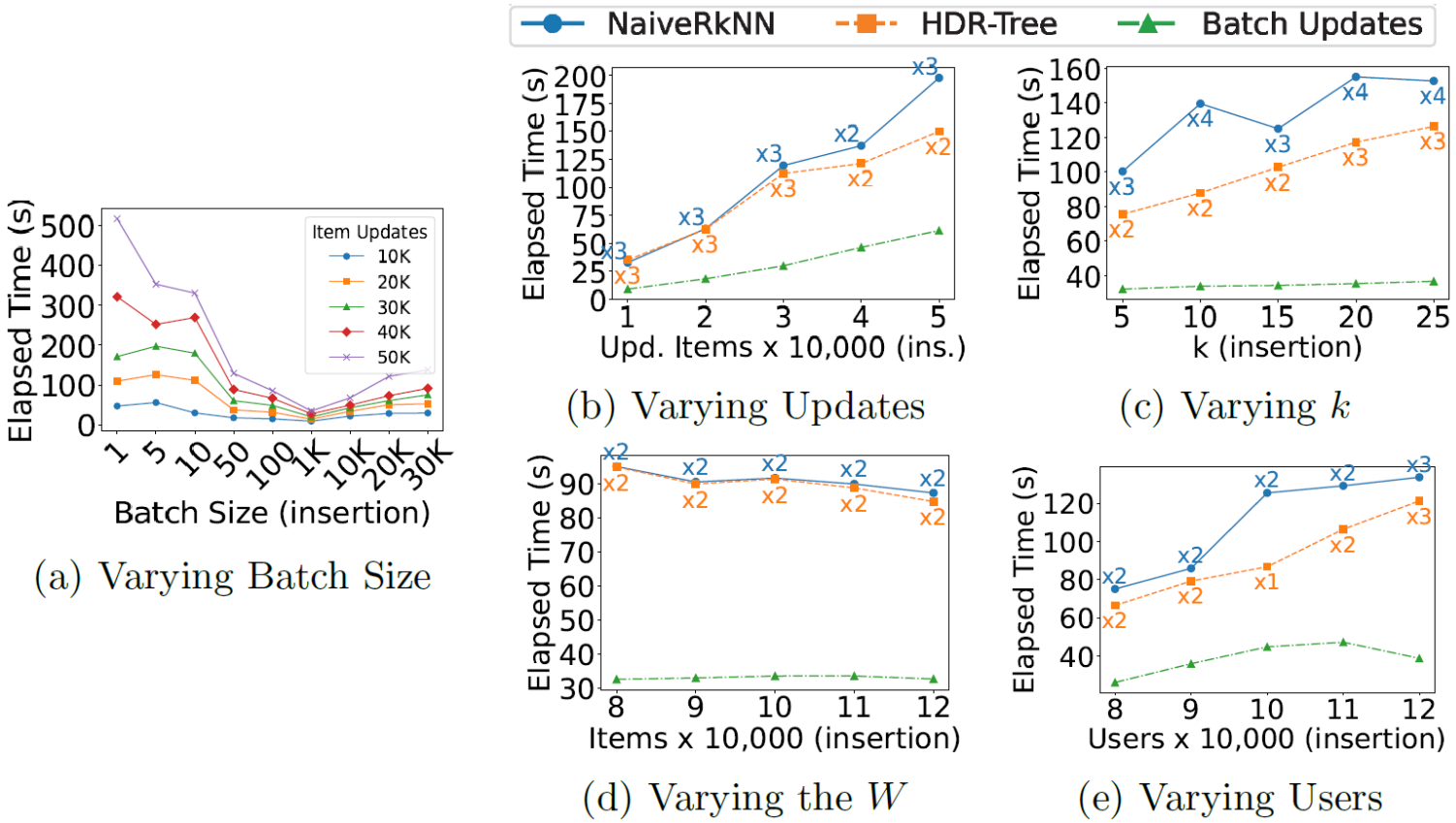


Fig. 4. Performance Analysis of Item Insertion with Varying Parameters

Experimental Results

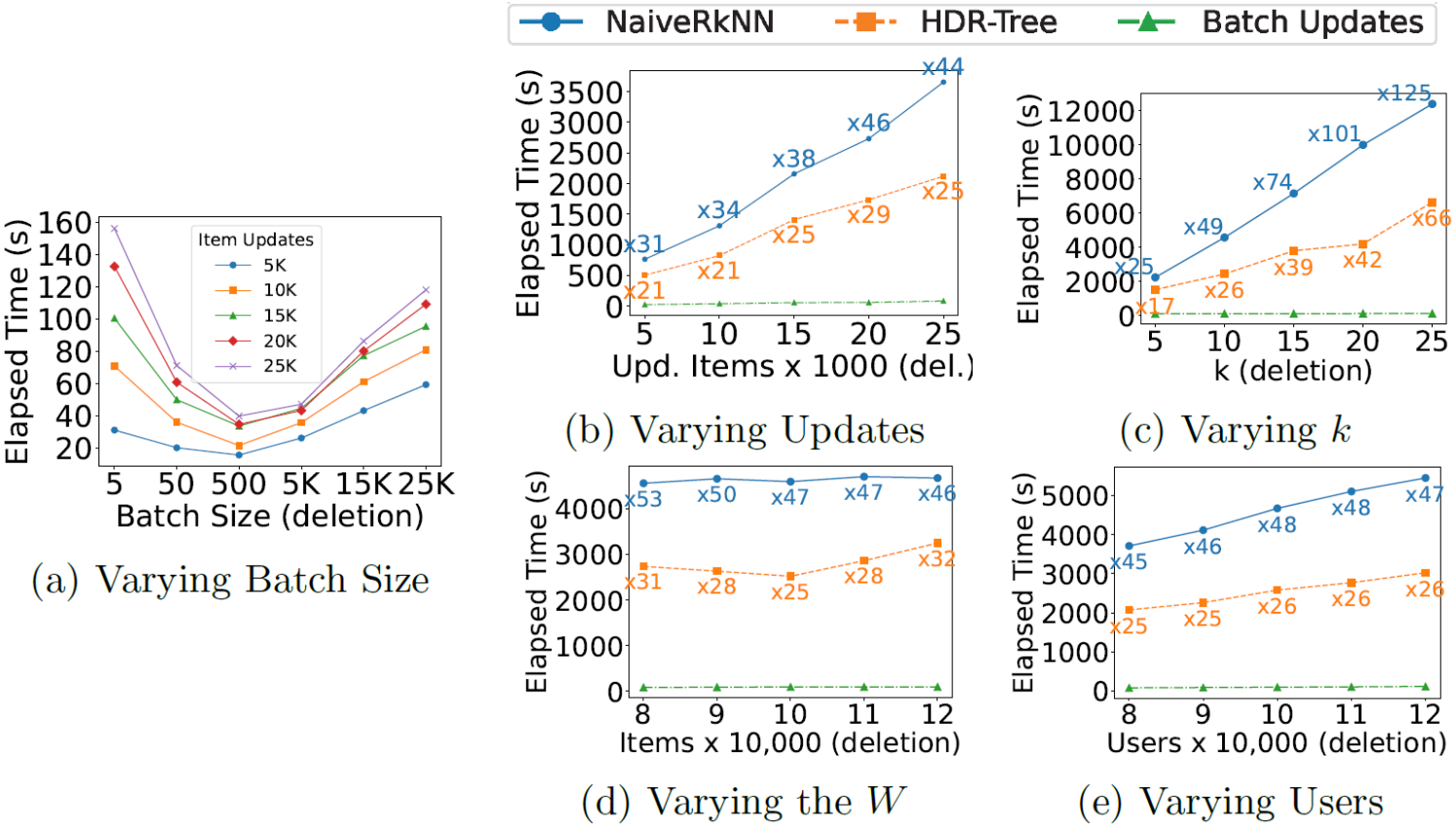
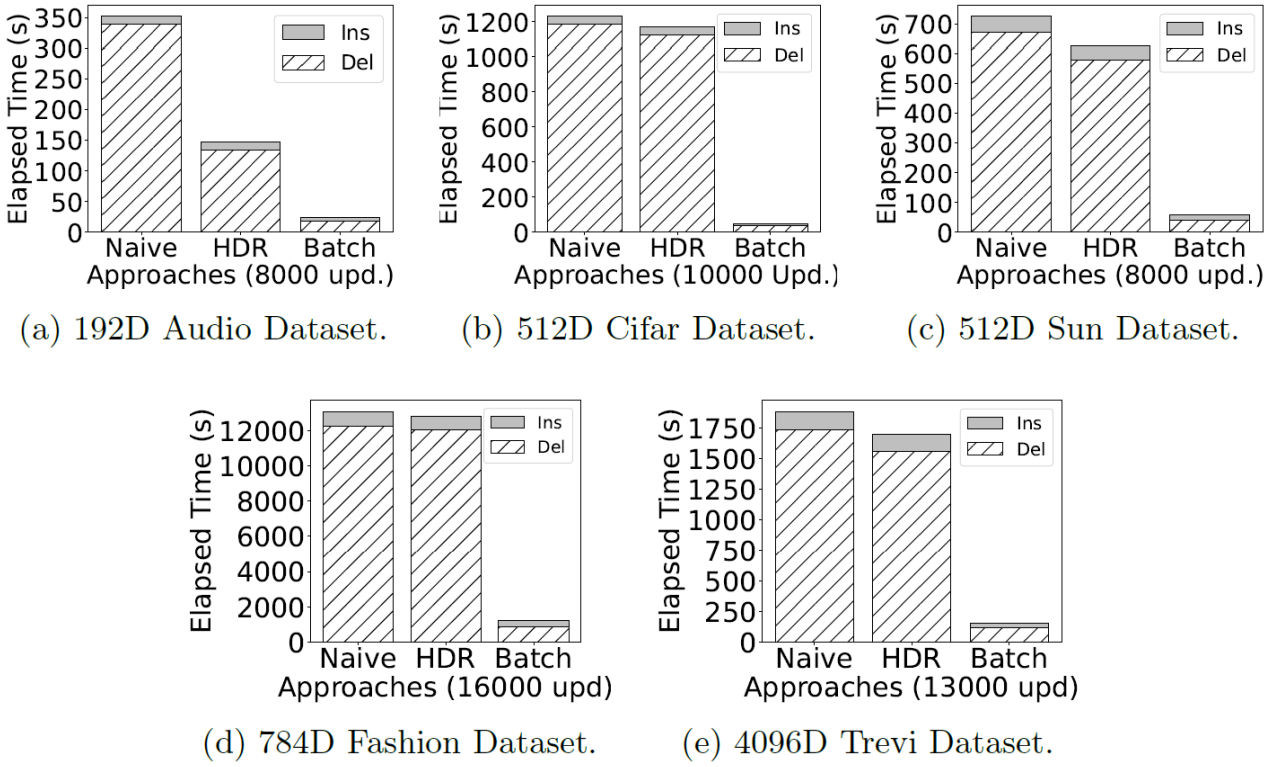


Fig. 5. Performance Analysis of Item Deletion with Varying Parameters

Experimental Results



(a) 192D Audio Dataset. (b) 512D Cifar Dataset. (c) 512D Sun Dataset.

(d) 784D Fashion Dataset. (e) 4096D Trevi Dataset.

Fig. 9. Performance Analysis of Varied-Dimensionality Datasets.

Conclusion

- **Summary**
  - Novel cluster-based approach for batch-dynamic kNN join.
  - Outperforms state-of-the-art methods significantly in efficiency and scalability.
- **Future Directions**
  - Parallel and distributed extensions for larger datasets.
  - Exploration of approximate solutions for extremely high-dimensional scenarios.

Thank You