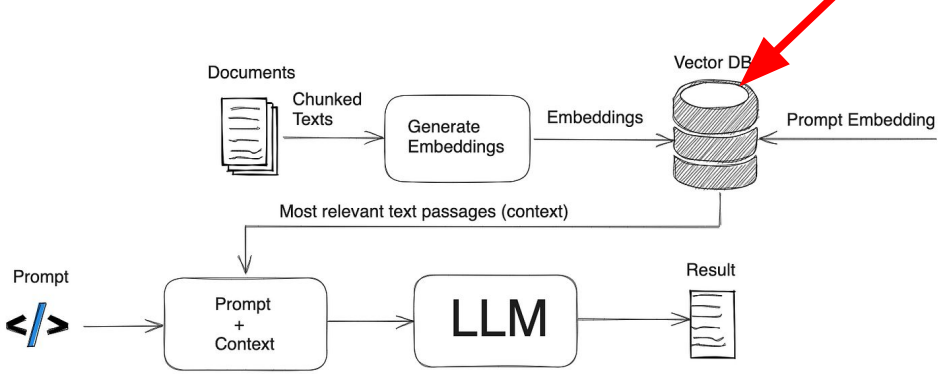
LEANN : A Low-Storage Vector Index

Presented by **Yichuan Wang**

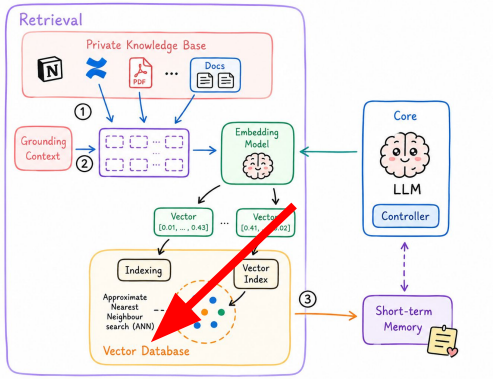
Collaboration with Zhifei Li, Shu Liu, Yongji Wu, Ziming Mao, Yilong Zhao, Xiao Yan, Zhiying Xu, Yang Zhou, Ion Stoica, Sewon Min, Matei Zaharia, Joseph E. Gonzalez

UC Berkeley

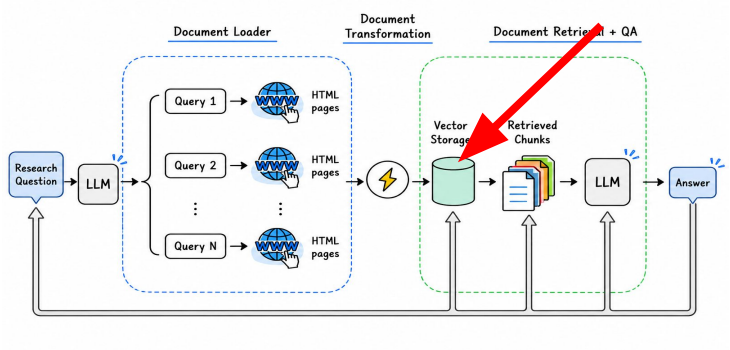
Vector Search is Central to the LLM and Agent Era.



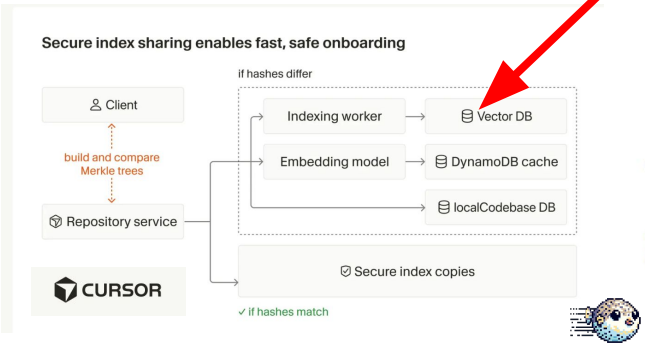
RAG



Agent Memory



Search agent and Search Engine



Coding Agent

Chroma
milvus
Pinecone
elastic
Weaviate
turbopuffer

Secret Cost of Advanced Vector Index

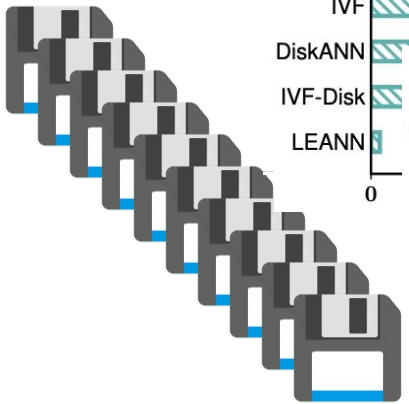
76GB of Docs

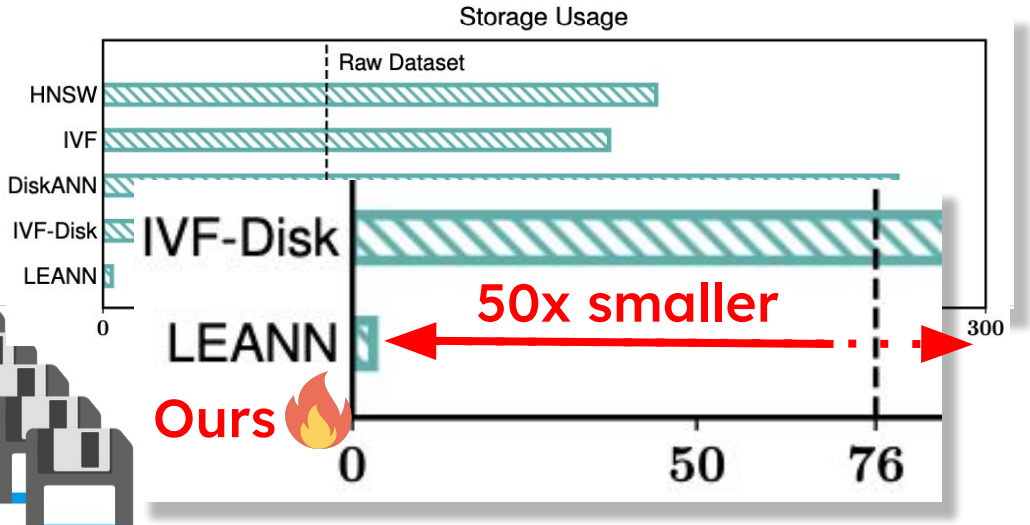
→ **~200GB** of index storage

➢ **270% storage overhead**

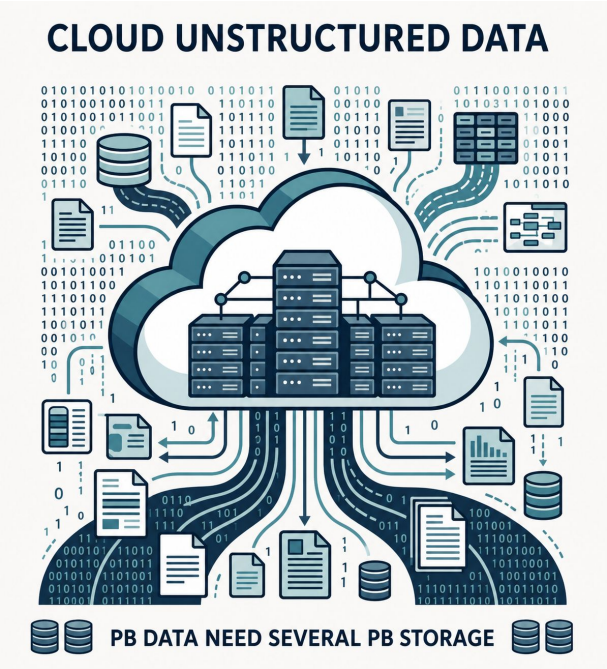
Huge Storage Overhead

→

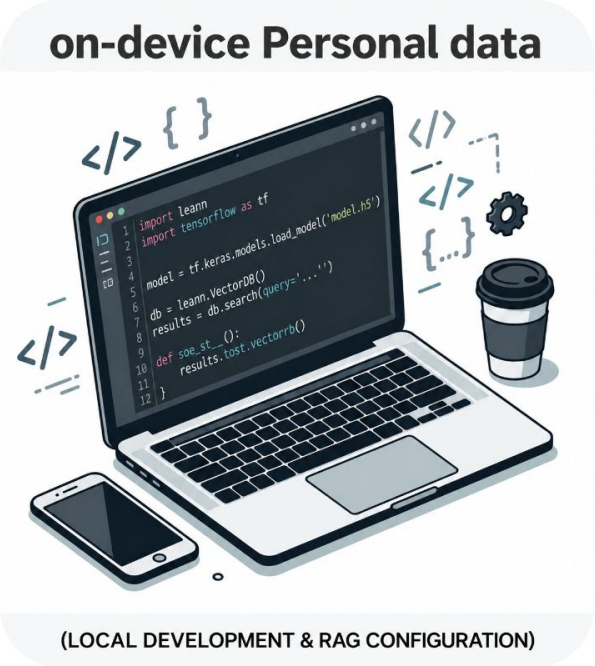




Goal: Enable Vector Search Over Everything



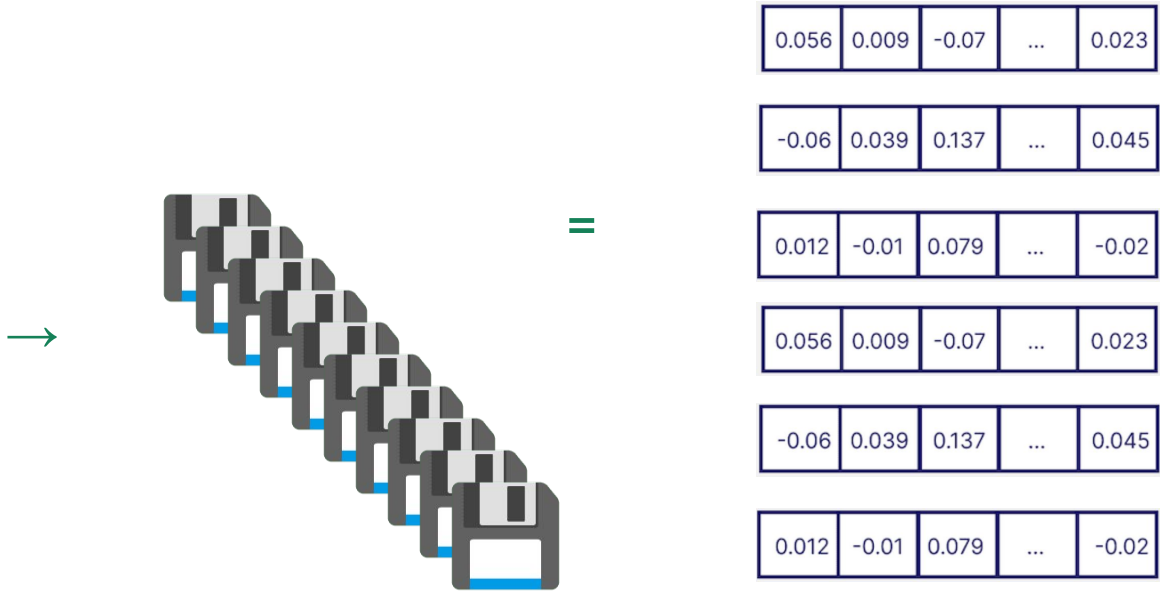
Impractical



Impossible ← Main focus

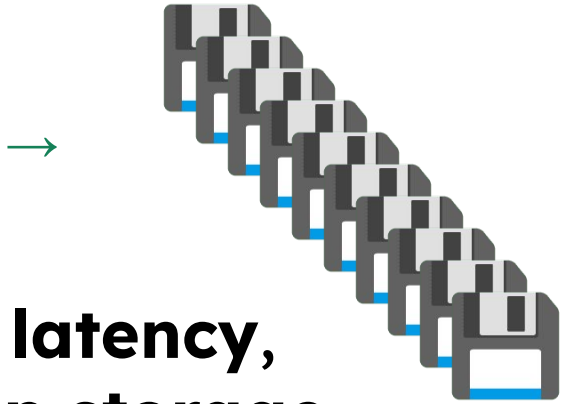
Secret Cost of Advanced Vector Index

➢ High dimensional embedding vectors

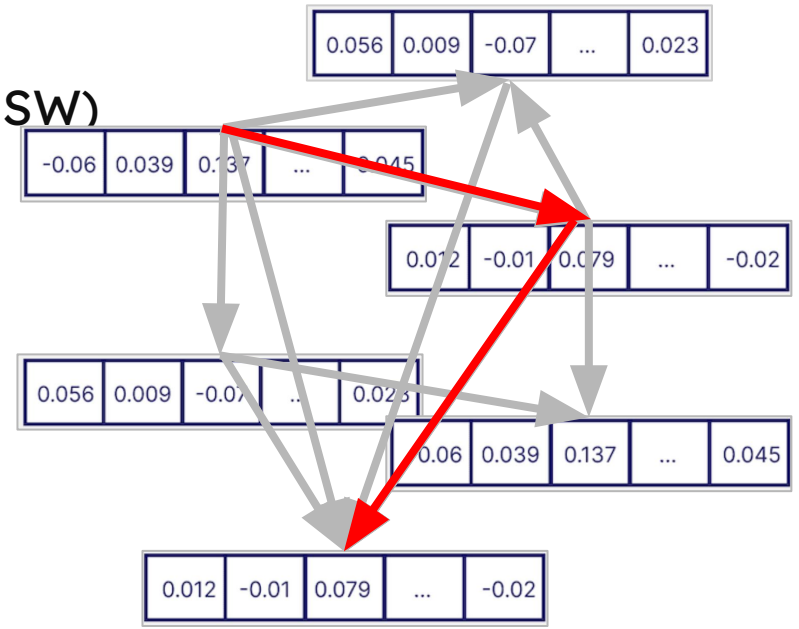


Secret Cost of Advanced Vector Index

➢ High dimensional embedding vectors
➢ Complex graph data-structures
Hierarchical Navigable Small World (HNSW)



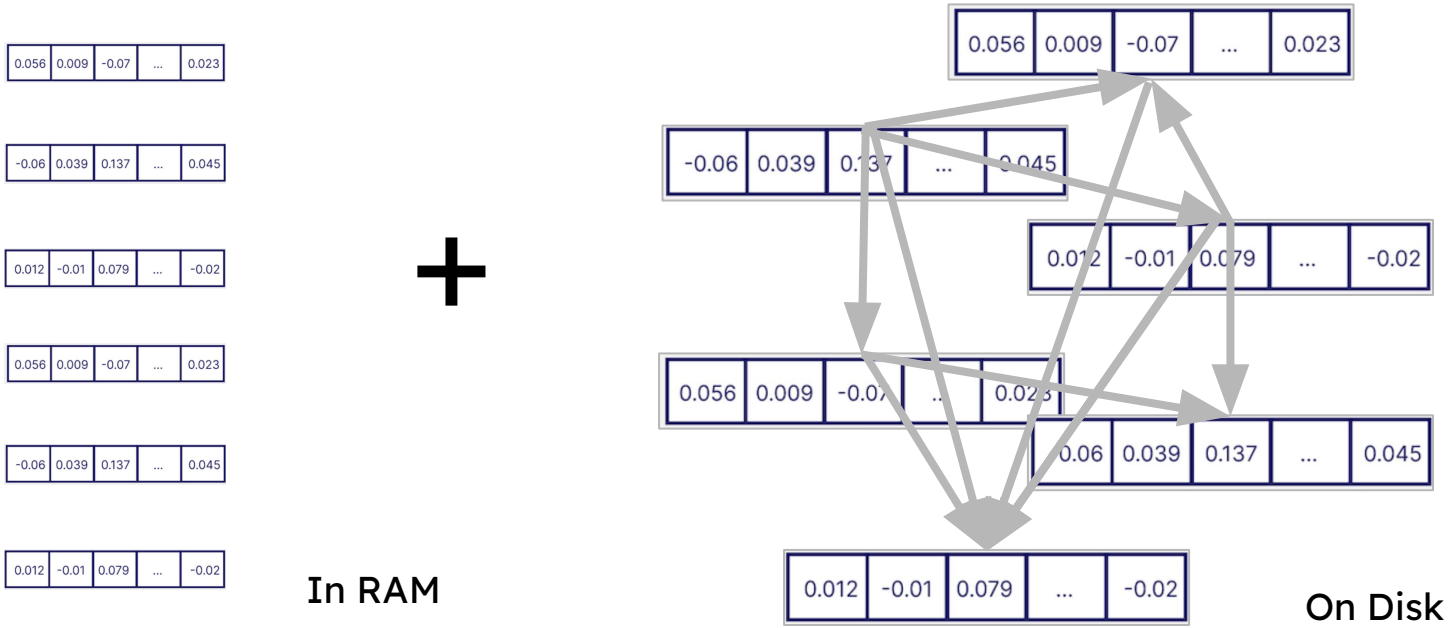
=



HNSW **cuts latency**,
but **blows up storage.**

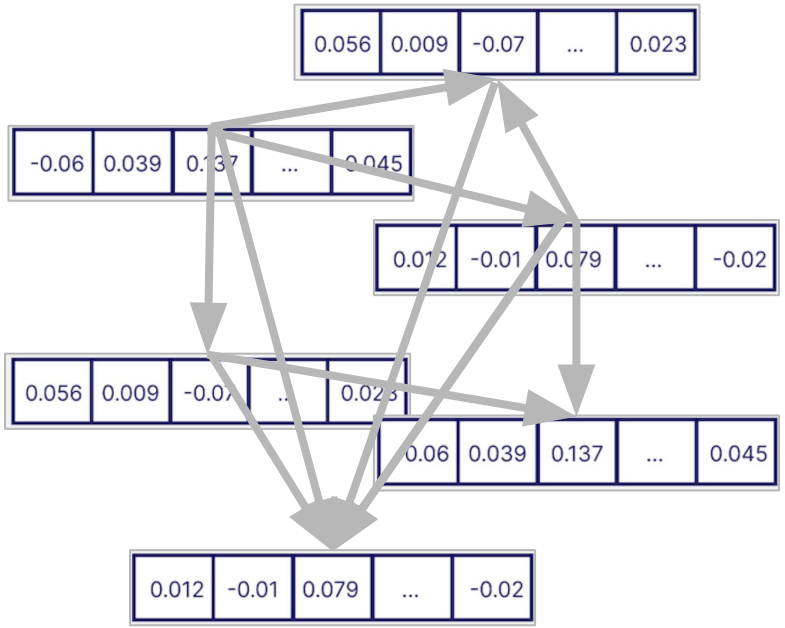
Existing Solution

- DiskANN and Starling
  - **Reduce memory overhead** but still **requires too much disk space**



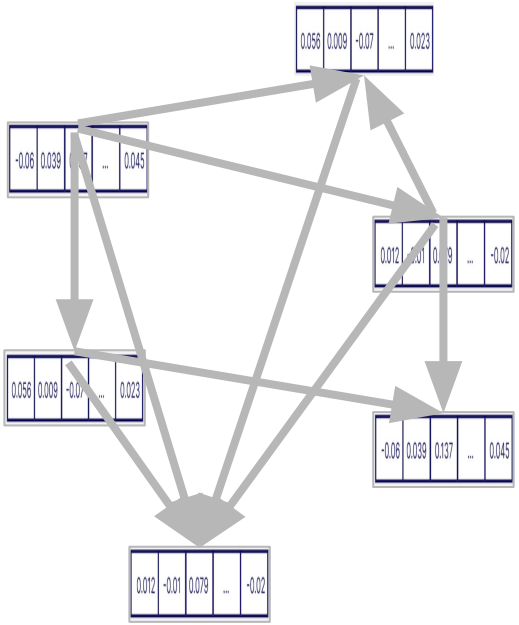
Existing Solution

- Product Quantization (PQ), RabitQ
  - Quantization methods **reduce recall accuracy**



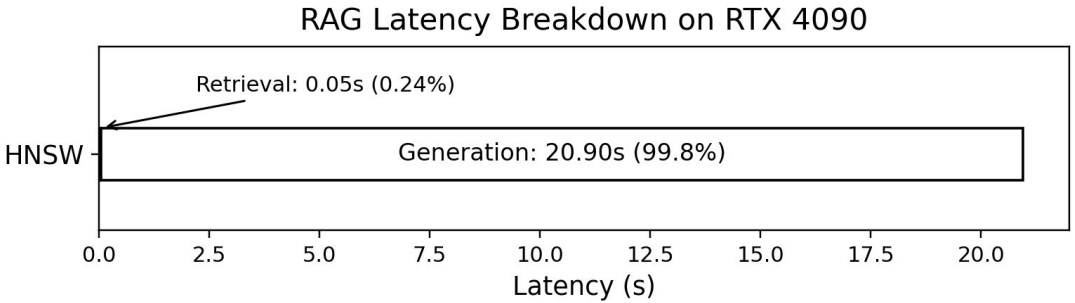
Existing Solution

- Product Quantization (PQ), RabitQ
  - Quantization methods **reduce recall accuracy**



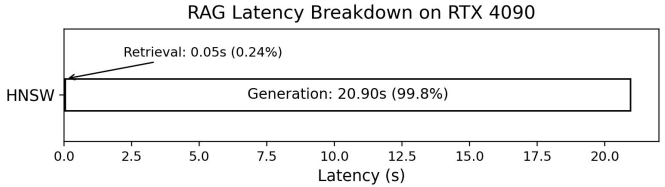
New Opportunities in (On-Device) RAG/Agent

- **Generation time dominates** the end-to-end RAG latency
  - Reasoning model & Long context input



New Opportunities in (On-Device) RAG/Agent

- **Generation time dominates** the end-to-end RAG latency
  - Reasoning model & Long context input



- The **QPS is not that high** on device

Opportunity: increase latency to unlock significant storage savings → Index Everything!

Two Key Insights in LEANN

1. HNSW graph traversal touches only **a small subset of vectors** per query.
   ➢ **Don’t save vectors → recompute them.**
2. Much of the HNSW graph structure is **redundant**
   ➢ **Prune the graph**

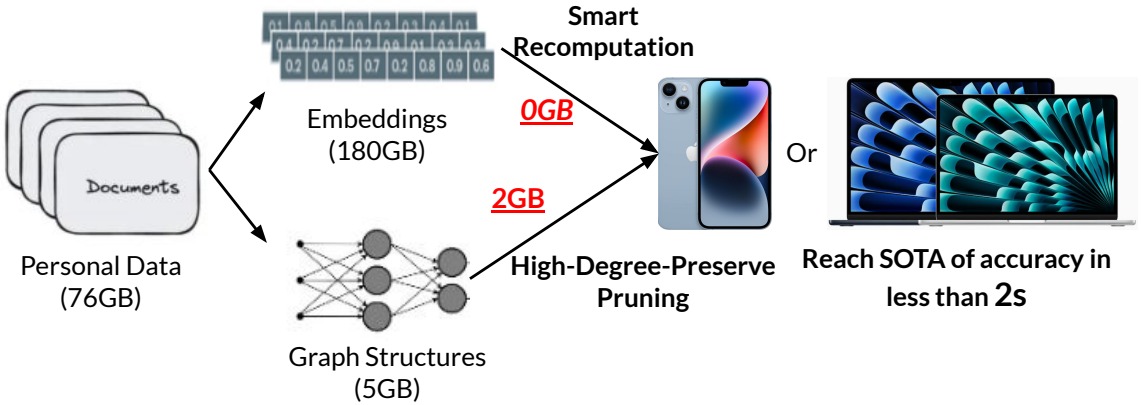
Trade-off: slight increase in latency for significantly reduced storage

Enable state-of-the-art RAG!

LEANN - Solution

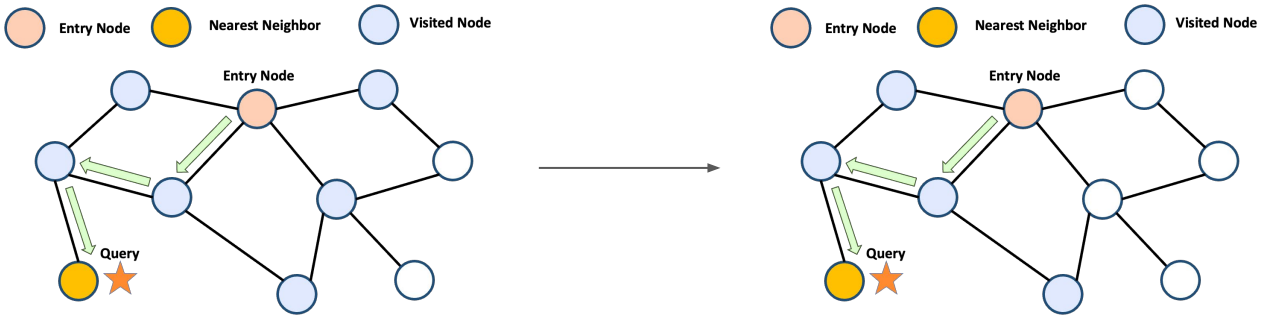
Fast HNSW Graph-Based Recomputation with two-level search and batched execution eliminates the need for storing pre-computed embeddings, while keeping latency low.

High-Degree Preserving Graph Pruning removes redundancy in the graph, cutting storage costs with minimal quality loss.



LEANN - two-level search

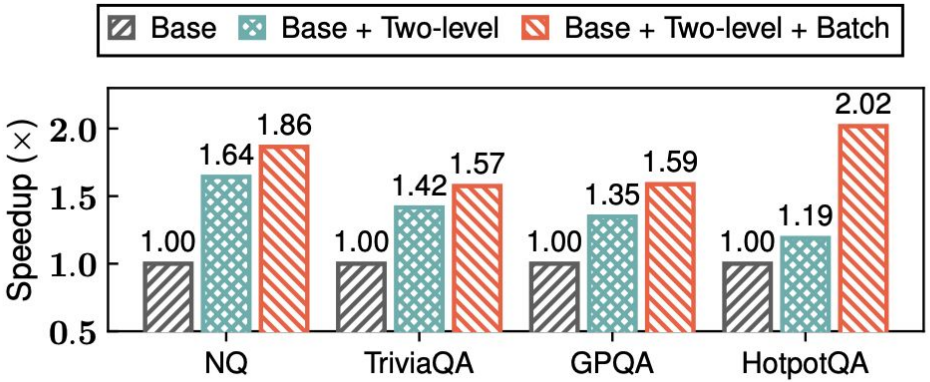
- Fast approximate evaluation of **all candidates**
- Applying exact computations only to **the most promising candidates**



*Please refer to the detailed algorithm in the paper.

LEANN - Dynamic Batching

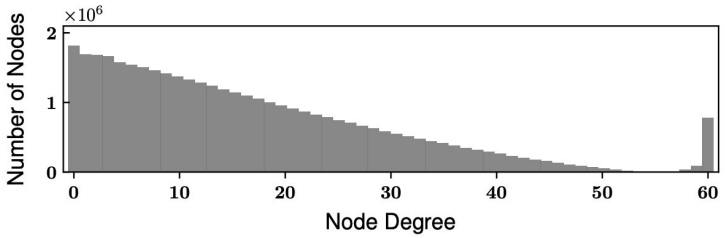
- Characteristic of GPU: **Batch effect**
  - **Batching nodes within one hop** to be recomputed
  - **Batching nodes acrocess hops** to be recomputed



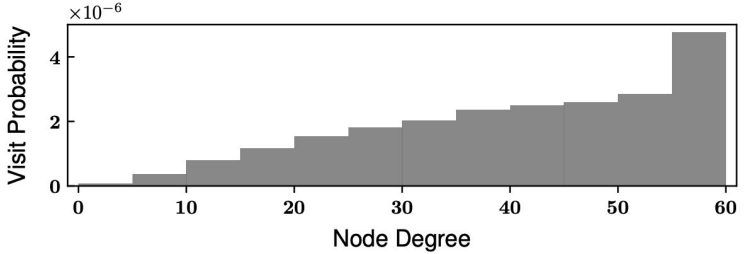
Results on RTX 4090

LEANN - Graph Pruning

- In HNSW, the metadata of graph is still large
  - How to reduce the graph size?
    - **Observation**: skewed node distribution and access pattern
    - **Solution:** keep high degree nodes**!**



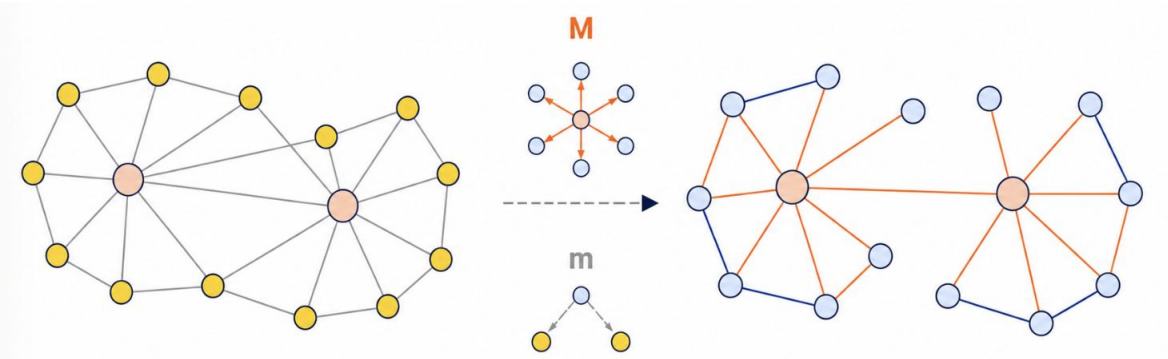
(a) Node (out)-degree distribution



(b) Node access probability.

LEANN - Graph Pruning

- High Degree Preserving Graph Pruning
  - Keep high out-coming edges on a small portion of **popular nodes**
  - **Constraint out-coming edges** for most of nodes
    - Still allow adding from others



*Please refer to the detailed algorithm in the paper.

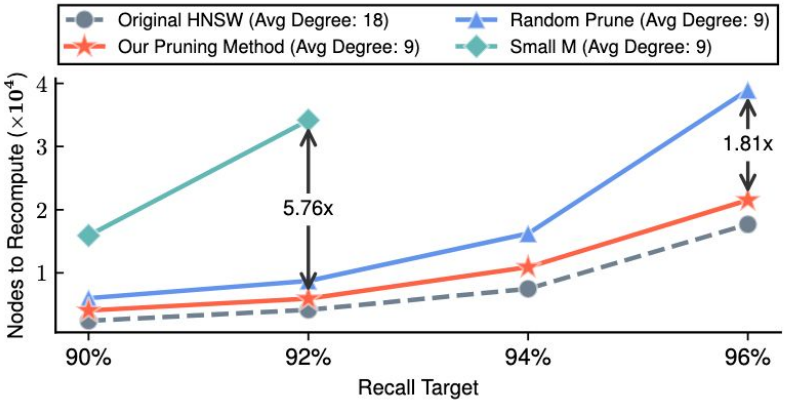
LEANN - Graph Pruning

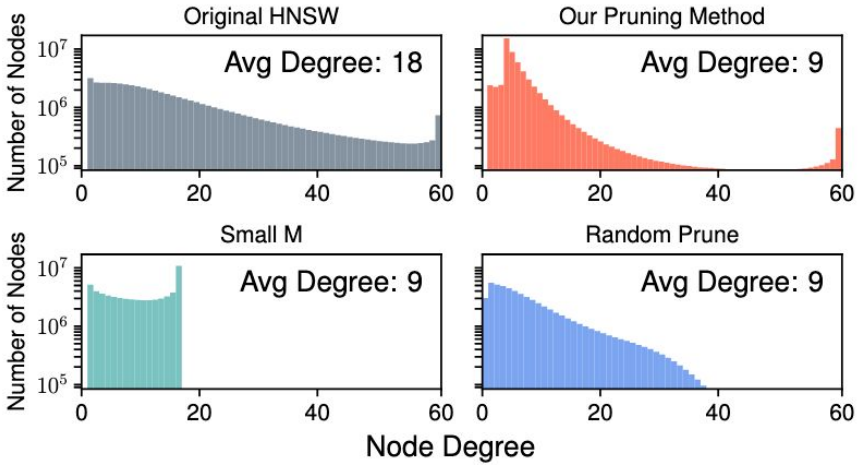
Search Performance:

LEANN **compress graph 2x being nearly lossless**

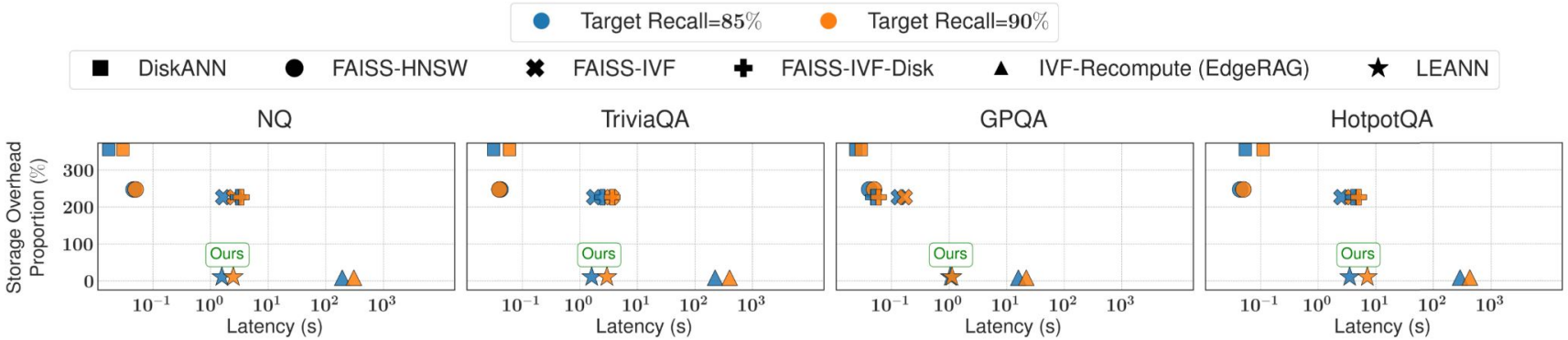
Degree distribution:

LEANN successfully **retain High degree nodes**





Results

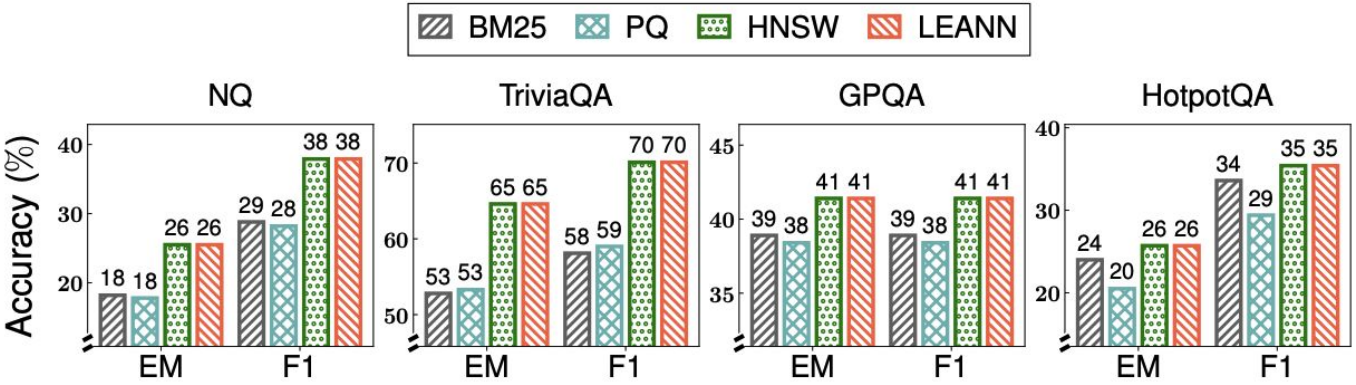


Reduces index size to **under 5% of the original raw data size**,

➢ achieving up to **50 times smaller storage** (>200G data),

➢ search under **1 seconds** across real-world benchmarks on an RTX 4090, adding only a **5% increase** in end-to-end RAG latency.

Accuracy Results



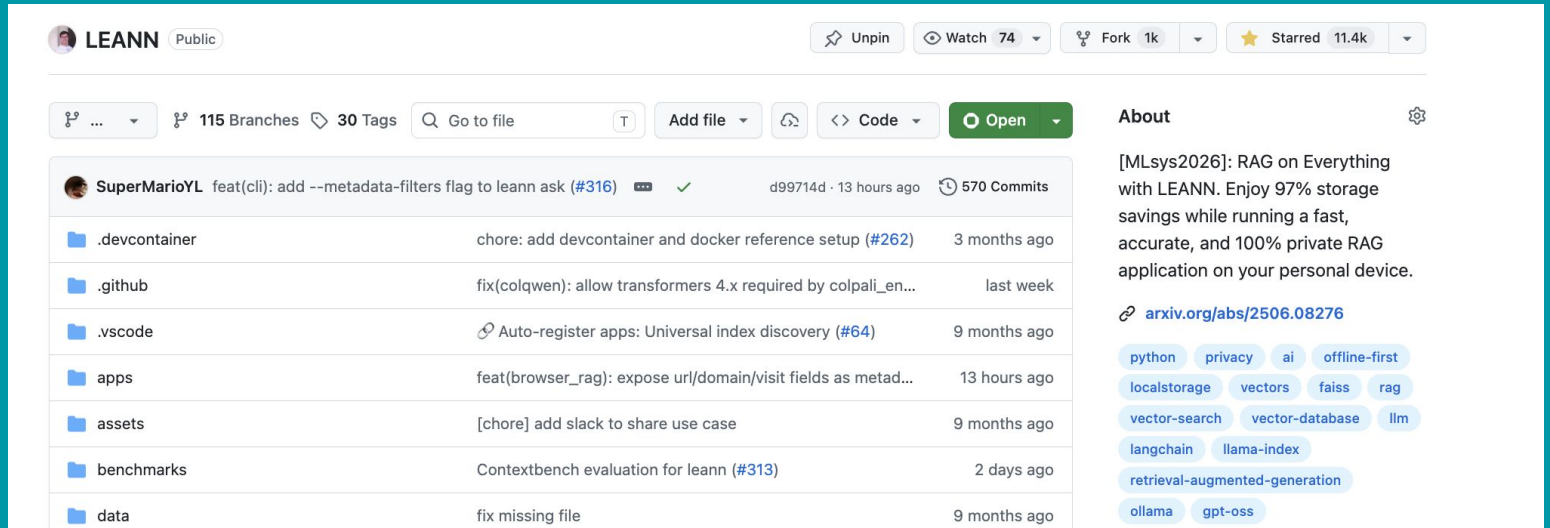
Better than BM25 and PQ(even larger storage than us) on most benchmark

LEANN - Storage Efficient Build and Update

- Reduce the **Peak Storage overhead** duration **building**
  - Soft assignment using sampled k-means
  - **Shard-wise graph construction**
  - **Graph merging**
- **Storage Efficient Add**
  - Using a temporary buffer to help

*Please refer to the detailed algorithm in the paper. The index construction cost is also reported in the paper.

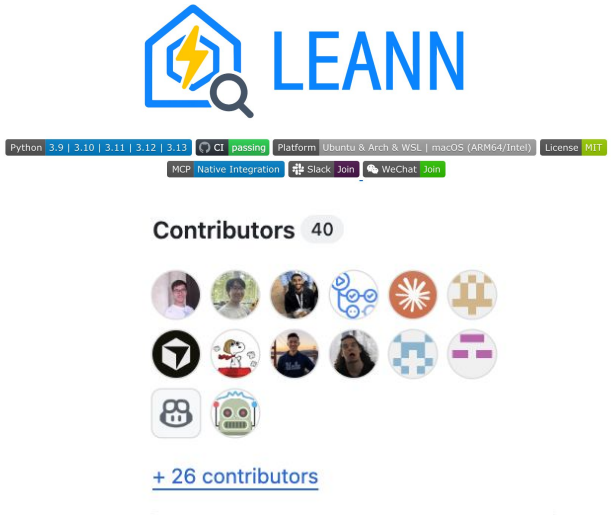
Open-source: RAG Everything with LEANN

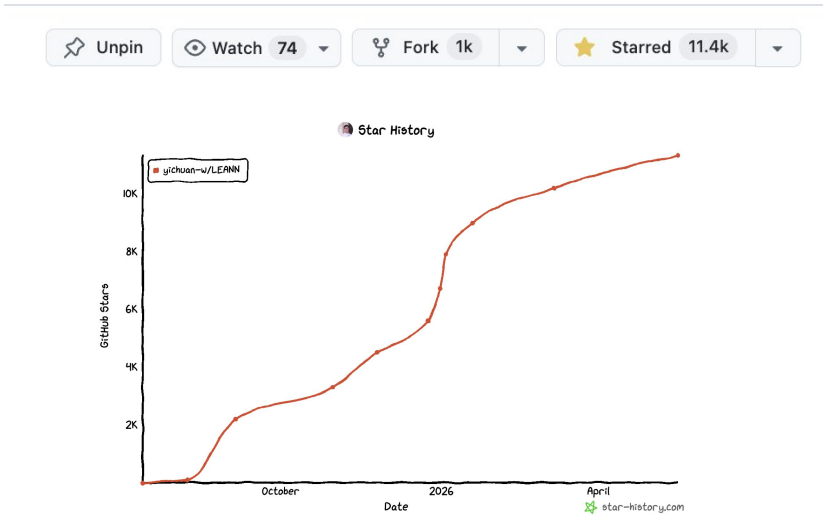


LEANN is Real! Growing Community

repo

- A lightweight, private toolkit for vector search and RAG, especially on your own computer.
- Rapidly gained a large user base **within months of release.**
  - **11k+ ⭐ stars, 1k+ forks**



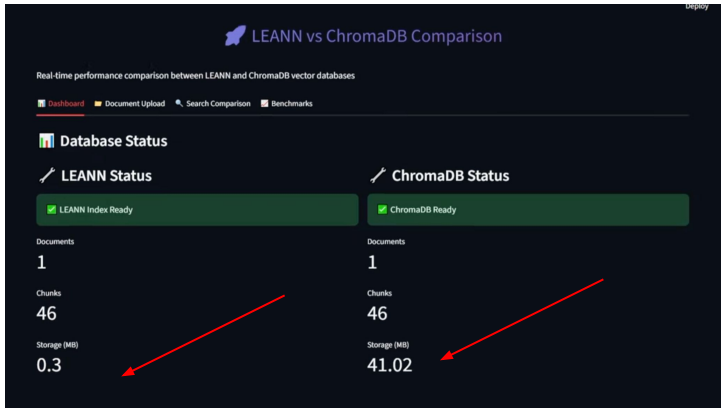


It works!

Lightweight & Deployed by community

📊 Storage Comparison

System	DPR (2.1M)	Wiki (60M)	Chat (400K)	Email (780K)	Browser (38K)
Traditional vector database (e.g., FAISS)	3.8 GB	201 GB	1.8 GB	2.4 GB	130 MB
LEANN	324 MB	6 GB	64 MB	79 MB	6.4 MB
Savings	91%	97%	97%	97%	95%



Muthu Kumaran ✓ · 1st
Staff Data Engineer
1d · Edited ·

The World's Smallest Vector Database: How LEANN Uses 97% Less Space Than Traditional Vector DB

ChromaDB, Pinecone, PgVector and Weaviate demand 1.5-7x more storage than your original data. A 1GB document collection becomes 7GB in your vector database.

This forces a brutal choice: expensive cloud subscriptions or bloated infrastructure. Privacy-conscious teams get hit hardest.

LEANN changes this equation completely. LEANN is an innovative vector database from Yichuan Wang and his team that democratizes personal AI.

I tested LEANN against ChromaDB on identical data, with the same embeddings model (nomic-embed-text from ollama) for the PDF file of 1 MB

Storage Comparison

ChromaDB: 41 MB storage required
LEANN: 0.3 MB storage required

136x Improvement

Result: 99.3% storage reduction with identical search accuracy

That's a 136x improvement.

Ivan Groenewald ✓ · Following
Chief Technology Officer @ Extraordinary Managed Services Limited | All...
4w ·

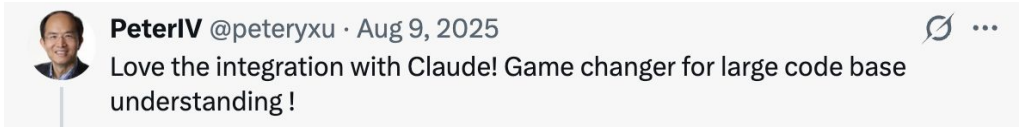
This actually works. It's easy to run and fun to use.

Tested by feeding in a complicated dataset, which produced a very small on-disk index.
Backed by Ollama running qwen3:4b as the LLM on a Mac Mini M2 Pro.

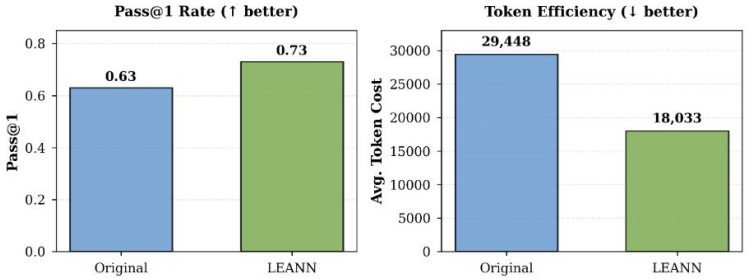
All local. All on-device. Minimal footprint.

Features: Claude-code Native

- **The first MCP-based** semantic retrieval engine for Claude Code
  - AST chunker #58
  - MCP server [link]
  - Merkle Tree integration #219
  - Search across repo
- Result:
  - 2x cost saving, 4x latency saving



LEANN **improves accuracy** and **reduces cost** for *Claude Code*



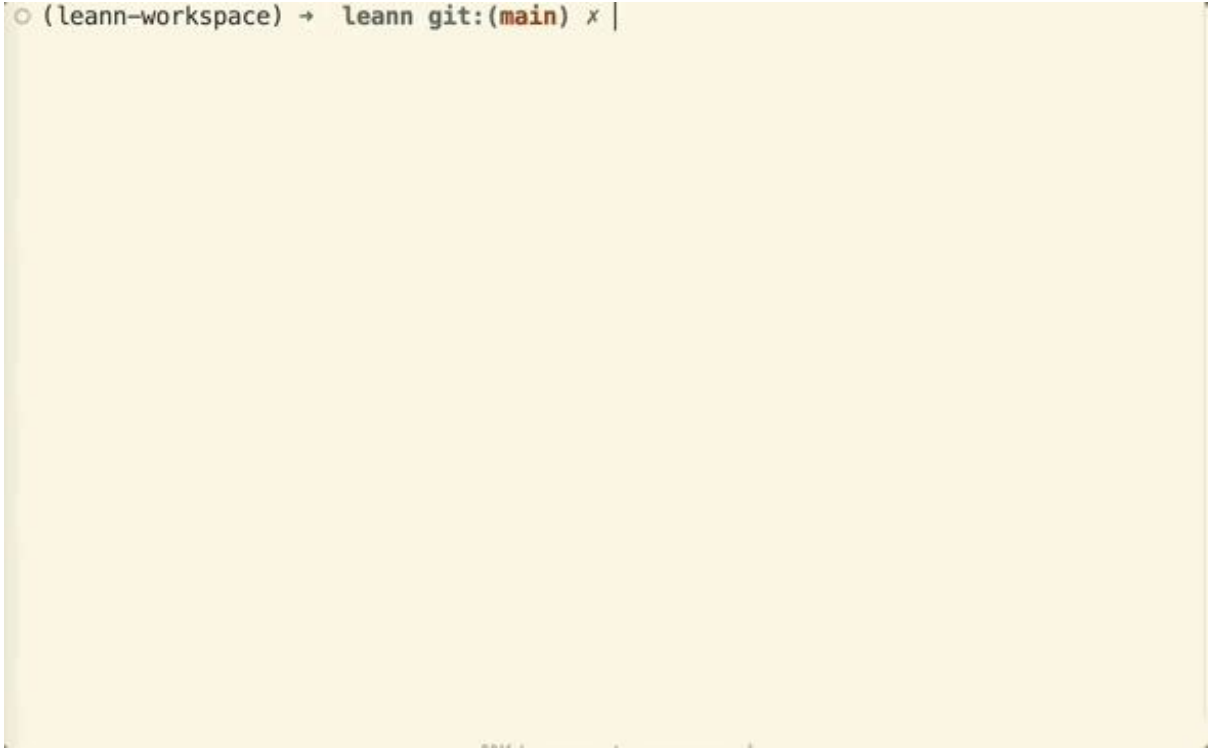
Swe-bench results from community [link]

Diverse applications:

- **Diverse Built-in application**
  - Email, Chat, Search history
  - File system (replace your spotlight)
  - Codebase
  - External knowledge bases (i.e., 60M documents)
  - Images #183, #162
  - Agent memory #131
  - Live application (slack, twitter #131)
  - …
- Features:
  - **Meta-data filter** (Range filter) #75
  - **Hybrid Search** (grep + vector search): #87, (BM25+ vector search): #211
  - **Instruct embedding model** #171
  - …

Demo:

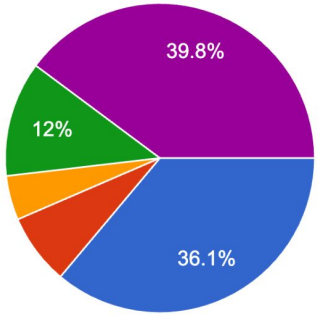
Query: “How many classes recommend to take for incoming EECS PhD”



LEANN in the wild

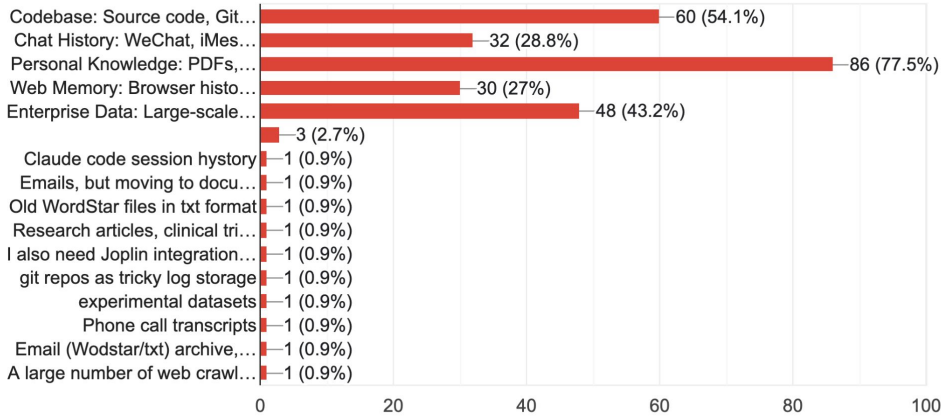
3. Does your work/company have business scenarios that align with LEANN? (Optional)

108 responses



8. What data are you currently indexing?

111 responses



Conclusion

Conclusion

Join the project!

Problem: Modern vector search is bottlenecked by **index storage**

Solution: LEANN **efficiently recomputes embeddings** on the fly and compresses the graph structure

Results: LEANN achieves reduces storage by up to 50x while matching the accuracy of HNSW with only a 5% increase in end-to-end RAG latency

Team: Yichuan Wang, Zhifei Li, Shu Liu, Yongji Wu, Ziming Mao, Yilong Zhao, Xiao Yan, Zhiying Xu, Yang Zhou, Ion Stoica, Sewon Min, Matei Zaharia, Joseph E. Gonzalez

Thank you!

Future Work

Looking Ahead

- Scaling low-storage vector indexing to **production-scale** semantic search (log analysis, Parquet-compatible, etc.)
- Support large-scale vector search over **real-world, high-quality corpora** for serving/post-training/agents
  - DS-Serve is the first prototype we developed! We aim to **reshape the search stack** in the era of agents.
  - **Poster session alert!**
- Unlocks new frontiers with richer, more personal, **multi-modal experiences—spanning vision (ongoing work),** audio, and beyond.

DS-Serve

Vision of LEANN repo:

- Seamlessly interacts with **all your private data**
- Builds **long-term, local AI/Agent Memory**
- Runs at minimal cost, with **zero cloud reliance**
- Unlocks new frontiers with richer, more personal, multi-modal experiences—spanning **vision (ongoing work), audio, and beyond.**

QA

Thank you!