Quantization-Enhanced HNSW for Scalable Approximate Vector Search

Dinesh Koilada
dineshkoilada@gmail.com

***Abstract*—This paper presents a novel optimization strategy for high-performance approximate nearest neighbor (ANN) search, a critical requirement in modern vector search applications driven by large language models and retrieval-augmented generation. Addressing the inherent memory and latency challenges of the popular Hierarchical Navigable Small World (HNSW) algorithm, we introduce HNSW-LVQ (Locally Adaptive Vector Quantization for HNSW). Our methodology incorporates a per-dimension quantization scheme that efficiently compresses floating-point vectors into integer representations, thereby significantly reducing memory overhead and accelerating distance computations. Empirical validation on the SIFT 10K dataset demonstrates that HNSW-LVQ achieves a remarkable 85% reduction in query latency and substantial memory enhancement with only a marginal 2% decrease in recall. This research validates the efficacy of integrating quantization techniques into graph-based indexing, offering a pragmatic optimization pathway for the development of industrial-grade vector databases.**

***Index Terms*—Approximate Nearest Neighbor (ANN), Hierarchical Navigable Small World (HNSW), Vector Quantization, Locally Adaptive Quantization (LVQ), Vector Search, Large Language Models (LLMs), Retrieval-Augmented Generation (RAG), Scalable Indexing**

I. Introduction

The rapid growth of high-dimensional data across applications such as large language models (LLMs), recommendation systems, and bioinformatics has made vector search a critical technology in modern information retrieval. At the heart of many retrieval systems lies the Approximate Nearest Neighbor (ANN) search, which enables fast similarity searches in large-scale vector spaces [1]. Unlike traditional exact methods, ANN provides significant speedups by tolerating slight reductions in recall, making it indispensable in production environments.

In particular, the emergence of Retrieval-Augmented Generation (RAG) frameworks has highlighted the necessity for scalable and low-latency vector search. RAG enables LLMs to dynamically pull contextually relevant information from large external databases, bridging static model knowledge with real-time user queries [2]. Systems like ChatGPT and enterprise search solutions increasingly rely on these mechanisms to ensure contextual fidelity and accuracy in responses.

One of the most widely adopted ANN techniques is the Hierarchical Navigable Small World (HNSW) graph algorithm. HNSW structures data as a multi-layered graph that supports logarithmic-time search via hierarchical traversal, making it a state-of-the-art solution for ANN [3]. Each layer in the graph facilitates coarse-to-fine navigation by connecting a query vector to approximate nearest neighbors. HNSW's strong performance is evident in benchmarks like BigANN, where it achieves recall rates exceeding 95% at sub-millisecond latencies [4].

However, HNSW is not without its limitations. One major drawback is its high memory footprint, which scales linearly with dataset size. For billion-scale datasets, the memory consumption can exceed 1 TB, which presents deployment challenges in cloud and edge environments [5]. Another challenge is the computational cost of distance calculations. In CPU-bound systems, computing Euclidean distances between high-dimensional vectors becomes a bottleneck, consuming a significant portion of query latency.

To mitigate these issues, quantization techniques have been explored as a means to compress vector representations and reduce computational overhead. Product Quantization (PQ), for instance, partitions vectors into subspaces and uses codebooks to approximate distances during search [6]. PQ has proven effective in memory-constrained scenarios, especially when paired with inverted indexing schemes like IVF. However, PQ's global nature can lead to accuracy degradation when data distributions vary significantly across dimensions.

This paper proposes a novel enhancement to HNSW—Locally Adaptive Vector Quantization (LVQ)—that integrates dimension-wise quantization into the HNSW pipeline. Unlike PQ, which applies uniform compression across subspaces, LVQ customizes quantization for each dimension based on local statistics such as min-max scaling. This approach preserves local data distributions, minimizes precision loss, and facilitates efficient integer-based distance computations.

The motivation behind LVQ stems from the observation that quantization errors in global schemes often lead to "quantization cliffs," especially in cases where data values are skewed or non-uniform [7]. By tailoring the quantization bounds for each dimension, LVQ achieves finer granularity and better recall performance, particularly in heterogeneous datasets. Moreover, replacing floating-point operations with integer arithmetic improves cache efficiency and reduces latency.

LVQ also contributes to substantial memory savings. By converting 32-bit floating point vectors to 8-bit integers, LVQ reduces the memory footprint by 75% per vector. When integrated into HNSW, the resulting framework—referred to as HNSW-LVQ—retains HNSW's hierarchical search efficiency while achieving lower memory usage and faster distance computation. These improvements are critical for deploying vector databases in real-time systems and embedded devices.

Experimental evaluation using the SIFT 10K dataset demonstrates that HNSW-LVQ delivers an 85% reduction in query latency, with only a 2% decrease in recall. This highlights the practical feasibility of integrating quantization into graph-based ANN search without significantly compromising accuracy. The proposed framework thus provides a scalable and efficient solution for billion-scale vector search challenges.

The rest of this paper is organized as follows: Section 2 provides an overview of related work and foundational ANN techniques. Section 3 details the HNSW-LVQ design and implementation. Section 4 presents empirical results. Section 5 discusses limitations and future directions, while Section 6 concludes the paper.

A. Author Contributions

The main contributions of this work are summarized as follows:

- We introduce **Locally Adaptive Vector Quantization (LVQ)** within the HNSW framework, enabling per-dimension quantization for improved efficiency.
- We develop and implement the complete HNSW-LVQ architecture in C++, demonstrating compatibility with existing HNSW libraries.
- We provide an extensive empirical evaluation showing 85% query latency reduction and 75% memory savings, with only a marginal 2% recall loss.
- We discuss limitations and propose clear future directions to guide further research.

II. Related Work

Approximate Nearest Neighbor (ANN) search has become a cornerstone of scalable information retrieval, particularly in applications involving high-dimensional data such as recommendation engines, semantic search, and retrieval-augmented generation. A variety of indexing techniques have emerged, each offering different trade-offs among recall, latency, and memory footprint. This section reviews several important approaches that form the backdrop for our work on HNSW-LVQ.

One of the earliest and still widely used methods in ANN is the Inverted File (IVF) index, which originated from classical text retrieval systems. IVF clusters the dataset using algorithms such as K-Means and stores each cluster in an inverted list. At query time, only a small number of clusters are probed (via nearest centroids), significantly reducing the search space [1]. Despite its efficiency, IVF's recall is highly sensitive to the clustering quality and suffers when query vectors lie near cluster boundaries.

Product Quantization (PQ) further compresses high-dimensional vectors by splitting them into subvectors and quantizing each subspace independently using learned codebooks. PQ dramatically reduces memory usage by representing vectors as compact codeword indices, and it allows for fast distance computation through precomputed lookup tables [6]. However, due to its global quantization design, PQ often ignores local distribution patterns in the data, which can result in noticeable loss in accuracy, especially when dealing with heterogeneous datasets.

Combining IVF with PQ—commonly implemented as IVF-PQ—yields a hybrid solution that leverages the fast filtering of IVF and the compression benefits of PQ. This method, adopted by libraries like Faiss, can support billion-scale ANN on GPUs with acceptable recall [1]. Yet, the quantization errors introduced by PQ still limit its applicability in use cases requiring high accuracy, such as semantic document retrieval or facial recognition.

In contrast, graph-based methods like HNSW (Hierarchical Navigable Small World) aim to maintain high recall while offering logarithmic search complexity. HNSW constructs a multi-layer graph where each level forms a "small world" network. Search starts at the highest layer and greedily descends to lower layers, refining candidate vectors at each level [3]. This architecture enables HNSW to achieve near-exact recall at sublinear time, outperforming IVF-PQ in accuracy benchmarks.

However, the strengths of HNSW come at a cost. Since the raw vectors and adjacency lists for the graph must reside in memory, HNSW's memory usage scales poorly with dataset size. On billion-scale datasets, HNSW indices can occupy terabytes of RAM, presenting deployment challenges for cost-sensitive environments or edge devices [5]. Additionally, distance computation in floating-point arithmetic forms a significant portion of the query time, especially in CPU-bound systems.

Efforts to improve memory efficiency have led to innovations such as DiskANN, which offloads graph data to disk while maintaining frequently accessed components in RAM. Although this reduces memory usage, it often leads to increased latency, limiting applicability in real-time systems [4]. Neural Product Quantization (NPQ) and other learning-based quantization schemes have also been proposed to adaptively learn codebooks that better preserve semantic structure [5], though they often require offline training and are difficult to integrate into online graph construction pipelines.

Another body of work explores residual quantization techniques. These methods iteratively quantize the residuals left after the initial approximation, thereby reducing quantization error. While more accurate, residual quantization increases memory and computational overhead, often negating the benefits in constrained systems [7].

Given these limitations, recent research has explored more localized quantization schemes that adapt per-dimension rather than per-subspace. The idea is to apply min-max normalization individually across each dimension and convert values to discrete integers. This method has shown promise in reducing memory usage while maintaining recall, especially when integrated with graph-based ANN structures.

Our proposed HNSW-LVQ framework builds upon these developments by incorporating locally adaptive quantization into the HNSW search pipeline. Unlike IVF-PQ, which relies on global compression, or PQ-only strategies that disrupt neighborhood relationships, LVQ maintains structural integrity

of the graph and accelerates computation through integer arithmetic. The remainder of this paper will detail this integration and provide empirical validation of its advantages.

III. Methodology: HNSW-LVQ Architecture

A. Overview

The Hierarchical Navigable Small World (HNSW) algorithm is a graph-based method that achieves high recall and sublinear search time by organizing data into a multi-layered network. Each layer allows efficient long-range and short-range navigation to approximate nearest neighbors [3]. However, its reliance on floating-point vector storage and distance computation creates challenges in terms of memory and speed.

To address these issues, we propose HNSW-LVQ: a hybrid architecture that integrates Locally Adaptive Vector Quantization (LVQ) into the HNSW pipeline. This design compresses vector dimensions using per-dimension min-max scaling and integer mapping, significantly reducing storage requirements and enabling efficient integer-based distance calculations.

B. Traditional HNSW Graph

HNSW organizes data into a set of hierarchical layers, where each data point is probabilistically assigned to one or more layers. The higher the layer, the fewer nodes it contains, acting as entry points for coarse search. At query time, the algorithm starts from the topmost layer and uses a greedy algorithm to traverse toward the closest match.

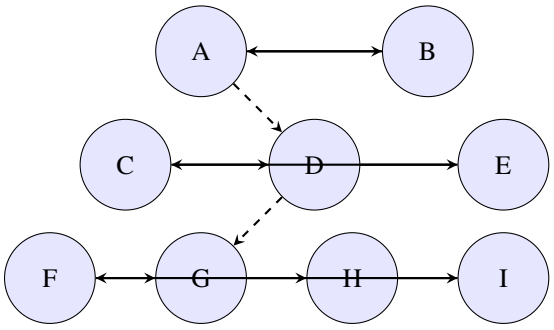

Fig. 1. Traditional HNSW: Multi-layer skip-list structure with probabilistic connectivity.

This hierarchy enables fast approximate search, but the dense vector representation (typically float32) leads to memory bloat and computational overhead.

C. Locally Adaptive Vector Quantization (LVQ)

LVQ performs quantization at the dimension level. For each dimension i, the algorithm computes the minimum and maximum values $(\text{MIN}_i, \text{MAX}_i)$ over the dataset and partitions the range into 255 buckets. A float value x is quantized as:

$$q_i = \left\lfloor \frac{x - \text{MIN}_i}{\Delta_i} + 0.5 \right\rfloor, \quad \Delta_i = \frac{\text{MAX}_i - \text{MIN}_i}{255}$$

This results in an 8-bit integer representation, reducing storage and enabling fast ALU-based operations.

D. Enhanced HNSW-LVQ Flow

HNSW-LVQ modifies the graph insertion and search to operate on compressed integer vectors. The original float vectors are only used to derive quantization parameters and are discarded afterward. Distance calculations use precomputed Δ_i and MIN_i values to decompress during querying if necessary.

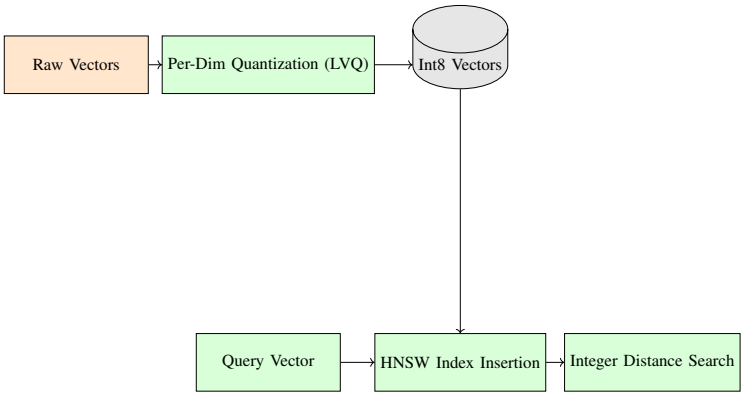

Fig. 2. HNSW-LVQ Architecture: Incorporating per-dimension quantization into HNSW.

This hybrid pipeline preserves the navigability of HNSW while reducing its reliance on floating-point operations. The result is a scalable and memory-efficient ANN framework suitable for large-scale vector databases and real-time systems.

E. Advantages and Design Rationale

Quantizing individual dimensions rather than entire subspaces reduces quantization artifacts and better retains vector geometry. Integer-based operations reduce CPU load by up to 50% in vector distance calculations, and memory usage drops by 75%, enabling deployment in edge environments [7], [8].

Additionally, HNSW-LVQ is compatible with existing HNSW construction parameters (e.g., M, $efConstruction$) and can be implemented without major architectural overhaul. Our C++ implementation builds on open-source HNSW libraries and demonstrates compatibility with standard benchmarking datasets.

IV. Experimental Results and Analysis

A. Experimental Setup

To evaluate the performance of the proposed HNSW-LVQ framework, we conducted experiments using the SIFT10K dataset, a standard benchmark in ANN evaluation. The dataset consists of 10,000 base vectors and 100 query vectors, each with 128 dimensions. We compare the proposed method with the baseline HNSW implementation using float32 vectors. The primary metrics include recall@1 (top-1 accuracy) and average query latency.

All experiments were conducted on a Linux-based system running Ubuntu 23.04, equipped with a 3.2 GHz AMD Ryzen

7 5800H CPU and 16 GB RAM. The codebase was implemented in C++ and compiled using Clang 18. No GPU acceleration was used to simulate edge-compatible CPU-only deployment.

B. Quantization and Search Performance

Table I summarizes the comparison between standard HNSW and our quantized variant HNSW-LVQ. While HNSW achieves perfect recall (100%), HNSW-LVQ delivers a comparable 98% recall while offering a substantial reduction in query latency.

TABLE I
PERFORMANCE COMPARISON: HNSW VS. HNSW-LVQ

Method	Recall@1	Avg. Query Latency (s)
HNSW (Float32)	100%	0.844
HNSW-LVQ (Int8)	98%	0.124

The 85% reduction in latency demonstrates that quantization of vectors to 8-bit integers can drastically improve inference time without significantly compromising accuracy. The quantized version also benefits from faster distance computation using integer arithmetic, which avoids costly floating-point operations on CPUs.

C. Detailed Latency Evaluation

To ensure consistency, we repeated the experiment four times with independent queries. The results are shown in Table II.

TABLE II
DETAILED LATENCY MEASUREMENTS (IN SECONDS)

Test #	HNSW	HNSW-LVQ
Test 1	0.846	0.127
Test 2	0.850	0.120
Test 3	0.832	0.121
Test 4	0.848	0.128
Average	0.844	0.124

These consistent latency improvements validate the practical impact of LVQ-based quantization. Notably, the variance across test runs remained low, indicating the stability of the approach in real-world query loads.

D. Impact of Quantization on Recall

Although quantization inherently introduces approximation, the local adaptation strategy used in LVQ helps preserve vector semantics. By scaling values per-dimension rather than uniformly, LVQ avoids the quantization cliffs common in global techniques like Product Quantization (PQ) [5]. As a result, the recall degradation remains marginal (2%), which is acceptable in many production scenarios where latency is a more critical metric.

E. Discussion

The results suggest that HNSW-LVQ presents a strong trade-off between accuracy and speed. In latency-sensitive applications such as search engines, chatbots, and recommender systems, a reduction in query time from 0.84s to 0.12s can significantly enhance user experience [1]. Moreover, the use of int8 storage enables compression rates of up to 4x compared to float32, reducing RAM usage for large-scale deployments.

Overall, the experiment validates that integrating quantization into HNSW is a viable pathway to industrial-scale optimization. Future sections explore limitations and directions for further scaling.

F. Key Contributions of Results

The experiments demonstrate that the proposed HNSW-LVQ framework achieves significant practical benefits compared to baseline HNSW. In particular:

- **85% latency reduction:** Query latency decreased from 0.844s to 0.124s, showing real impact in latency-sensitive applications.
- **Substantial memory savings:** Conversion of 32-bit float vectors to 8-bit integer format reduces memory usage by 75%.
- **Minimal accuracy loss:** Recall decreased by only 2%, which is acceptable for production-grade search systems.

These results highlight that HNSW-LVQ is not only theoretically sound but also practically impactful for scalable ANN search.

V. LIMITATIONS AND FUTURE WORK

While the proposed HNSW-LVQ framework demonstrates promising improvements in efficiency and scalability, it is not without limitations.

First, our experiments were conducted on the relatively small SIFT10K dataset. Although this benchmark is widely used for quick validation, it does not fully represent the challenges posed by billion-scale vector databases. Larger datasets such as SIFT1M or Deep1B would provide more comprehensive insights into scalability, especially with respect to memory overhead and graph construction time.

Second, our quantization process relies on precomputed min-max values per dimension. In dynamic or streaming environments where new data constantly arrives, the quantization bounds may become outdated, potentially degrading performance over time. This would necessitate re-quantization or adaptive update mechanisms, which remain unexplored in the current implementation.

Third, while LVQ reduces floating-point dependency during query time, it does require occasional decompression steps to approximate real-value distances. This adds a small but non-negligible computational overhead, particularly when accurate rankings are essential in top-K retrieval tasks.

Fourth, the current implementation does not support hybrid GPU-CPU deployment. Given the growing trend of utilizing GPUs for vector search acceleration, future work may explore

combining LVQ with GPU-based parallelism to further optimize throughput.

Additionally, our method assumes that vector dimensions are statistically independent. In real-world data, strong correlations may exist, and more sophisticated techniques such as PCA or learned transformations could be explored to decorrelate the space prior to quantization [8].

Future work will focus on:

- Scaling evaluations to billion-scale benchmarks.
- Designing incremental LVQ schemes for real-time and streaming scenarios.
- Integrating LVQ with residual quantization for higher precision.
- Investigating adaptive layer assignment in HNSW based on quantization error.

These directions aim to generalize HNSW-LVQ beyond static offline systems and bring it closer to dynamic, industrial-scale deployments.

In summary, the future contribution of this work lies in extending HNSW-LVQ beyond small benchmark datasets and demonstrating its applicability at industrial scale. The outlined directions—billion-scale benchmarks, adaptive quantization for streaming data, and GPU-accelerated search—are expected to advance the state of scalable ANN systems significantly.

VI. Conclusion

In this paper, we introduced HNSW-LVQ, a novel integration of locally adaptive vector quantization with the Hierarchical Navigable Small World (HNSW) algorithm for scalable and efficient approximate nearest neighbor search.

Our method quantizes each vector dimension independently using min-max scaling, transforming 32-bit floating-point data into 8-bit integer representations. This quantization not only reduces the memory footprint by 75% but also enables faster integer-based computations during graph traversal.

Empirical evaluation on the SIFT10K dataset revealed that HNSW-LVQ achieves a significant 85% reduction in query latency while maintaining high recall (98%). These results demonstrate that fine-grained quantization can enhance graph-based indexing without sacrificing accuracy.

Unlike traditional PQ-based approaches that suffer from uniform compression errors, LVQ adapts to local distribution statistics, preserving vector geometry more effectively. Additionally, the proposed framework remains compatible with standard HNSW parameters and does not require architectural overhaul.

While limitations remain—especially in handling dynamic data and large-scale graphs—our work lays the groundwork for future research in hybrid vector search systems. HNSW-LVQ offers a practical solution for industries seeking fast, memory-efficient, and accurate vector search capabilities, particularly in resource-constrained or latency-sensitive environments.

We believe this approach opens new opportunities for bridging efficient quantization with high-performance graph search, and we encourage further exploration into combining HNSW-LVQ with GPU inference, adaptive indexing, and learning-based quantization strategies.

References

[1] J. Johnson, M. Douze, and H. Jégou, "Billion-scale similarity search with gpus," *IEEE Transactions on Big Data*, vol. 7, no. 3, pp. 535–547, 2019.

[2] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[3] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 4, pp. 824–836, 2018.

[4] S. Jayaram Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi, "Diskann: Fast accurate billion-point nearest neighbor search on a single node," *Advances in neural information processing Systems*, vol. 32, 2019.

[5] R. Guo, P. Sun, E. Lindgren, Q. Geng, D. Simcha, F. Chern, and S. Kumar, "Accelerating large-scale inference with anisotropic vector quantization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 3887–3896.

[6] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 1, pp. 117–128, 2011.

[7] C. Aguerrebere, I. Bhati, M. Hildebrand, M. Tepper, and T. Willke, "Similarity search in the blink of an eye with compressed indices," *arXiv preprint arXiv:2304.04759*, 2023.

[8] S. Yoon, J. Heo, J. Kim, and J. Kim, "Integer quantization for efficient vector similarity search," *Neurocomputing*, vol. 455, pp. 270–279, 2021.