

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH  
TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



HỌC PHẦN MỞ RỘNG  
Cấu trúc dữ liệu và giải thuật

Tăng tốc tìm kiếm gần đúng bằng thuật toán ScaNN

Lớp: TN01

Nhóm: 1

Học kì: 251

STT	Họ và tên	MSSV
1	Phan Phước Thiện Quang	2412843
2	Trần Văn Thiên Kim	2411816
3	Lê Đức Nguyên Khoa	2411603

TP. Hồ Chí Minh, tháng 11/2025



## Mục lục

<b>1</b>	<b>Giới thiệu chung</b>	<b>1</b>
1.1	Bối cảnh và Thách thức của Vector Search . . . . .	1
1.2	Giới thiệu thuật toán ScaNN . . . . .	1
<b>2</b>	<b>Mục tiêu và Phạm vi thực hiện</b>	<b>1</b>
2.1	Mục tiêu nghiên cứu . . . . .	1
2.2	Danh mục công việc đã thực hiện . . . . .	2
<b>3</b>	<b>Tổng quan lý thuyết chuyên sâu về ScaNN</b>	<b>2</b>
3.1	Partitioning (Phân vùng không gian) . . . . .	2
3.2	Scoring (Chấm điểm và Tăng tốc) . . . . .	3
3.2.1	Asymmetric Hashing (AH) . . . . .	3
3.2.2	Anisotropic Vector Quantization (AVQ) . . . . .	4
3.3	Reordering (Sắp xếp lại) . . . . .	5
<b>4</b>	<b>Kết quả thực nghiệm và Thảo luận</b>	<b>5</b>
4.1	Kết quả thực nghiệm . . . . .	5
4.1.1	Độ chính xác (Recall@K) . . . . .	5
4.1.2	Tốc độ truy vấn (Latency) . . . . .	6
4.2	Tổng kết thực nghiệm . . . . .	7
<b>5</b>	<b>Kết luận và Hướng phát triển</b>	<b>7</b>
5.1	Kết luận . . . . .	7
5.2	Hướng phát triển (Future Work) . . . . .	8
	<b>TÀI LIỆU THAM KHẢO</b>	<b>9</b>



## 1 Giới thiệu chung

### 1.1 Bối cảnh và Thách thức của Vector Search

Trong kỷ nguyên số hóa hiện nay, việc xử lý dữ liệu phi cấu trúc (văn bản, âm thanh, hình ảnh, video) đang trở thành một yêu cầu cốt lõi. Phương pháp biểu diễn ngữ nghĩa thông qua Vector Embedding (nhúng vector) đã mở ra hướng đi mới, cho phép máy tính "hiểu" được sự tương đồng về mặt ý nghĩa giữa các dữ liệu thay vì chỉ so sánh từ khóa đơn thuần.

Tuy nhiên, khi số chiều dữ liệu tăng cao (hàng trăm đến hàng nghìn chiều) và số lượng bản ghi đạt ngưỡng hàng triệu, bài toán tìm kiếm láng giềng gần nhất (Nearest Neighbor Search) gặp phải thách thức lớn về hiệu năng, thường được gọi là "lời nguyền của số chiều" (Curse of Dimensionality).

Hiện nay, thuật toán HNSW (Hierarchical Navigable Small World) dựa trên đồ thị được xem là tiêu chuẩn phổ biến nhất (State-of-the-Art). Tuy nhiên, HNSW bộc lộ những hạn chế đáng kể khi vận hành ở quy mô lớn:

- **Thời gian Indexing:** Quá trình xây dựng chỉ mục tốn nhiều thời gian và tài nguyên tính toán.
- **Khả năng cập nhật:** Khó khăn trong việc cập nhật dữ liệu theo thời gian thực (real-time updates), thường yêu cầu xây dựng lại chỉ mục.
- **Tài nguyên bộ nhớ:** Tiêu tốn lượng lớn RAM để lưu trữ cấu trúc đồ thị và các liên kết, gây tốn kém chi phí hạ tầng.

### 1.2 Giới thiệu thuật toán ScaNN

ScaNN (Scalable Nearest Neighbors), phát triển bởi Google Research, là giải pháp được thiết kế để giải quyết các thách thức trên bằng cách tiếp cận dựa trên lượng tử hóa (quantization-based).

Sự hiệu quả của ScaNN đã được kiểm chứng thực tế. Khi tích hợp vào AlloyDB (cơ sở dữ liệu tương thích pgvector), so với HNSW trên PostgreSQL tiêu chuẩn (số liệu tháng 4/2024), ScaNN đạt được:

- Tốc độ tạo chỉ mục nhanh hơn tới **8 lần**.
- Tốc độ truy vấn nhanh hơn tới **4 lần**.
- Sử dụng bộ nhớ ít hơn **3 – 4 lần**.
- Thông lượng ghi (write throughput) cao hơn tới **10 lần**.

## 2 Mục tiêu và Phạm vi thực hiện

### 2.1 Mục tiêu nghiên cứu

Nhóm nghiên cứu đặt ra 5 mục tiêu cốt lõi nhằm làm chủ công nghệ này:

1. **Nắm vững lý thuyết:** Hiểu rõ kiến trúc của thuật toán ScaNN, đặc biệt là cơ chế lượng tử hóa dị hướng (AVQ).
2. **So sánh và Đối chiếu:** Phân biệt được các điểm khác biệt cốt lõi, ưu nhược điểm giữa ScaNN và các thuật toán ANN khác như HNSW hay IVF (Inverted File Index).



3. **Ứng dụng thực tiễn:** Nắm bắt cách thức tích hợp ScaNN vào các bài toán thực tế.
4. **Xây dựng Demo:** Phát triển hệ thống Search Engine đơn giản sử dụng ScaNN để tìm kiếm Top-K gần đúng cho dữ liệu văn bản.
5. **Đánh giá định lượng:** Thực hiện đo đặc hiệu năng (Latency) và độ chính xác (Recall) thông qua thực nghiệm nghiêm túc.

## 2.2 Danh mục công việc đã thực hiện

- **Nghiên cứu lý thuyết:** Tìm hiểu sâu về kiến trúc *Tree-quantization*, cơ chế AVQ và AH. Phân tích các tài liệu công bố của Google Research.
- **Cài đặt thuật toán:** Hiện thực hóa thuật toán ScaNN bằng ngôn ngữ Python, sử dụng thư viện ‘scann’ chính chủ trên nền tảng Google Colab để tận dụng tài nguyên tính toán.
- **Thực nghiệm:** Tiến hành benchmark trên bộ dữ liệu quy mô trung bình lớn gồm **500.000 phần tử**. Thiết lập kịch bản đo đặc các chỉ số Recall@K và Latency.
- **Ứng dụng Demo:** Xây dựng giao diện Search Engine cho phép nhập truy vấn văn bản, chuyển đổi sang vector embedding và truy xuất kết quả Top-K trong thời gian thực.

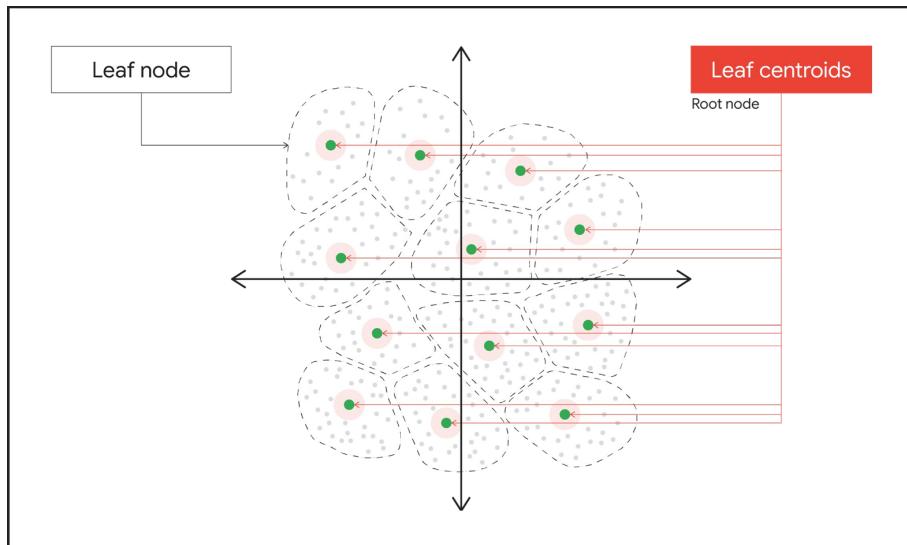
## 3 Tổng quan lý thuyết chuyên sâu về ScaNN

ScaNN thuộc họ các chỉ mục ANN dựa trên lượng tử hóa cây (tree-quantization based ANN indices). Quy trình hoạt động có thể được chia thành 3 giai đoạn liên kết chặt chẽ:

### 3.1 Partitioning (Phân vùng không gian)

Giai đoạn này nhằm mục đích "chia để trị", giảm thiểu không gian tìm kiếm từ toàn bộ cơ sở dữ liệu xuống một tập con nhỏ hơn.

- **Cấu trúc Cây:** ScaNN sử dụng cấu trúc cây (thường là 2 tầng). Tầng gốc chứa các tâm cụm (centroids), và tầng lá chứa các điểm dữ liệu thực tế.
- **Training K-Means:** Các centroid được xác định bằng thuật toán phân cụm spherical k-means. Quá trình này tối ưu hóa việc phân chia không gian sao cho các điểm dữ liệu trong cùng một vùng (partition) có sự tương đồng cao nhất với centroid của vùng đó.
- **Cơ chế Pruning:** Khi truy vấn, thay vì quét hết dữ liệu, ScaNN chỉ so sánh query vector với các centroid. Chỉ  $n$  vùng (leaves) có centroid gần nhất mới được chọn để tìm kiếm chi tiết (tham số `scann.num_leaves_to_search`).



Hình 1: Chỉ mục hai tầng cho vector 2D

### 3.2 Scoring (Chấm điểm và Tăng tốc)

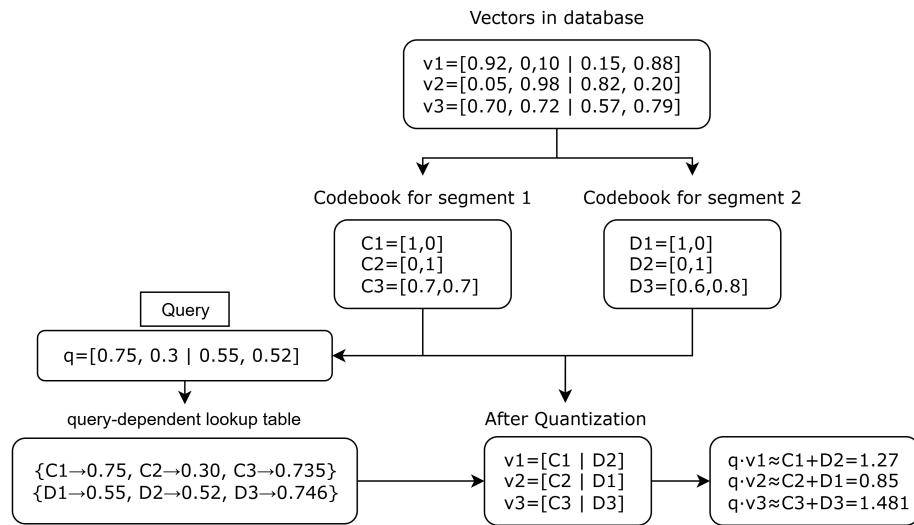
Quá trình truy vấn được khởi tạo bằng việc tính toán khoảng cách từ vector truy vấn đầu vào đến các trọng tâm (centroids) của các phân vùng dữ liệu. Dựa trên kết quả này, thuật toán thực hiện sàng lọc không gian tìm kiếm bằng cách chỉ lựa chọn một tập hợp con các phân vùng có trọng tâm lân cận nhất để truy xuất dữ liệu chi tiết. Các phân vùng nằm ngoài phạm vi này sẽ bị loại bỏ (pruning) hoàn toàn khỏi quá trình tìm kiếm nhằm giảm thiểu khối lượng tính toán. Số lượng phân vùng được chọn để duyệt được quy định bởi tham số cấu hình của hệ thống.

Tại giai đoạn tính điểm (Scoring), để tối ưu hóa tốc độ xử lý, ScaNN không thực hiện tính toán trực tiếp trên các vector gốc mà áp dụng các kỹ thuật nén và lượng tử hóa dữ liệu. Phương pháp này cho phép xấp xỉ hóa các phép tích tích vô hướng hoặc khoảng cách một cách hiệu quả, chuyển đổi các tác vụ tính toán ma trận phức tạp thành các phép toán số học cơ bản nhẹ nhàng hơn. Nhờ đó, hệ thống có khả năng quét qua không gian dữ liệu quy mô lớn và đưa ra các ứng viên tiềm năng với độ trễ cực thấp.

#### 3.2.1 Asymmetric Hashing (AH)

Kỹ thuật này giải quyết bài toán bộ nhớ và tốc độ tính toán:

- **Chia đoạn (Subspace Decomposition):** Mỗi vector dữ liệu  $v$  (kích thước  $D$ ) được chia thành  $M$  đoạn nhỏ.
- **Codebook:** Mỗi đoạn con được lượng tử hóa bằng cách gán nó cho một vector đại diện (codeword) gần nhất trong một bảng mã (Codebook).
- **Lookup Table (Bảng tra cứu):** Khi truy vấn, tích vô hướng giữa query vector  $q$  và dữ liệu  $v$  được xấp xỉ bằng tổng các tích vô hướng của các đoạn con. Các giá trị này được tính trước và lưu trong bảng tra cứu, biến phép nhân ma trận phức tạp thành phép cộng đơn giản, tăng tốc độ tính toán lên đáng kể.

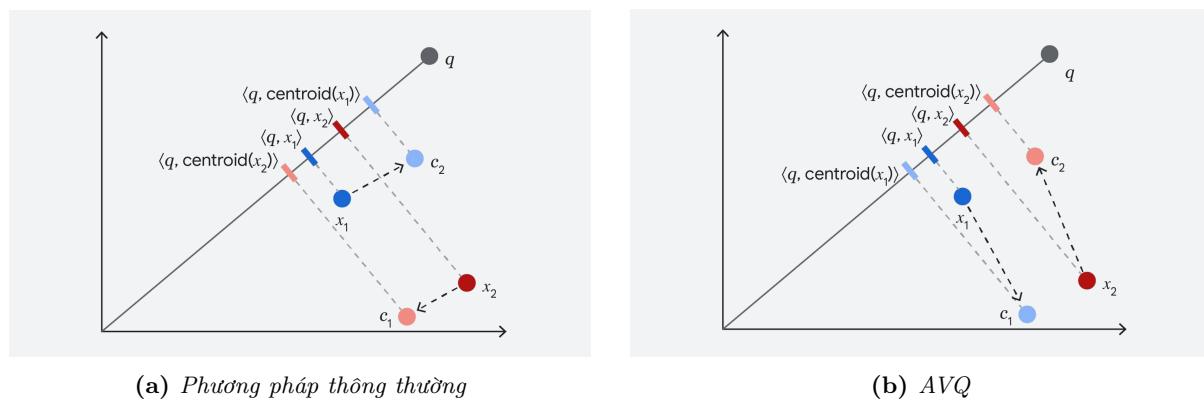


Hình 2: Ví dụ Asymmetric Hashing

### 3.2.2 Anisotropic Vector Quantization (AVQ)

Đây là công nghệ cốt lõi của ScaNN, giải quyết bài toán tối ưu hóa sai số trong không gian vector.

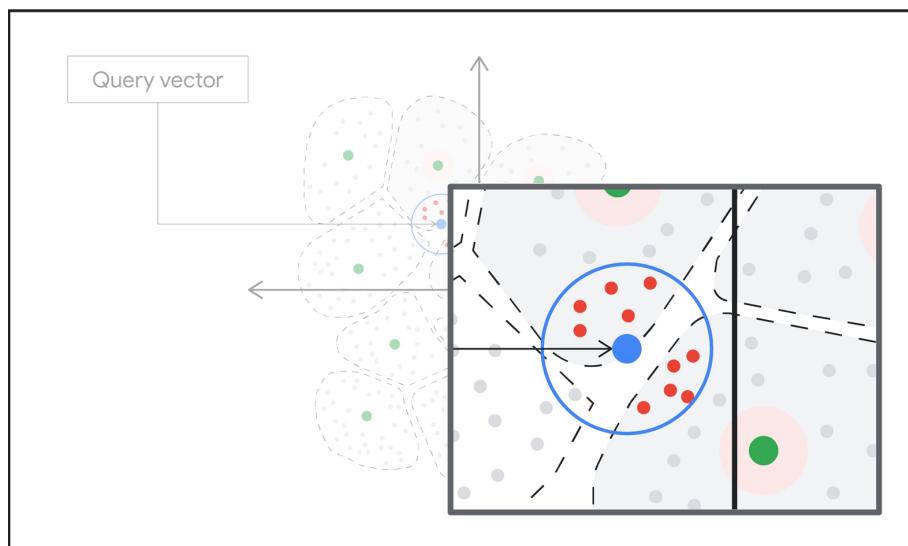
- Vấn đề của Lượng tử hóa đẳng hướng (Isotropic Quantization):** Các phương pháp cũ như Product Quantization nén sai số đồng đều trên mọi hướng. Tuy nhiên, tích vô hướng  $\langle q, x \rangle$  rất nhạy cảm với sai số song song với vector dữ liệu, trong khi sai số vuông góc ít ảnh hưởng hơn, dẫn đến việc phân bổ tài nguyên nén chưa tối ưu.
- Giải pháp Lượng tử hóa Dị hướng (AVQ):** AVQ khắc phục điều này bằng một hàm mất mát (loss function) mới, trừ phạt nặng các sai số nằm song song với hướng của vector gốc và chấp nhận sai số lớn hơn ở hướng vuông góc. Chiến lược này giúp tập trung độ chính xác vào chiều quan trọng nhất ảnh hưởng trực tiếp đến kết quả tích vô hướng.
- Hệ quả và Tác động:** Dù vector nén có thể lệch xa vector gốc về vị trí hình học (sai số Euclidean lớn), nhưng thứ hạng (ranking) khi so khớp bằng tích vô hướng lại được bảo toàn rất tốt. Sự đánh đổi này giúp ScaNN đạt độ chính xác (Recall) vượt trội so với các đối thủ cùng phân khúc nén dữ liệu.



### 3.3 Reordering (Sắp xếp lại)

Do giai đoạn Tính điểm (Scoring) sử dụng dữ liệu nén nên kết quả chỉ mang tính ước lượng. Giai đoạn Sắp xếp lại (Reordering) là bước tinh chỉnh cuối cùng để khôi phục độ chính xác, bao gồm ba bước xử lý nghiêm ngặt:

- **Over-fetching:** Thuật toán trích xuất một tập ứng viên sơ bộ Top-M lớn hơn yêu cầu ( $M > K$ , tham số `pmore_reorder_num_neighbors`) nhằm giảm thiểu rủi ro bỏ sót các kết quả tiềm năng do sai số trong quá trình nén.
- **Exact Computation (Tính toán):** Hệ thống thực hiện tính toán lại khoảng cách chính xác cho  $M$  ứng viên này bằng cách sử dụng vector gốc (full-precision) thay vì dữ liệu nén, giúp loại bỏ hoàn toàn các sai số lượng tử hóa trước đó.
- **Final Selection:** Danh sách ứng viên được tái sắp xếp dựa trên độ đo thực tế vừa tính toán, từ đó chọn ra Top-K kết quả tốt nhất và loại bỏ hiệu quả các trường hợp dương tính giả (false positives).



Hình 4: Ví dụ Reordering

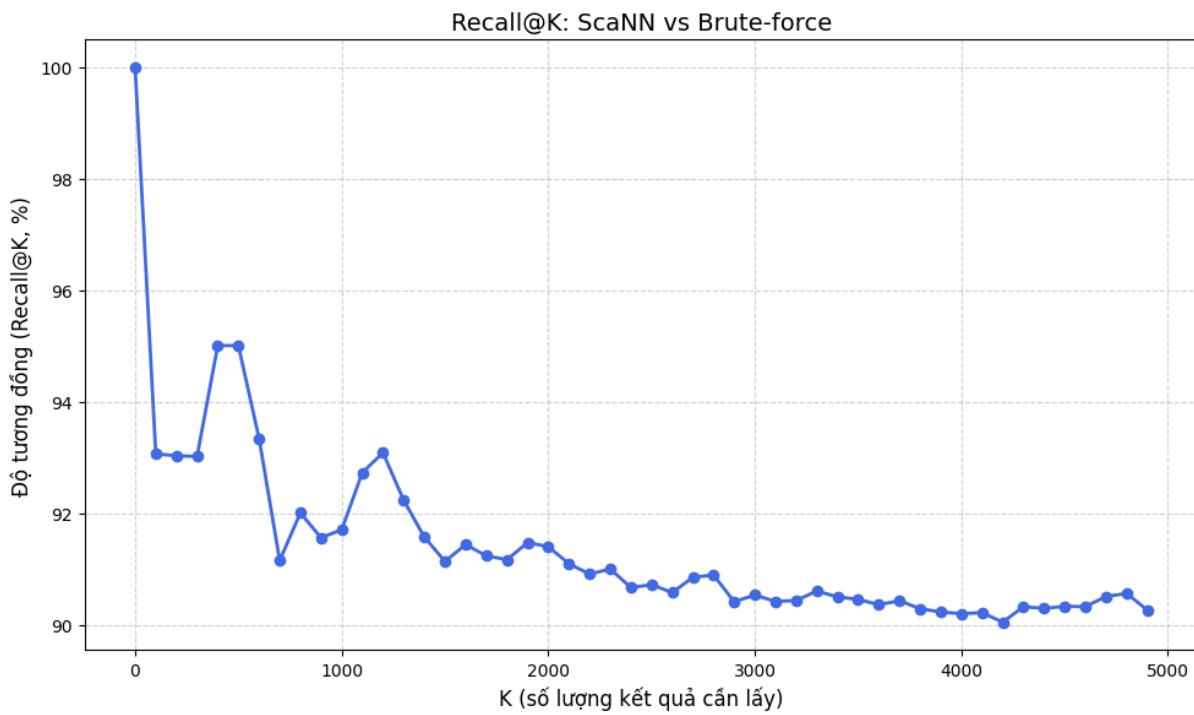
## 4 Kết quả thực nghiệm và Thảo luận

### 4.1 Kết quả thực nghiệm

Nhóm đã thực hiện benchmark so sánh SeaNN và Brute-force trên tập dữ liệu 100k - 500k vector.

#### 4.1.1 Độ chính xác (Recall@K)

Sau khi đối chiếu với kết quả của Brute-force ta thu được biểu đồ dưới đây:



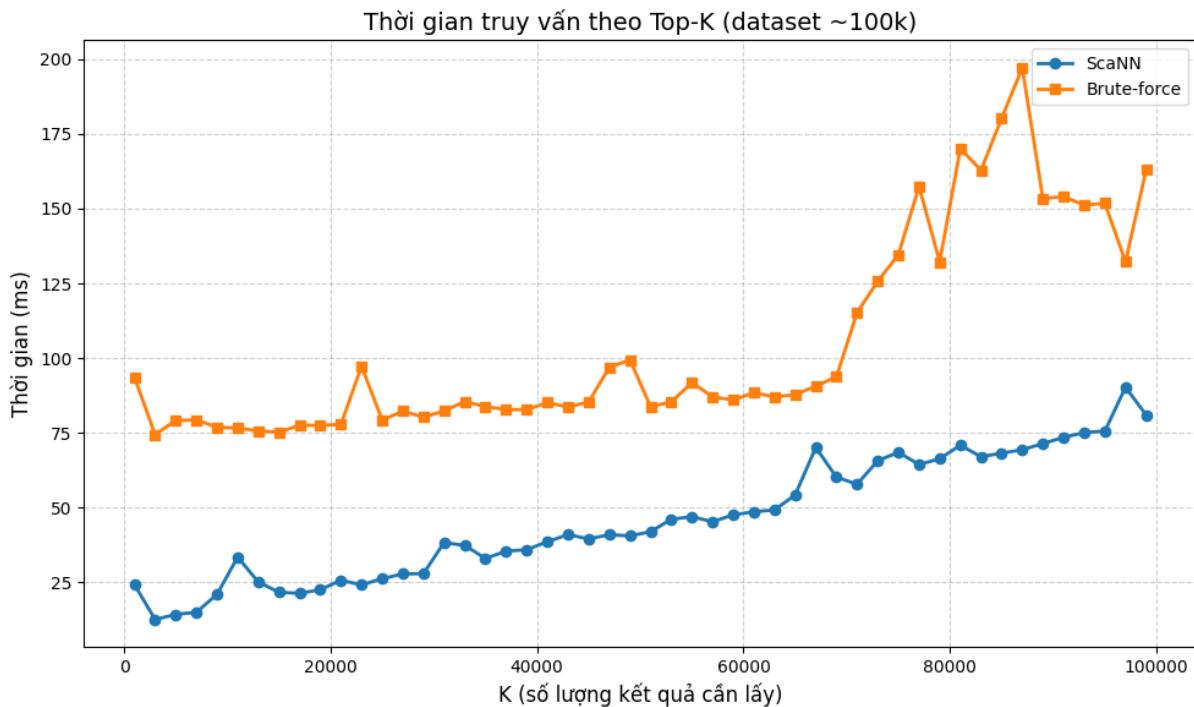
Hình 5: Biểu đồ Recall theo số lượng K

Nhận xét: Nhận xét:

- **High Recall:** ScaNN duy trì độ chính xác cao ( $> 90\%$ ) ngay cả khi số lượng kết quả cần lấy (K) tăng lên tới 5000.
- **Stability:** Mặc dù có sự sụt giảm nhẹ ở các giá trị K nhỏ đầu tiên (do bản chất xấp xỉ), thuật toán nhanh chóng ổn định ở mức recall  $\approx 91\%-92\%$ .
- **Kết luận:** ScaNN chấp nhận hy sinh một lượng nhỏ độ chính xác ( $< 10\%$ ) để đổi lấy tốc độ vượt trội.

#### 4.1.2 Tốc độ truy vấn (Latency)

So sánh tốc độ truy vấn của ScaNN và Brute-force, ta có biểu đồ dưới đây:



Hình 6: So sánh thời gian thực thi (ms)

Nhận xét:

- **Vượt trội:** Đường màu xanh (ScaNN) luôn nằm thấp hơn đáng kể so với Brute-force (màu cam).
- **Scalability:** Khi K tăng lớn ( $> 60,000$ ), Brute-force tăng vọt thời gian xử lý (đạt đỉnh 200ms) trong khi ScaNN tăng trưởng tuyến tính chậm và ổn định hơn nhiều.
- **Hiệu năng:** Trung bình ScaNN nhanh hơn Brute-force khoảng **2-3 lần** trên tập dữ liệu thử nghiệm (100k vectors).

## 4.2 Tổng kết thực nghiệm

**Sự đánh đổi hiệu năng (Performance Trade-off):** Kết quả thực nghiệm khẳng định quy luật đánh đổi: Giảm  $\approx 8\%$  độ chính xác  $\iff$  Tăng gấp 300% tốc độ.

- Đây là sự đánh đổi chấp nhận được trong các bài toán thực tế (Recommender System, Search Engine) nơi tốc độ phản hồi quan trọng hơn độ chính xác tuyệt đối 100
- Thuật toán chứng minh được khả năng mở rộng (Scalable) tốt khi kích thước tập K tăng lên.

## 5 Kết luận và Hướng phát triển

### 5.1 Kết luận

Báo cáo đã trình bày toàn diện về thuật toán ScaNN, từ lý thuyết cốt lõi (AVQ, AH) đến thực nghiệm. ScaNN chứng minh là một công cụ mạnh mẽ, cân bằng tốt giữa tài nguyên bộ nhớ, tốc độ truy vấn và độ chính xác, khắc phục được các nhược điểm của HNSW trên tập dữ liệu lớn.



## 5.2 Hướng phát triển (Future Work)

Dựa trên kết quả hiện tại, nhóm đề xuất các hướng nghiên cứu tiếp theo:

- Mở rộng tập dữ liệu:** Thử nghiệm trên các bộ dữ liệu chuẩn công nghiệp (như SIFT1M, GIST1M) với quy mô hàng triệu vector để đánh giá khả năng chịu tải cực đại.
- Tối ưu tham số tự động:** Nghiên cứu xây dựng cơ chế tự động tinh chỉnh (auto-tuning) các tham số num\_leaves và num\_leaves\_to\_search dựa trên phân phối dữ liệu cụ thể.
- Triển khai trên GPU:** Tìm hiểu khả năng tăng tốc phần cứng của ScaNN khi chạy trên GPU thay vì CPU hiện tại.



## TÀI LIỆU THAM KHẢO

### Tài liệu

- [1] Google Cloud. ScaNN for AlloyDB: Efficient vector similarity search. Whitepaper, Google, 2023.  
Truy cập ngày: 03-12-2025.
- [2] Google Research. ScaNN algorithms documentation. <https://github.com/google-research/google-research/blob/master/scann/docs/algorithms.md>,  
n.d. GitHub Repository. Truy cập ngày: 03-12-2025.