DS SERVE: A Framework for Efficient and Scalable Neural Retrieval

Jinjian Liu[1*], **Yichuan Wang**[1*], Xinxi Lyu[2], Rulin Shao[3], Joseph E. Gonzalez[1], Matei Zaharia[1], Sewon Min[1]

[1]UC Berkeley [2]UIUC [3]University of Washington [*]Equal contribution

Key Result: RAG Performance with DS SERVE + LLM

DS SERVE consistently improves accuracy across reasoning-intensive benchmarks.

Figure: Downstream accuracy (%) with LLaMa 3.1 8B Instruct; Exact Search provides further gains

Overview

DS SERVE transforms large in-house datasets into **high-throughput**, **memory-efficient** neural retrieval systems with web UI and API.

- **Scale:** 400B words, 2B vectors, 5TB embeddings
- **Throughput:** Up to **10,000 QPS** (index-level)
- **Memory:** <200 GB RAM
- **Accuracy:** Matches/exceeds commercial search APIs

DS SERVE is the largest publicly accessible vector store.

Motivation: Why is This Hard?

- **Scaling neural retrieval is hard:** High throughput + low memory + strong accuracy at billion-scale is non-trivial.
- **Traditional ANN doesn't scale:** IVFPQ suffers latency-accuracy tradeoffs; HNSW demands excessive RAM.
- **End-to-end tooling is lacking:** Few frameworks offer ready-to-use retrieval with web UI, API, and feedback collection.

System Architecture

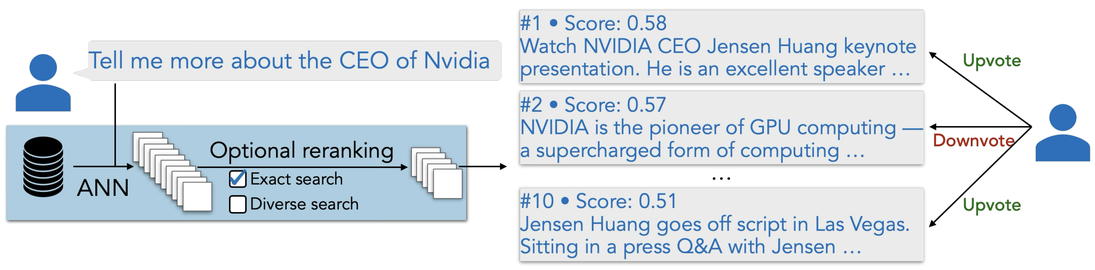


Figure: DS SERVE pipeline: Query → ANN retrieval → Optional reranking → Top-k results with voting

Datastore: CompactDS

DS SERVE is built on **CompactDS**, a high-quality corpus comparable to much larger Common Crawl data:

- **380B words** (∼2B vectors)
- Web crawl, Wikipedia, research papers
- **Largest publicly available** neural retrieval datastore

Prior work: MS MARCO (≤10M vectors), Wikipedia, BEIR. Commercial DBs impose limits well below billion-scale.

Technical Approach: Why DiskANN?

The Problem with IVFPQ:

- Heavy quantization → accuracy loss
- Increasing `nprobe` → throughput drops
- Memory-bound at billion scale

DiskANN Solution:

- Compressed vectors in RAM
- Full-precision vectors + graph on NVMe SSD
- Implicit reranking during graph traversal
- Massively parallel I/O

Feature	IVFPQ (Traditional)	DiskANN (DS Serve)
Accuracy	**Lower**: Quantization noise reduces recall.	**Higher**: Full-precision vectors on disk ensure high recall.
Throughput	∼**100 QPS**: More distance computations.	>**10K QPS**: Fewer computations via navigation graph and parallel I/O.
Latency	**Higher**: Sequential inverted list scanning.	**Lower**: Efficient graph traversal.

DiskANN vs IVFPQ

DiskANN is **more accurate AND faster** than IVFPQ at recommended configs.

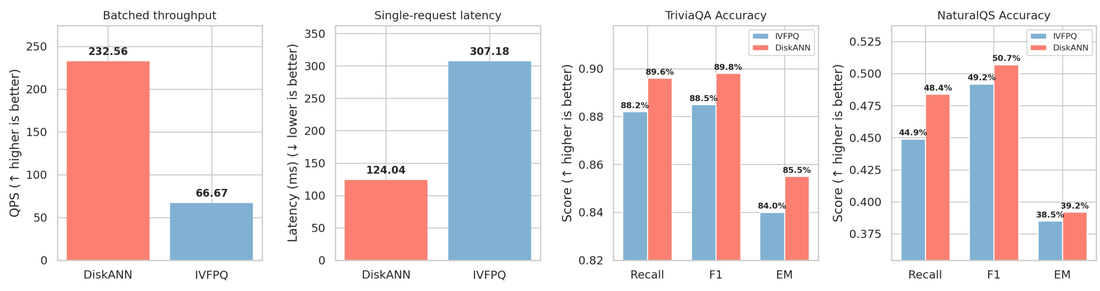
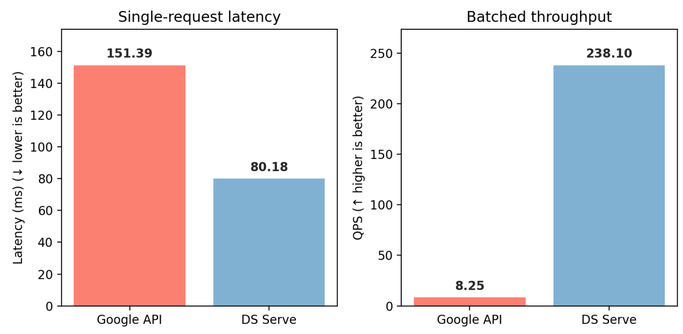


Figure: DiskANN: ∼**2.3**× **throughput**, ∼**2.2**× **lower latency** vs IVFPQ

Applications

1. **Retrieval-Augmented Generation (RAG)**
   - Superior accuracy vs commercial search
   - Low latency for interactive use
   - Powers efficient RAG with high-quality results
2. **Data Attribution & Curation**
   - Semantic attribution over pre-training corpora
   - More robust than N-gram matching for paraphrased queries
   - Deduplication, decontamination, filtering
3. **Training Search Agents**
   - High-frequency rollouts without rate limits
   - Free, controllable backend
   - Removes API bottlenecks in RL training
4. **Research Benchmarks**
   - Built-in voting for labeled data collection
   - Handles long/complex queries effectively

DS SERVE vs Google Search API



DS SERVE: ∼**30**× **throughput**, ∼**2**× **lower latency—free of costs**.

Optional Search Modes

Diverse Search (UI toggle):

- Applies MMR to reduce redundancy
- Use when results contain duplicates

Exact Search (GPU required):

- Reranks ANN candidates with GritLM
- Improves accuracy for harder queries

DiskANN Ablation Studies

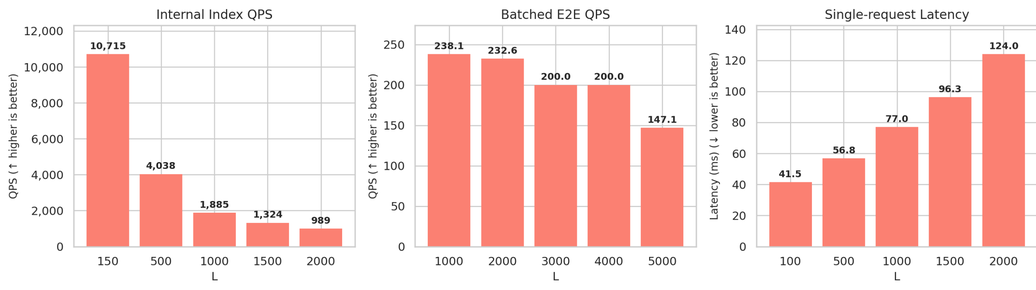


Figure: Search list size L controls accuracy-latency tradeoff

L≈100 sufficient for most queries; higher L improves hard queries.

Try It Now!

- **Web UI:** `http://api.ds-serve.org:30888/ui`
- **Code:** `github.com/Berkeley-Large-RAG/RAG-DS-Serve`
- **Paper:** Available on project page

Acknowledgements: CompactDS, Massive Serve, FAISS, DiskANN