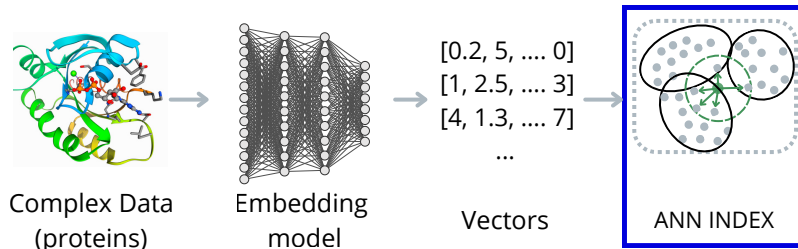


Temporally Dynamic Learned Index for Similarity Search over Complex Data

Author: Mgr. Emma Sommerová
Supervisor: RNDr. David Procházka
Consultant: doc. RNDr. Vlastislav Dohnal, Ph.D.

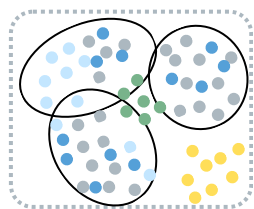
Motivation – Complex Data Analysis

Modern applications store complex data, such as proteins or images, as compact vector embeddings and use **indexes** to accelerate **approximate nearest neighbor (ANN)** search over these large datasets. **Learned indexes** further improve search quality by using machine learning to route queries to the correct data partitions. For example, AlphaFind [6] uses a **Learned Metric Index** [3] over AlphaFoldDB [4] to return the structures most similar to a query chain – avoiding an inefficient comparison against all hundreds of millions of vectors.



Problem – Dynamic Environment

Learned indexes assume a **static** scenario (all data available at build time), but real collections grow – AlphaFoldDB gains structures continually. As **inserts and deletes shift the data distribution**, the learned partitions become outdated.



A dynamic index needs to implement insert and delete operations without compromising search quality

Most indexes either need periodic rebuilds or underperform on at least one operation.

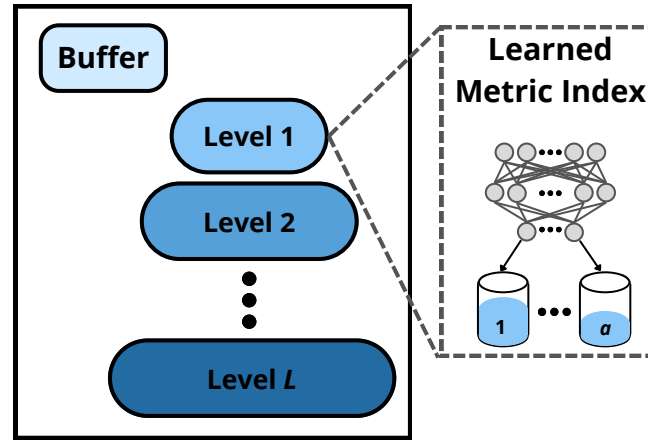
● Original data ● OOD insert
● Deleted data ● ID insert ● Distribution shift

References

- [1] J. L. Bentley and J. B. Saxe, "Decomposable searching problems I: Static-to-dynamic transformation," J. Algorithms, 1980.
- [2] Yu. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," IEEE TPAMI, 2020.
- [3] D. Procházka et al., "Scaling learned metric index to 100M datasets," in SISAP, 2024.
- [4] M. Varadi et al., "AlphaFold protein structure database," Nucleic Acids Res., 2022.
- [5] M. Douze et al., "The Faiss library," IEEE Trans. Big Data, 2026.
- [6] D. Procházka et al., "AlphaFind: discover structure similarity across the proteome in AlphaFold DB," Nucleic Acids Res., 2024.

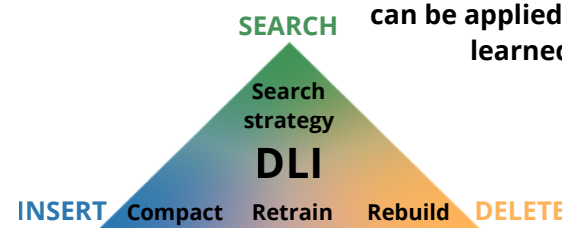
Solution – Dynamicity via Temporal Decomposition

Dynamic Learned Index



We introduce the **Dynamic Learned Index (DLI)**, which extends the **Bentley-Saxe decomposition** [1] into the domain of learned indexes, using the LMI as its base index.

- DLI organizes the data into several **levels**, each a static LMI, that are **created gradually** as vectors are inserted. This allows the index to adapt to updates by **local rebuilds** of smaller levels.
- We propose, implement, and evaluate **strategies for insert, delete, and search** that extend the multilevel structure to approximate search, e.g., normalizing model outputs to enable cross-level navigation during search.
- The method generalizes into a **framework that can be applied to any learned index.**



Analysis of Index Degradation

Recall, the standard metric for ANN indexes reveals that an index has degraded, but not what caused it. We propose three measures that explain **why** this happens, not just **when**.

1. Partitioning Quality of Index

- GT Buckets** – number of buckets containing true neighbors for a given query
- GT Candidates** – number of objects in the buckets containing GT

2. Navigation Quality of Index

- Wasted Effort** – how many index needed to process, compared to ideal ordering

Experimental Evaluation

- 1200+ experiments
- 4 datasets, up to 100M vectors.
- Compared DLI to three reference indexes: HNSW [2], IVFFlat [5], and LMI

Index comparison (Insert/Delete workloads)					
	Memory MB	Insert/s ↑	Delete/s ↑	Query/s ↑	Recall ↑
DLI	32	6652	5808	66	93.5
HNSW	6037	56	–	493	91.3
IVFFlat	31	6297	584	67	95.0
LMI	37	3870	7160	15	97.4

↓ Memory MB Insert/s ↑ Delete/s ↑ Query/s ↑ Recall ↑ Worst

We do not report delete throughput for HNSW, which only marks objects as deleted and filters them at search time rather than physically removing them.

DLI bridges gaps between other indexes, proving to be viable in unpredictable dynamic scenarios.