# Breaking the Single-Reference-Vector Barrier in Approximate Nearest Neighbor Search

Jiadong Xie
The Chinese University of Hong Kong
jdxie@se.cuhk.edu.hk

Jeffrey Liang
Australian National University
jeffrey.liang@anu.edu.au

Siyi Teng
The Chinese University of Hong Kong
syteng@se.cuhk.edu.hk

Jeffrey Xu Yu
The Hong Kong University of Science and Technology (Guangzhou)
jeffreyxuyu@hkust-gz.edu.cn

Yingfan Liu*
Xidian University
liuyingfan@xidian.edu.cn

## Abstract

Approximate nearest neighbor (ANN) searches are commonly employed in various machine learning applications, such as recommendation systems, but traditional ANN searches typically involve only a single reference vector in a query. To broaden the capabilities of ANN search and support multi-reference-vector queries, thereby enabling a wider range of machine learning applications, we introduce all/any-$k$ ANN search. They aim to find vectors that are similar to all or any of the multi-reference vectors in a query, respectively. To effectively and efficiently support all/any-$k$ ANN search, we first propose distance metrics to evaluate the ranking of vectors among those in the dataset for exact all/any-$k$ NN. Building on this, we introduce search algorithms and prove they can search according to the proposed distance metrics on graph indexes designed for traditional ANN. Additionally, we further introduce two-stage search algorithms for all/any-$k$ ANN search to further enhance their search performance. We conduct extensive experiments on real-world datasets to validate the efficiency and effectiveness of our proposed algorithms compared to existing approaches.

## CCS Concepts

• **Information systems → Information retrieval query processing**.

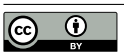
## Keywords

Multiple Vector Search, High-Dimensional Vector

**ACM Reference Format:**
Jiadong Xie, Jeffrey Liang, Siyi Teng, Jeffrey Xu Yu, and Yingfan Liu. 2026. Breaking the Single-Reference-Vector Barrier in Approximate Nearest Neighbor Search. In *Proceedings of the ACM Web Conference 2026 (WWW '26), April 13–17, 2026, Dubai, United Arab Emirates.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3774904.3792208

**Resource Availability:**
The source code of this paper has been made publicly available at https://github.com/Xiejiadong/Multi-Vector-Queries.

*Yingfan Liu is the corresponding author.




*WWW '26, Dubai, United Arab Emirates*

ACM ISBN 979-8-4007-2307-0/2026/04
https://doi.org/10.1145/3774904.3792208

## 1 Introduction

In the evolving landscape of machine learning models, different data types can be transformed into vector representations, such as text [31], images [33], audio [3], and graphs [17], enabling real-world applications to conduct queries within vector spaces. Thus, the ability to perform vector queries on high-dimensional vectors is essential for contemporary machine learning applications, like retrieval augmented generation [5, 21, 22], web search [6, 11, 28], recommendation systems [32, 37], and passage search [23]. Traditional vector queries involve a single-reference vector in each query, representing an item, and aim to identify similar items to it. That is, given a single-reference vector $q$, the query aims to find the $k$-approximate nearest neighbors ($k$-ANN) of $q$, i.e., the $k$-ANN are located within a small-radius ball centered at $q$.

In real-world machine learning applications [9, 19, 26, 38, 43, 53], multi-reference vectors are often required in a query to consider distances from target vectors to all reference vectors. For example, vectors of embeddings can be used in digital curation to identify overlaps in different styles of painting [26]. Specifically, curating a gallery with two painting styles requires identifying paintings that bridge the stylistic gap for a smoother transition by placing them between the two. Essentially, here, the goal is to find paintings that are similar to both given styles. Another scenario where finding objects similar to all query objects is when dealing with perturbed versions of an object as query vectors. It can locate the original object within the dataset by finding objects that are similar to all query objects [38]. Furthermore, there is another requirement in the applications: finding objects that are similar to any of the query objects. For instance, in a recommendation system, it supports increasing the items as a reference to avoid over-recommending homogeneous items [43]. Here, finding objects similar to any of the query objects can offer a wider array of choices during searches.

Therefore, to enhance $k$-ANN search beyond a single-reference vector, we propose multi-reference vector search in terms of finding objects similar to all/any of the query objects. Formally, given a query $q = [q_1, \cdots, q_m]$ with $m$ query vectors $q_i \in \mathbb{R}^d$ and a dataset $D \subset \mathbb{R}^d$, all-$k$ ANN search aims to find vectors in $D$ similar to all query vectors, i.e., within the intersection of balls with a small radius centered at the $m$ query vectors. Conversely, any-$k$ ANN searches aim to find vectors in $D$ similar to any of the given query vectors, i.e., within the union of balls with a small radius centered at the $m$ query vectors. To illustrate the concepts more vividly, as depicted in Fig. 1, we conduct case studies using Recipe dataset [30, 36], in which

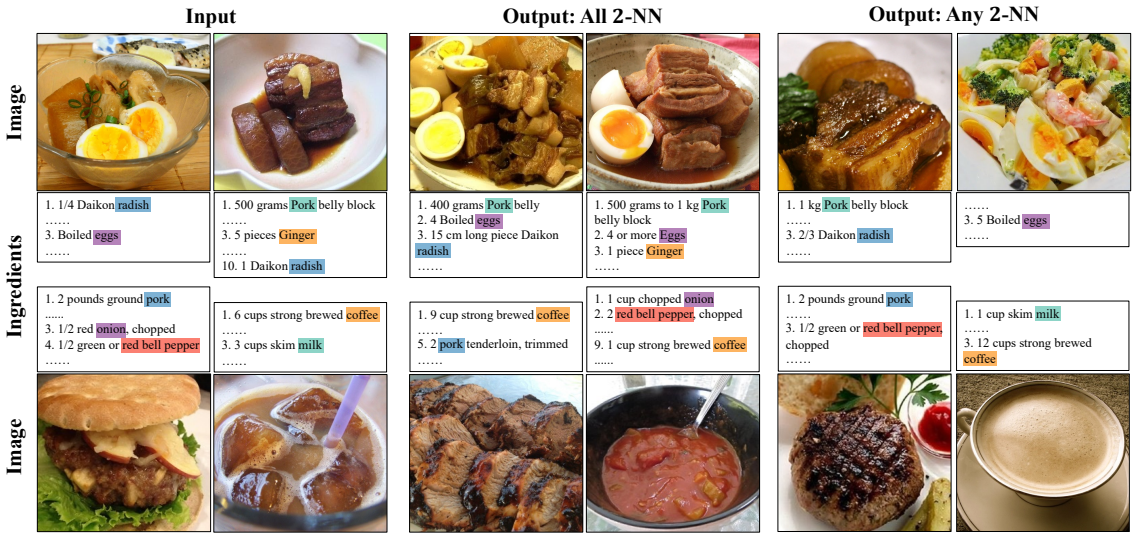
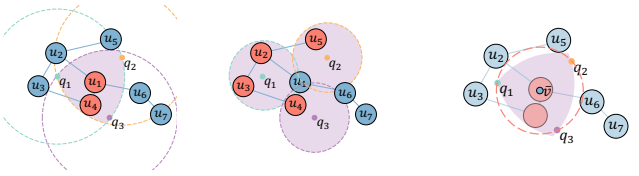



**Figure 1: Case Studies of All/Any-2 NN on Recipe Dataset**

vectors are embeddings generated from an encoder for each recipe modality (ingredients, instructions, and images). Two recipes from the dataset serve as references, with the objective being to identify recipes in the dataset that are similar to all or any of the reference vectors. In the first example, both input recipes are dishes featuring eggs and pork. The all-2 NN results comprise most ingredients from both recipes, whereas the any-2 NN results resemble at least one of the input recipes, and all retrieval results are similar types of dish. In the second case, we input two vastly different recipes: a hamburger and a coffee. The all-2 NN retrieved from the dataset seems to differ from the input recipes, a barbecue and a soup, but ingredients reveal overlaps with both inputs. Conversely, the any-2 NN results are close to one of the input recipes. While the input query vectors may appear distant from each other, the all/any-$k$ ANN queries have the potential to reveal relationships between the vectors in the input.

In the literature, numerous approaches have studied ANN search over multi-attribute data by representing each attribute of an item as a separate vector and then amalgamating these vectors into a single reference vector for search [8, 19, 41, 46, 47, 51–54]. To achieve high efficiency without constructing additional indexes for search, these approaches typically avoid performing ANN search directly on the entire combined vector of a query. Instead, they partition the query's reference vector back into its attribute-specific components, conduct individual $k'$-ANN searches for each component, and finally merge the partial results to obtain the overall $k$-ANN. Inspired by these approaches, we aim to adapt their methods to our problem. Specifically, we initially acquire the $k'$-ANN $\mathcal{R}_i$ for every vector $q_i \in q$. Next, we consider selecting the top-$k$ vectors from $\bigcup_{i=1}^{m} \mathcal{R}_i$ as the final all/any-$k$ ANN. However, a notable issue of this method lies in determining the optimal value of $k'$ for each query vector to find all/any-$k$ ANN. For all-$k$ ANN search, setting a small value of $k'$ can lead to suboptimal final results. This is because all-1 NN may not be proximate to any of the query vectors, e.g., it may reside in a region at the centroid of the query vectors. Conversely, setting a large value of $k'$ to each query vector will compromise search efficiency. For any-$k$ ANN search, the value of $k'$ required for each vector is typically smaller than or equal to $k$ and varies in different query vectors. But without prior knowledge, it is difficult to set the optimal $k'$ for each vector to obtain the final any-$k$ ANN search, and the suboptimal choices of $k'$ can hamper search efficiency by traversing the examination of unnecessary vectors in any-$k$ ANN search. In our problem of all/any-$k$ ANN search, the vectors in the query are located in the same vector representation space. This enables us to consider introducing a search algorithm where the search is concurrently guided under all query vectors to identify the all/any-$k$ ANN results, rather than conducting $k'$-ANN searches separately on each vector and then merging the results.

The main contributions of this work are summarized below. ❶ We introduce two distance metrics for evaluating the ranking of each vector within the dataset for both all and any-$k$ NN. Next, we prove that these two distance metrics are applicable to be used in the search within the current graph index designed for $k$-ANN search. This leads us to propose search algorithms for all and any-$k$ ANN, where the search can be simultaneously guided by all query vectors using the proposed distance metrics. ❷ We further decouple the two steps in our proposed search algorithm, i.e., the algorithm first approaches the target region and then gradually moves away from it to complete the search process. For all-$k$ ANN search, we present a novel approach involving first calculating a vector, then using it as the query vector for 1-ANN search to reach the target region. This can reduce the time required for the first step by reducing the distance computation time. For any-$k$ ANN search, the target region may not be continuous in space. Hence, we propose to perform a 1-ANN search of each query vector in the first step to identify all target regions for enhancing the effectiveness. While the second step of our algorithms remains unchanged, it starts the search using the vectors obtained from the first step. ❸ We conduct extensive experiments using datasets of real-world applications, which validate that our approaches are effective and efficient over existing approaches.



(a) All-$k$ NN (b) Any-$k$ NN

**Figure 2: An Example of All/Any-$k$ NN**



**Figure 3: Illustration of MEB**

## 2 Preliminaries

Let $D \subset \mathbb{R}^d$ be a high-dimensional dataset consisting of $n$ $d$-dimensional vectors. We denote the L2 norm (i.e., Euclidean distance) between two vectors $u, v \in \mathbb{R}^d$ as $\delta(u, v)$. For a given query vector $q$, the **$k$-nearest neighbor ($k$-NN)** search aims to find the $k$ vectors in $D$ with the minimum distances from $q$. Alternatively, we can define $k$-NN search by considering a ball centered at the query vector. That is, $k$-NN search obtains the $k$ nearest neighbors $\mathcal{R}$ by finding a ball $B(q, r^\star)$ centered at the query $q$ with the smallest radius $r^\star$ that contains $k$ vectors in the dataset $D$. Here, $r^\star$ is also the $k$-th smallest distance between a vector in $D$ and query vector $q$. Formally, we define it as follows.

**Definition 2.1:**[$k$-NN] Given a dataset $D$ and a query vector $q$, the set of $k$-NN of $q$ is $\mathcal{R} = \{u \in D \mid \delta(u, q) \leq r^\star\}$, where $r^\star = \arg\min_r r$ s.t. $|\{u \in D \mid \delta(u, q) \leq r\}| \geq k$.

With multi-vector query $q = [q_1, \cdots, q_m]$, where $q_i \in \mathbb{R}^d$ for $i \in \{1, .., m\}$, the definition of $k$-NN can be expanded by associating $m$ balls $B_i(q_i, r)$ centered at each query vector $q_i$. We consider two types of searches on multi-vector queries: **all** and **any**-$k$ NN queries, focusing on the intersection and union of $m$ balls, respectively. For example, as depicted in Fig. 2, dataset $D$ contains 7 vectors $u_1, \ldots, u_7$, and the query $q$ consists of three vectors $q_1, q_2$, and $q_3$. The all-2 NN of $q$ are $u_1$ and $u_4$, illustrated in Fig. 2(a), located within the intersection of the balls centered at $q_1, q_2$, and $q_3$ with the smallest radius. On the other hand, the any-4 NN of $q$ are $u_2, u_3, u_4$, and $u_5$, as shown in Fig. 2(b), within the union of balls centered at $q_1, q_2$, and $q_3$ with the smallest radius. It is important to note that $u_1$ is excluded from the union of the three balls since it lies at around the centroid of the balls, which is not included in any ball.

Formally, we present the definitions of all/any-$k$ NN as follows.

**Definition 2.2:**[All-$k$ NN] Given a dataset $D$ and a query of $m$ vectors $q = [q_1, \cdots, q_m]$, the set of all-$k$ NN of $q$ is

$$\mathcal{R}^{\forall}_{r^\star} = \bigcap_{i=1}^{m} \{u \in D \mid \delta(u, q_i) \leq r^\star\},$$

where $r^\star = \arg\min_r r$ s.t. $\left|\bigcap_{i=1}^{m} \{u \in D \mid \delta(u, q_i) \leq r\}\right| \geq k$.

**Definition 2.3:**[Any-$k$ NN] Given a dataset $D$ and a query of $m$ vectors $q = [q_1, \cdots, q_m]$, the set of any-$k$ NN of $q$ is

$$\mathcal{R}^{\exists}_{r^\star} = \bigcup_{i=1}^{m} \{u \in D \mid \delta(u, q_i) \leq r^\star\},$$

where $r^\star = \arg\min_r r$ s.t. $\left|\bigcup_{i=1}^{m} \{u \in D \mid \delta(u, q_i) \leq r\}\right| \geq k$.

As the exact $k$-NN search is time-consuming [25, 27, 40, 42] due to the curse of dimensionality [20], in this paper, we study the **all/any-$k$ approximate nearest neighbor ($k$ ANN) search**, and its quality is measured by the *recall*. Here, let $D_G$ be the ground-truth vector set, and $D_R$ be the set found by an all/any-$k$ ANN search algorithm, *recall@k* is defined as $|D_G \cap D_R|/k$.

The state-of-the-art approaches for $k$-ANN search are to construct a **proximity graph (PG)** [13, 14, 27, 29, 34, 35, 45, 48, 49]. Here, a proximity graph is a directed graph $G = (V, E)$, where $v_i \in V$ represents a unique vector $v_i \in D$, and an edge, $(v_i, v_j)$, in $E$ represents that two vectors are close to each other based on the distance function $\delta(v_i, v_j)$, or precisely $v_j$ is one of the $k$-NN of $v_i$. The search algorithm on the proximity graph, named **beam search**, starts its exploration from a designated entry node or randomly chosen nodes. It maintains a queue of $w$ nodes with the current smallest distance to the query vector, where $w$ is a given parameter, called beam width. Through iterative explorations, i.e., it explores the neighbors of the nodes in the queue each time, it seeks nodes closer to $q$. The algorithm terminates when no closer neighbors to $q$ are found among the nodes in the queue.

## 3 Baseline Approaches and Limitations

Many approaches focus on ANN search for multiple attributes by amalgamating vectors from different attributes of an item into a single-reference vector for search [19, 41, 52–54]. Their approach involves dividing the single-reference vector in a query, then executing separate $k$-ANN queries for each divided vector and combining their results to obtain the final results. We consider adapting their methods to our problem as a baseline algorithm. Specifically, let $\mathcal{R}_i$ denote the $k$-ANN result of query vector $q_i$, it determines the $r^\star$ among the $\bar{\mathcal{R}} = \bigcup_{i=1}^{m} \mathcal{R}_i$, i.e., for all-$k$ ANN, $r^\star$ is the smallest value such that $\left|\{u \in \bar{\mathcal{R}} | \forall i \in \{1, \cdots, m\}, \delta(u, q_i) \leq r^\star\}\right| \geq k$; and for any-$k$ ANN, $r^\star$ is the smallest value such that $\left|\{u \in \bar{\mathcal{R}} | \exists i \in \{1, \cdots, m\}, \delta(u, q_i) \leq r^\star\}\right| \geq k$.

Although it can be proved that it guarantees finding the any-$k$ NN of the query when the $k$-ANN retrieval is exact for each query $q_i$ (details are shown in Appendix), it may miss true results for the all-$k$ NN query. This is because some of the all-$k$ NN might not be included in $k$-NN of any query vector $q_i$. For example, consider the case illustrated in Fig. 2, where a query contains three vectors $q_1$, $q_2$, and $q_3$. When the query is an all-2 ANN search, it retrieves $\{u_2, u_3\}$, $\{u_5, u_6\}$, and $\{u_4, u_6\}$ as the 2-NN for $q_1$, $q_2$, and $q_3$ respectively, as these are their respective 2-NN vectors. In this case, the all-1 NN, $u_1$, of the query $q = [q_1, q_2, q_3]$ will be missed in the final output.

To address this issue, we further utilize the iterative merging method from Milvus [41]. That is, the $k'$-ANN search performs with an adaptive $k'$ for each query vector $q_i$, and starting with $k' = k$. At each time, the algorithm doubles $k'$ iteratively until the vectors in the final result set are included in all the $k'$-ANN of every query vector $q_i$, i.e., if $\bar{\mathcal{R}}^{\forall} = \{u \in \bar{\mathcal{R}} | \forall i \in \{1, \cdots, m\}, \delta(u, q_i) \leq r^\star\}$ with the smallest $r^\star$ is the final result set, then the algorithm terminates when $\forall i \in \{1, \cdots, m\}, \bar{\mathcal{R}}^{\forall} \subseteq \mathcal{R}_i$. For example, recalling the example shown in Fig. 2, $k' = k = 2$ initially, and it retrieves $\{u_2, u_3\}$, $\{u_5, u_6\}$, and $\{u_4, u_6\}$ as the 2-ANN for $q_1$, $q_2$, and $q_3$ respectively, and determine an smallest $r^\star$ such that final results are $u_6$ and $u_4$. However, given that neither of them is included in all 2-ANN sets of each query vector, the algorithm doubles $k'$ to 4. Next, it retrieves $\{u_2, u_3, u_1, u_4\}$, $\{u_5, u_6, u_1, u_4\}$, and $\{u_4, u_6, u_1, u_7\}$ as the 4-ANN for $q_1$, $q_2$, and $q_3$ respectively, culminating in the final results $\{u_1, u_4\}$.

Hence, the algorithm finds $\{u_1, u_4\}$ as the all-2 ANN, as both of them are in all 4-ANN sets of each query vector. It can be proved that, the final result set $\bar{\mathcal{R}}^{\forall}$ is guaranteed to be the all-$k$ NN of query vectors $q = [q_1, \cdots, q_m]$, when the $k'$-ANN retrieval is exact for each query $q_i$, the details are shown in Appendix.

**The Optimal $k'$ Setting Issues**: It is still inefficient in practice for both all and any-$k$ ANN search. ❶ Although the algorithm ensures accuracy in retrieving results for all-$k$ ANN, it tends to be inefficient in practice. It is because, for optimal performance, the iterative merging approach must initially establish an optimal value, $k'$, to obviate the need for iterative doubling. Otherwise, the iterative doubling of $k'$ necessitates repeated $k'$-ANN searches on each query vector, leading to decreased efficiency. However, determining the ideal $k'$ value before the search operation requires prior knowledge, making it difficult to determine the optimal value of $k'$. ❷ Conducting a separate $k$-ANN search on each query vector for any-$k$ ANN may also lose some efficiency in practice. For example, in Fig. 2, the any-4 NN of query $q = [q_1, q_2, q_3]$ are $u_2, u_3, u_4, u_5$, i.e., 2-NN of $q_1$, 1-NN of $q_2$ and $q_3$. Thus, it may be possible to identify a $k' < k$ such that performing $k'$-ANN searches for each query vector is adequate for acquiring the any-$k$ ANN. Even more, the optimal $k'$ value may vary for distinct query vectors. Therefore, it is inefficient to retrieve the complete $k$-ANN for every query vector in order to obtain the final results of any-$k$ ANN search.

## 4 Distance Metrics and Search Algorithms

To address the optimal $k'$ setting issue presented in the previous section for all/any-$k$ ANN search, we aim to eliminate conducting individual $k'$-ANN searches for each query vector. Instead, we consider performing a global search for all and any-$k$ ANN.

To achieve this, we introduce two distance metrics to assess the ranking of each vector among those in the dataset for all/any-$k$ NN, and prove that they are suitable for search based on the proximity graphs. This allows us to employ the beam search on graph indexes designed for $k$-ANN search by simply switching the distance metrics for executing all and any-$k$ ANN searches.

**Distance Metrics of Vectors in Dataset**: Reducing the value of $r^\star$ leads to a reduced $k$ for all/any-$k$ NN. In other words, increasing $r^\star$ enlarges the size of balls centered at each query vector, thereby expanding their intersections and unions, i.e., enlarging the size of $\mathcal{R}^{\forall}_{r^\star}$ and $\mathcal{R}^{\exists}_{r^\star}$. Hence, the ranking of each vector in all/any-$k$ NN is determined by their appearance order in the intersections and unions of balls when the value of $r^\star$ increases from 0. Thus, we select the minimum value of $r^\star$ that includes each vector in $\mathcal{R}^{\forall}_{r^\star}$ or $\mathcal{R}^{\exists}_{r^\star}$ to determine its rank in all/any-$k$ NN, i.e., a vector with a smaller $r^\star$ value means it appears earlier and has a higher ranking.

This smallest value of $r^\star$ for a vector $u$, named **all/any-radius** and denoted as $\dot{r}^\star(u)$, equals to $\arg\min_r r$ s.t. $u \in \mathcal{R}^{\forall}_r$ for all-$k$ NN (or $\arg\min_r r$ s.t. $u \in \mathcal{R}^{\exists}_r$ for any-$k$ NN). To calculate all/any-radius for determining vectors' rank, we rely on the following theorem.

**Theorem 4.1:** *Given a dataset $D$, for a vector $u \in D$, its all-radius $\dot{r}^\star(u)$ equals to $\max_{i\in\{1,\cdots,m\}} \delta(q_i, u)$, and its any-radius $\dot{r}^\star(u)$ equals to $\min_{i\in\{1,\cdots,m\}} \delta(q_i, u)$.*

**Proof Sketch:** For all-radius of $u$, let $r' = \max_{i\in\{1,\cdots,m\}} \delta(q_i, u)$. For all $i \in \{1, \cdots, m\}$, $\delta(q_i, u) \le r'$, implying $u \in \mathcal{R}^{\forall}_{r'}$. As $\dot{r}^\star(u) = \arg\min_r r$ s.t. $u \in \mathcal{R}^{\forall}_r$ by definition, it follows that $r' \ge \dot{r}^\star(u)$. Conversely, $\dot{r}^\star(u) = \arg\min_r r$ s.t. $u \in \mathcal{R}^{\forall}_r$, hence for all $i \in \{1, \cdots, m\}$, $\delta(q_i, u) \le \dot{r}^\star(u)$, indicating $\dot{r}^\star(u) \ge \max_{i\in\{1,\cdots,m\}} \delta(q_i, u)$. Thus, the equality $\dot{r}^\star(u) = \max_{i\in\{1,\cdots,m\}} \delta(q_i, u)$ is upheld.

For any-radius of $u$, let $r' = \min_{i\in\{1,\cdots,m\}} \delta(q_i, u)$. There exists one $i \in \{1, \cdots, m\}$, $\delta(q_i, u) = r'$, implying $u \in \mathcal{R}^{\exists}_{r'}$. As $\dot{r}^\star(u) = \arg\min_r r$ s.t. $u \in \mathcal{R}^{\exists}_r$ by definition, it follows that $r' \ge \dot{r}^\star(u)$. Conversely, $\dot{r}^\star(u) = \arg\min_r r$ s.t. $u \in \mathcal{R}^{\exists}_r$, hence $\exists i \in \{1, \cdots, m\}$, $\delta(q_i, u) \le \dot{r}^\star(u)$, indicating $\dot{r}^\star(u) \ge \min_{i\in\{1,\cdots,m\}} \delta(q_i, u)$. Therefore, $\dot{r}^\star(u) = \min_{i\in\{1,\cdots,m\}} \delta(q_i, u)$ is upheld. □

Leveraging the above theorem, we can first compute $\delta(q_i, u)$ for every query vector $q_i$ in the query $q$ in $O(d)$ time, and then aggregate them to calculate the all/any-radius. Therefore, for each vector in the dataset, it needs a time complexity of $O(m \cdot d)$ .

The beam search excels in performance on proximity graphs for $k$-ANN search due to the following two reasons [27, 34, 42, 48, 49, 52]. ❶ Since the Euclidean distances adhere to the triangle inequality, for a node $u$ and its neighbor $v$ in the proximity graph and a query vector $q$, we have $\delta(q, u) - \delta(u, v) \le \delta(q, v) \le \delta(q, u) + \delta(u, v)$, which implies that the neighbors of a node might be closer to the query than the current node. Hence, we can iteratively explore the neighbors to approach the region of the $k$-NN of the query vector. ❷ When a vector is one of the $k$-NN, the vectors represented by the neighbors of its corresponding nodes in the proximity graph have a high probability of also being $k$-NN. Hence, we can continuously explore the $k$-NN of the query vector, once we enter the region of the $k$-NN of the query vector, i.e., already finding at least one of the $k$-NN of the query vector.

Therefore, as follows, we demonstrate that the all/any-radius also exhibits these two properties, proving its suitability as a distance metric in beam search to search on proximity graphs.

We first prove that the neighbors of a node in the proximity graph have the potential to have a smaller all/any-radius compared to the current node, which aids in iteratively navigating the search towards the target region, i.e., the region contains all/any-$k$ NN.

**Theorem 4.2:** *For two vectors $u, v \in \mathbb{R}^d$, their all/any-radius satisfies $\dot{r}^\star(v) - \delta(u, v) \le \dot{r}^\star(u) \le \dot{r}^\star(v) + \delta(u, v)$.*

The proof is omitted here, which is included in Appendix.

According to Theorem 4.2, utilizing the all/any-radius as the distance metric holds the same potential as using Euclidean distance, suggesting that the neighbors of a node in the proximity graph might be closer to the query. Moreover, based on Theorem 4.2, we have $\left|\dot{r}^\star(u) - \dot{r}^\star(v)\right| \le \delta(u, v)$ for any two vectors $u, v \in D$. Thus, if node $u$ is among the all/any $k$-NN of a query, a neighbor $v$ of $u$ in the proximity graph has a higher probability of also being one of the all/any $k$-NN compared to other nodes in the dataset. It is because the neighbors of node $u$ are the closest vectors in dataset.

Thus, we have shown that the all/any-radius serves as a suitable distance metric in the beam search conducted on proximity graphs. Building on this insight, we introduce our search algorithm, named RadiusSearch. Similar to beam search, it maintains a queue containing $w$ nodes. Through iterative exploration, it explores the neighbors of nodes in the queue to identify those with a smaller value of all/any-radius. The details of our search algorithm are shown in Algorithm 1. Initially, it selects a node $u$ in $G$ and inserts

**Algorithm 1**: RadiusSearch $(G, q, k, w)$

**Input** : a PG $G$, a query $q = [q_1, q_2, \cdots, q_m]$, $k$ for top-$k$, and beam width $w$
**Output** : all/any-$k$ ANN of $q$

1 Select a node $u \in G$ to insert into the set $S$;
2 **while** there exists an unvisited node in $S$ **do**
3  $u \leftarrow$ the unvisited node with smallest value of $\dot{r}^{\star}(\cdot)$ of $q$ in $S$;
4  Mark $u$ visited;
5  **for** each unvisited $v$ of $(u, v) \in G$ that is not in $S$ **do**
6  Insert $v$ into $S$;
7  **while** $|S| > w$ **do**
8  Remove the node with the largest value of $\dot{r}^{\star}(\cdot)$ of $q$ from $S$;
9 **return** the top-$k$ nodes with smallest value of $\dot{r}^{\star}(\cdot)$ of $q$ in $S$;

it into set $S$, which is unvisited (line 1). The node $u$ is the entry point if specified by the given proximity graph or is randomly selected from the vertex set. In the while-loop (lines 2-8), if there exists an unvisited node in $S$, the search tries to find nodes with a smaller value of all/any-radius as follows. First, it selects the unvisited node with the smallest distance, $u$, to the query $q$ from $S$, and marks it visited (lines 3-4). Second, it adds every node, $v$, into $S$ if there is a directed edge from $u$ to $v$, and $v$ is not in $S$, which implies that $v$ is unvisited yet (lines 5-6). Third, it deletes the node with the largest value of all/any-radius to $q$ from $S$ if $|S|$ (the size of $S$) is greater than $w$ (lines 7-8). Since our search algorithm, RadiusSearch, extends beam search by modifying the distance measure, indicating they have the same time and space complexity.

## 5 Two-Stage Searches for All/Any-$k$ ANN

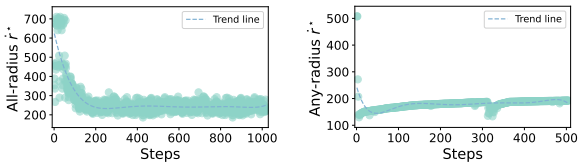
In this section, we present more efficient and effective approaches for the all and any-$k$ ANN search, respectively.

As illustrated in Fig. 4, at a high level, RadiusSearch for all/any-$k$ ANN search comprises two stages. Initially, the search approaches the target vector region, i.e., the region of all/any-1 NN, despite significant oscillations in vector distances. Next, the search stabilizes and gradually moves away from the target vector region in an approximate manner to search the all/any-$k$ ANN. Building upon this observation, we enhance our algorithms for all and any-$k$ ANN searches by decoupling two steps.

We refine our algorithm presented in the previous section as follows. ❶ Regarding any-$k$ ANN search, we identify that the region of its $k$-NN may not be a continuous space, limiting the effectiveness of RadiusSearch in achieving high recall. Thus, we propose a two-step algorithm for any-$k$ ANN search: initially approaching the separate non-continuous regions of the any-$k$ NN, followed by leveraging RadiusSearch to retrieve the results. ❷ For the two-step algorithm of the all-$k$ ANN search, we optimize its first step to approach the all-1 NN. By computing a vector near the all-1 NN first and conducting a traditional 1-ANN search to reach the target region initially, we reduce the distance computations from $O(m \cdot d)$ to $O(d)$ in the first step. Next, in a similar way, we employ RadiusSearch to acquire the results in the second step.

### 5.1 Search Algorithm for Any-$k$ ANN Search

First, we present an issue in the search process of RadiusSearch.




(a) All-10 ANN (b) Any-10 ANN

**Figure 4:** RadiusSearch **Search Process When** $w = 100$ **on** SIFT

**Non-continuous Space Issue of Any-$k$ NN**: The region of the any-$k$ NN may be non-continuous, since some of the balls centered at query vectors may not intersect with other balls. This makes the any-$k$ NN might not exist within a continuous space alongside the any-1 NN. For instance, if the any-2 NN of query $q = [q_1, q_2]$ are $u_1$ and $u_2$ with $\delta(q_1, u_1) = \delta(q_2, u_2) = 0$, a substantial Euclidean distance separates vectors $u_1$ and $u_2$ when $\delta(q_1, q_2)$ is large, disrupting the continuity of the space. However, similar to the principles RadiusSearch operates on all-$k$ ANN search, when executing a RadiusSearch for any-$k$ ANN search, it initially navigates towards the region of the any-1 NN. Since $k$-NN might not exist within a continuous space alongside the any-1 NN, RadiusSearch encounters an issue in searching for any-$k$ ANN after initially approaching the region of the any-1 NN. It is also evident in the results illustrated in Fig. 4(b). At step 6, RadiusSearch has already reached the any-1 NN. However, there exists another region of vectors with a small value of any-radius value that is not in proximity to the any-1 NN, and it will only be discovered around step 300.

Thus, despite RadiusSearch addressing the optimal $k'$ setting issue, it still has suboptimal performance due to the discontinuity in the target region of any-$k$ NN. This is also validated in Exp. 2 (Section 6), where the search performance of RadiusSearch may be inferior to existing approaches, e.g., Milvus.

To address this issue, we introduce a two-step search algorithm. RadiusSearch+ **of Any-$k$ ANN Search**: In the first step, instead of directly employing RadiusSearch to approach the region of any-1 NN, we opt to conduct separate 1-ANN searches for each query vector to approach the distinct $m$ regions which may not be in a continuous space, i.e., it involves performing a beam search to seek 1-ANN for each query vector individually. Next, in the second stage, we start RadiusSearch by initially inserting the $m$ 1-ANN of each query vector into the set $S$ (line 1 in Algorithm 1). By executing this strategy, RadiusSearch avoids approaching only one of the regions in the first step, and starting RadiusSearch at these non-continuous regions enables an easier identification of any-$k$ ANN.

### 5.2 Search Algorithm for All-$k$ ANN Search

To propose the two-step algorithm for all-$k$ ANN search, it is necessary to initially compute a vector in $\mathbb{R}^d$ that is proximate to the all-1 NN. Note that, in Section 2, the example (Fig. 2(a)) shows that the all-1 NN is close to the centroid of all query vectors. However, this scenario is not universally applicable. For instance, in a two-dimensional space where $q_1 = (0, 0), q_2 = \cdots = q_{10} = (4, 4)$, the centroid is very close to $(4, 4)$, yet the all-1 NN are around $(2, 2)$.

Therefore, a new method to locate a vector in $\mathbb{R}^d$ that is close to the all-1 NN is necessary.

**A Vector Near the All-1 NN**: Assuming the dataset contains arbitrary vectors in $\mathbb{R}^d$. As the value of $r^\star$ increases, initially, $\mathcal{R}^\forall_{r^\star}$ contains no data points because the balls $\{u \in D \mid \delta(u, q_i) \le r^\star\}$ have no intersections. The first data point exists in $\mathcal{R}^\forall_{r^\star}$ as $r^\star$ increases, when the boundaries of the balls $\{u \in D \mid \delta(u, q_i) \le r^\star\}$ intersect. This first data point, being the all-1 NN, due to the dataset containing arbitrary vectors in $\mathbb{R}^d$, is referred to as the **optimal intersection vector** $\bar{v}$. Hence, $\bar{v}$ is the vector in $\mathbb{R}^d$ that has the smallest value of all-radius. Indeed, the real dataset contains only a finite set of vectors (not arbitrary vectors in $\mathbb{R}^d$), hence, the all-1 NN of the dataset may only appear when $r^\star$ continues to increase. According to the insights from Theorem 4.2, a vector $u$ with a small Euclidean distance to $\bar{v}$ may exhibit a similar all/any-radius value as $\bar{v}$. Thus, the all-1 NN of the dataset $D$ will be near the optimal intersection vector $\bar{v}$, i.e., $\bar{v}$ is a vector near the all-1 NN.

Theoretically, we can prove that the optimal intersection vector $\bar{v}$ corresponds to the center of the minimum enclosing ball (MEB) of all the query vectors. The MEB $B$ of a set of vectors $S \subseteq \mathbb{R}^d$ is a $d$-dimensional ball with minimized radius that contains $S$ within it.

**Theorem 5.1:** *Given a query $q = [q_1, \cdots, q_m]$, the optimal intersection vector $\bar{v}$ of $q$ is the center of the MEB of vectors $\{q_1, \cdots, q_m\}$.*

**Proof Sketch:** Since $\bar{v}$ is the first point found in the intersection of all balls $B_i(q_i, r^\star)$, there exists a ball $B_k(q_k, r^\star)$ corresponding to a query $q_k$ such that $\bar{v}$ is exactly on the boundary of the ball and the distance from the ball center to $\bar{v}$ is the largest, i.e., $\delta(\bar{v}, q_k) = r^\star$. As the distance from the ball center to $\bar{v}$ is the largest and $\bar{v}$ is in the intersection of all balls, the ball $B(\bar{v}, r^\star)$ centered at $\bar{v}$ contains all other query vector $q_i$ and is thus an enclosing ball. Next, assuming that the ball $B(\bar{v}, r^\star)$ is not the minimum enclosing ball, there must exist a ball $B(c, r)$ centered at $c$, such that $r < r^\star$, encompassing all query vectors. Hence, $\delta(c, q_i) \le r$, meaning that $c$ will appear in the intersection of all balls $B_i(q_i, r)$, which contradicts the optimality of $\bar{v}$ as the intersecting vector with $r^\star$. □

For example, as shown in Fig. 3, $\bar{v}$ in the center of MEB of vector set $S = \{q_1, q_2, q_3\}$, which is the point that has the minimum all-radius. Consider the vectors shown in Fig. 2(a), the all-2 NN $u_1$ and $u_4$ are near $\bar{v}$. Therefore, we can utilize the MEB center of query vectors $c$ as a vector in the target vector region for search in the first step. Hence, in the first stage of the all-$k$ ANN search, we execute a beam search by inputting the query vector as $c$, and use the Euclidean distance as the distance metric.

**Issue of the Existing Approaches for Calculating MEB**: There have been many approaches for determining the center of MEB [10, 12, 24, 50], and the state-of-the-art approaches can achieve a linear time complexity in the number of vector queries. However, these methods typically treat the dimension as a constant or assume $d \ll m$, rendering them unsuitable for ours. In our scenario, the dimension $d$ is consistently much greater than the number of vectors $m$ since we are dealing with high-dimensional vectors, and the user-defined query vectors are not excessively numerous. Thus, we need to consider dimension $d$ in terms of time complexity for proposing the algorithm of MEB.

**Our Method for Calculating the Center of MEB**: Given that $m < d$, we find that the query vectors are all positioned on the boundary of the MEB [12]. Hence, we first formulate the problem of determining the MEB's center according to the following theorem.

**Theorem 5.2:** *Given a query $q = [q_1, \cdots, q_m]$, the MEB of vectors $q_1, \cdots, q_m$ is the ball $B(c, r)$, where $c = \arg\min_{c \in \mathbb{R}^d} c^\top (c - 2q_1)$ s.t. $\forall q_i \in q, 2(q_1 - q_i)^\top c = \|q_1\|^2 - \|q_i\|^2$ and $r = \delta(q_1, c)$. Here, $\|u\|$ denotes the L2 norm of a vector $u$.*

**Proof Sketch:** According to the definition of MEB, and since all $q_i$ are on the boundary of MEB, we have the ball $B(c, r)$ is MEB, when $r$ and $c$ satisfy that $c = \arg\min_{c \in \mathbb{R}^d} r$ s.t. $\forall q_i, q_j \in q, \delta(c, q_i) = \delta(c, q_j)$ and $r = \delta(q_1, c)$. Since $\forall q_i, q_j \in q, \delta(c, q_i) = \delta(c, q_j) \Leftrightarrow \forall q_i \in q, \delta(c, q_1) = \delta(c, q_i)$; $\delta(c, q_1) = \delta(c, q_i) \Leftrightarrow \|q_i\|^2 + \|c\|^2 - 2c^\top q_i = \|q_1\|^2 + \|c\|^2 - 2c^\top q_1 \Leftrightarrow 2(q_1 - q_i)^\top c = \|q_1\|^2 - \|q_i\|^2$; $\|c\|^2 = c^\top c$; and $\|q_1\|^2$ is given, we have $c = \arg\min_{c \in \mathbb{R}^d} c^\top (c - 2q_1)$ s.t. $\forall q_i \in q, 2(q_1 - q_i)^\top c = \|q_1\|^2 - \|q_i\|^2$. □

Note that $\forall q_i \in q, 2(q_1 - q_i)^\top c = \|q_1\|^2 - \|q_i\|^2$ are linear constraints, and $c^\top (c - 2q_1)$ is convex. Hence, the formalized problem is a convex quadratic program with linear equality constraints. We can utilize the interior-point method with optimization via Schur complement to effectively address this problem [44], with a resolution time complexity of $O(\sqrt{m} \cdot (d^3 + m^3))$.

Since $m < d$, we first calculate the affine hull of $m$ vectors within $\mathbb{R}^d$, possessing a maximum dimension of $m - 1$ [4]. Next, we utilize an orthonormal basis for the coordinate system of the affine hull to project these $m$ vectors onto the affine hull. As orthonormal bases are isometric, the original Euclidean distances between the vectors are preserved [39]. Hence, our task reduces to finding the center of the MEB within this affine hull. Therefore, the dimension of the variable $c$ in the problem presented in Theorem 5.1 can be reduced to at most $m - 1$, thereby reducing the time complexity of solving the problem to $O(m^{3.5})$. Additionally, calculating the affine hull of $m$ vectors and projecting them onto the affine hull requires $O(m^2 d)$ time [4]. Thus, the overall time complexity of our MEB center calculation method is $O(m^2 d + m^{3.5})$.

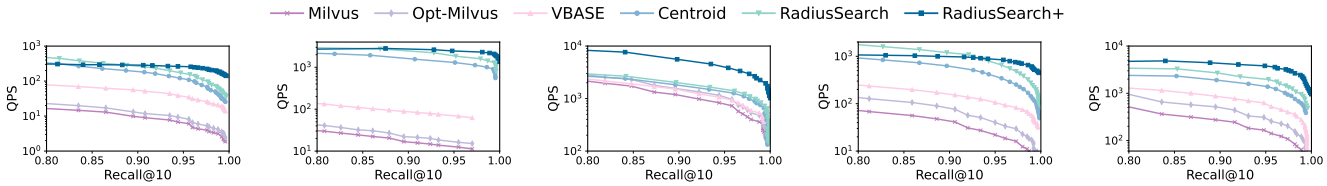
RadiusSearch+ **for All-$k$ ANN Search**: In summary, our two-stage algorithm for all-$k$ ANN search, named RadiusSearch+, operates as follows. Initially, it determines the optimal intersection vector $\bar{v}$ by computing the MEB of the query vectors. Next, it employs a beam search to find a traditional 1-ANN of $\bar{v}$, and uses the Euclidean distance as the metric. Finally, the results are obtained by RadiusSearch, which initiates from the results of the beam search, i.e., by replacing line 1 of Algorithm 1 with the insertion of nodes from the beam search results into $S$.

## 6 Evaluation

In this section, we conduct extensive experiments on real-world datasets and report our findings.

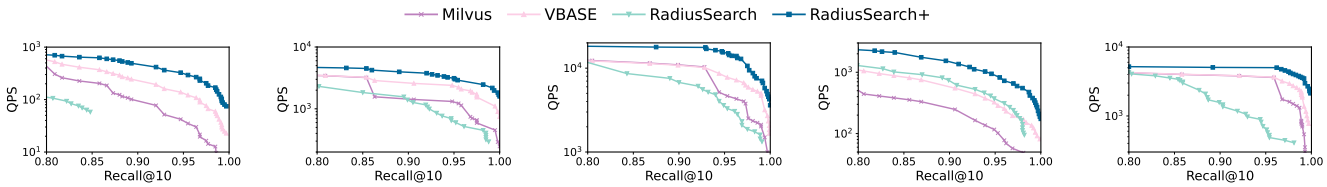
**Datasets**: We employ 5 real-world datasets with different numbers of dimensions/vectors, from diverse applications including image (SIFT [1], GIST [1], Recipe [36]), audio (Msong [3]), and text (Crawl [2]). The details of the datasets are included in Appendix.

We generate 1,000 queries for each dataset for the evaluation, and each query is assigned a set of 5 vectors by default (i.e., $m = 5$). To select vectors for each query, we begin by randomly selecting a vector from the query dataset and finding its 10-NN within the same




(a) Recipe (b) Msong (c) SIFT (d) GIST (e) Crawl

**Figure 5: QPS v.s. Recall Curves for Comparing All-$k$ ANN Search Performance (Exp. 1)**




(a) Recipe (b) Msong (c) SIFT (d) GIST (e) Crawl

**Figure 6: QPS v.s. Recall Curves for Comparing Any-$k$ ANN Search Performance (Exp. 2)**

query dataset. These 10-NN are distinct from the base dataset used to construct the index since the query dataset is provided separately. Next, the 5 vectors are randomly picked from this 10-NN set.

**Algorithms**: In the experiments, we evaluate our approaches against two existing methods by modifying their techniques to tackle our problem as baseline algorithms. **(1)** Milvus [41], discussed in Section 3, performs $m$ $k$-ANN searches for each query vector in the query and aggregates the results for any-$k$ ANN. For all-$k$ ANN, it utilizes a parameter $k'$ that iteratively doubles when the results are insufficient. **(2)** Opt-Milvus, for all-$k$ ANN queries, the experimental results reveal that $k'$ in Milvus is always no larger than $2k$, indicating a maximum of two iterative searches. Hence, we also compare our approaches to the method Opt-Milvus by setting its $k' = 2k$ directly, which can run without iteration to enhance its efficiency. **(3)** VBASE [52] is a state-of-the-art vector database system that supports multi-attribute queries, i.e., the items in the database comprising multiple vectors, a multi-vector query utilizing an aggregate function on these multiple vectors as a distance metric. Here, for our proposed all and any-$k$ ANN search queries, each item in the database can be seen to contain $m$ identical vectors, with the aggregate function being our proposed all/any-radius in Section 4. At a high level, it involves a two-phase search method. Initially, it searches for the 1-ANN of each query vector and subsequently conducts all/any-$k$ ANN searches based on these results. **(4)** Centroid, to highlight the effectiveness of our approaches in all-$k$ ANN queries, we also compare to a baseline, Centroid, where our RadiusSearch algorithm directly starts search from the centroid of all query vectors. **(5)** RadiusSearch and RadiusSearch+, our approaches proposed in Section 4 and 5, respectively. All our source codes are available at https://anonymous.4open.science/r/Multi-Vector-Queries/.

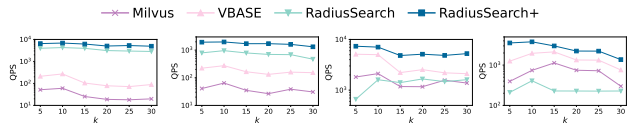
To ensure fair comparisons, all compared approaches construct an identical HNSW index [29] as the proximity graph for search. We fix the parameters for the HNSW index as $EF = 400$ and $M = 32$ across all datasets. For evaluating query performance, we fine-tune additional parameters on each dataset to achieve a Pareto-optimal recall-QPS curve. We set $k = 10$ for queries and assess the search algorithms' performance using a single thread. Parameters within the search algorithm, i.e., the beam width, are incrementally adjusted to attain the desired recall levels in our experiments.

**Performance Metrics**: Following existing ANN benchmarks [27, 42, 49], we employ QPS (Queries Per Second) to measure efficiency and use Recall to show the accuracy. QPS is the quantity of queries processed per second, while Recall is precisely defined in Section 2.

**Experimental Environments**: The experiments are conducted on a Linux server with an AMD EPYC 7443 24-Core Processor and 1024 GB memory. All algorithms are implemented in C++14. The code is compiled with g++ 8.5 under O3 optimization.

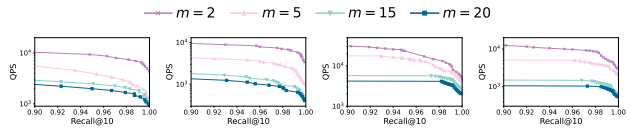
**Exp. 1: Overall Performance of All-$k$ ANN Search.** Fig. 5 presents the QPS-recall curves for our approaches and two existing methods. Across all datasets, our approaches consistently enhance overall query performance, significantly improving QPS by over 1 order of magnitude with the same recall. This improvement primarily stems from employing all-radius as the distance metric, enabling a global search without the need to set optimal $k'$ for each query vector. In comparing RadiusSearch+ with RadiusSearch, the results show that RadiusSearch+ may underperform RadiusSearch initially at lower recall levels but outperforms as recall increases. It is due to the fixed time required to compute the optimal intersection vector $\bar{v}$ while adjusting search parameters, which accounts for a significant ratio of overall time when the beam width $w$ is small (as illustrated in Exp. 5). But this limitation will decrease, and it even can be ignored as $w$ increases, i.e., aiming at a high recall level. Comparing Centroid with RadiusSearch, starting the search at the centroid of all query vectors does not enhance efficiency; in fact, it can even worsen it. This is primarily due to the centroid not consistently being proximate to the all-1 NN. Moreover, not starting the search from the entry point compromises search performance and fails to leverage the hierarchical structure of HNSW. Furthermore, Opt-Milvus enhances the efficiency of Milvus without impacting recall; however, it still much worse than our approaches.

**Exp. 2: Overall Performance of Any $k$-ANN Search.** Fig. 6 plots the QPS-recall curves by varying the parameter named beam width for the different search algorithms. Our two-step approach,



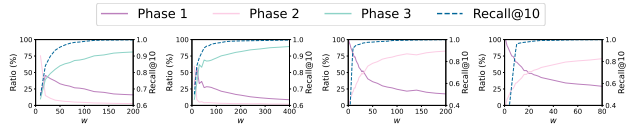

(a) SIFT (All) (b) Crawl (All) (c) SIFT (Any) (d) Crawl (Any)

**Figure 7: QPS at Recall@$k$ = 0.99 When Varying the $k$ (Exp. 3)**




(a) SIFT (All) (b) Crawl (All) (c) SIFT (Any) (d) Crawl (Any)

**Figure 8: Search Performance When Varying the $m$ (Exp. 4)**




(a) SIFT (All) (b) Crawl (All) (c) SIFT (Any) (d) Crawl (Any)

**Figure 9: Time Decomposition of** RadiusSearch+ **(Exp. 5)**

RadiusSearch+, consistently outperforms across all datasets. The first step of RadiusSearch+ incurs similar time costs regardless of the beam width parameter $w$, implying that its overall time ratio decreases as higher recall levels are achieved. Thus, the enhancements of RadiusSearch+ become more pronounced with higher recall targets. In contrast, the performance of RadiusSearch may even lag behind two existing approaches, as discussed in Section 5.1, since it solely approaches the region of any-1 NN, potentially leading to greater distances from the vectors in any-$k$ NN.

**Exp. 3: Varying the $k$ for Queries.** Fig. 7 presents the QPS of various algorithms for all and any-$k$ ANN searches when achieving a recall of 0.99. We adjust the $k$ values for both search types across $\{5, 10, 15, 20, 25, 30\}$ and present their corresponding QPS. Results on other datasets are included in Appendix. We can observe that RadiusSearch+ consistently outperforms all other approaches across all datasets and for both search types. Moreover, our method demonstrates stable performance when varying the $k$ value to achieve a recall of 0.99, highlighting its scalability across different scenarios of varying $k$ values.

**Exp. 4: Varying the Number of Vectors $m$ in Queries.** In this set of experiments, we examine the impact of the number of vectors in each query on RadiusSearch+. By varying the number of vectors $m$ in each query across $\{2, 5, 15, 20\}$, we plot the QPS-recall curves in Fig. 8 and results on other datasets are included in Appendix. The results indicate that the required time increases as $m$ increases to achieve the same recall level. However, the rate of time increase is proportionally lower than the scale of increase in $m$, which demonstrates the scalability of RadiusSearch+ in handling queries with a much larger number of vectors.

**Exp. 5: Time Decomposition of Our Approaches.** In Fig. 9, we present the time breakdown of RadiusSearch+ and the recall under varying values of beam width used in the search on SIFT and Crawl datasets. Results on other datasets are included in Appendix. For all-$k$ ANN search, RadiusSearch+ comprises three phases: (1) calculating the optimal intersection vector $\bar{v}$; (2) performing 1-ANN search when the query is $\bar{v}$; and (3) executing all-$k$ ANN via RadiusSearch. The results reveal that the first phase consumes a substantial amount of time, exceeding 50% when $w$ is small, i.e., at low recall levels. As $w$ increases, achieving a reasonable recall level, i.e., above 0.99, the proportion of time of the first phase is reduced, thereby enhancing the search performance of RadiusSearch+ for all-$k$ ANN search. For the any-$k$ ANN search, the process involves two phases: (1) conducting 1-ANN search for each query vector; and (2) executing any-$k$ ANN via RadiusSearch. While the first phase exclusively entails 1-ANN search under Euclidean distance, which is fixed under different values of $w$, it must be performed for each query vector, leading to a relatively high time ratio when $w$ is small. With an increase in $w$, this ratio significantly decreases, becoming much smaller than the time spent on the second phase.

**Exp.s 6 & 7: Different Selection Strategy and Other Distance Metrics.** We evaluate our approach RadiusSearch+ by varying the strategy for selecting query vectors and extending it to support cosine distance and inner product (two other commonly utilized distance metrics). The detailed results can be found in Appendix.

## 7 Conclusion and Future Work

In this paper, we introduce and study the all/any-$k$ ANN search problem. We propose two distance metrics for assessing rankings of vectors among the dataset for all/any-$k$ NN. Building upon this, we design a proximity graph-based search algorithm. By decoupling the two stages in our proposed search algorithm and further developing algorithms for the first stage, we boost efficiency and effectiveness for the all and any-$k$ ANN search problem, respectively. Our extensive experiments demonstrate that our proposed algorithms outperform all baseline methods by up to an order of magnitude in efficiency with superior quality in all/any-$k$ results.

We would like to mention the following extensions and possible directions as future work. (1) RadiusSearch+ algorithm is exclusive to graph-based indexing methods. Conversely, RadiusSearch algorithm can be adapted for use in non-graph-based approaches by employing the all/any radius as the distance metric, such as SCANN [18] and RabitQ [15, 16]. Exploring methods to enhance the efficiency of non-graph-based indexing for answering the all/any-$k$ ANN queries is a potential area for future research. (2) The theoretical guarantee of our approaches for all/any-$k$ ANN can be ensured by replacing HNSW with LMG proposed in [45]. However, LMG needs $O(n^2d)$ time and size for index construction. It is worth exploring a practical index with theoretical guarantees for the single-reference $k$-ANN, as it could be seamlessly integrated into our algorithms to achieve theoretical guarantees on results.

## Acknowledgment

This work was supported by the Jing-Jin-Ji Regional Integrated Environmental Improvement-National Science and Technology Major Project of Ministry of Ecology and Environment of China (No. 2025ZD1200600).

## References


[1] 2010. Datasets for approximate nearest neighbor search. http://corpus-texmex.irisa.fr/.

[2] 2023. Common Crawl. https://commoncrawl.org/.

[3] Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. 2011. The Million Song Dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference, ISMIR 2011*. University of Miami, 591–596.

[4] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *NeurIPS 2020*.

[6] Qi Chen, Xiubo Geng, Corby Rosset, Carolyn Buractaon, Jingwen Lu, Tao Shen, Kun Zhou, Chenyan Xiong, Yeyun Gong, Paul N. Bennett, Nick Craswell, Xing Xie, Fan Yang, Bryan Tower, Nikhil Rao, Anlei Dong, Wenqi Jiang, Zheng Liu, Mingqin Li, Chuanjie Liu, Zengzhong Li, Rangan Majumder, Jennifer Neville, Andy Oakley, Knut Magne Risvik, Harsha Vardhan Simhadri, Manik Varma, Yujing Wang, Linjun Yang, Mao Yang, and Ce Zhang. 2024. MS MARCO Web Search: A Large-scale Information-rich Web Dataset with Millions of Real Click Labels. In *WWW*. ACM, 292–301.

[7] Tingyang Chen, Cong Fu, Xiangyu Ke, Yunjun Gao, Yabo Ni, and Anxiang Zeng. 2025. Stitching Inner Product and Euclidean Metrics for Topology-aware Maximum Inner Product Search. In *SIGIR 2025*. ACM, 2341–2350.

[8] Yaoqi Chen, Ruicheng Zheng, Qi Chen, Shuotao Xu, Qianxi Zhang, Xue Wu, Weihao Han, Hua Yuan, Mingqin Li, Yujing Wang, Jason Li, Fan Yang, Hao Sun, Weiwei Deng, Feng Sun, Qi Zhang, and Mao Yang. 2024. OneSparse: A Unified System for Multi-index Vector Search. In *WWW 2024*. ACM, 393–402.

[9] Richard Connor, Alan Dearle, David Morrison, and Edgar Chávez. 2023. Similarity Search with Multiple-Object Queries. In *SISAP 2023*, Oscar Pedreira and Vladimir Estivill-Castro (Eds.), Vol. 14289. Springer, 223–237.

[10] Mark de Berg, Otfried Cheong, Marc J. van Kreveld, and Mark H. Overmars. 2008. *Computational geometry: algorithms and applications, 3rd Edition*. Springer.

[11] Ishita Doshi, Dhritiman Das, Ashish Bhutani, Rajeev Kumar, Rushi Bhatt, and Niranjan Balasubramanian. 2021. LANNS: A Web-Scale Approximate Nearest Neighbor Lookup System. *Proc. VLDB Endow.* 15, 4 (2021), 850–858.

[12] Kaspar Fischer, Bernd Gärtner, and Martin Kutz. 2003. Fast Smallest-Enclosing-Ball Computation in High Dimensions. In *Algorithms - ESA 2003, 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003, Proceedings (Lecture Notes in Computer Science, Vol. 2832)*. Springer, 630–641.

[13] Cong Fu, Changxu Wang, and Deng Cai. 2022. High Dimensional Similarity Search With Satellite System Graph: Efficiency, Scalability, and Unindexed Query Compatibility. *IEEE Trans. Pattern Anal. Mach. Intell.* 44, 8 (2022), 4139–4150.

[14] Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. 2019. Fast Approximate Nearest Neighbor Search With The Navigating Spreading-out Graph. *Proc. VLDB Endow.* 12, 5 (2019), 461–474.

[15] Jianyang Gao, Yutong Gou, Yuexuan Xu, Yongyi Yang, Cheng Long, and Raymond Chi-Wing Wong. 2025. Practical and Asymptotically Optimal Quantization of High-Dimensional Vectors in Euclidean Space for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 3, 3 (2025), 202:1–202:26.

[16] Jianyang Gao and Cheng Long. 2024. RaBitQ: Quantizing High-Dimensional Vectors with a Theoretical Error Bound for Approximate Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 3 (2024), 167.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable Feature Learning for Networks. In *SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 855–864.

[18] Ruiqi Guo, Philip Sun, Erik Lindgren, Quan Geng, David Simcha, Felix Chern, and Sanjiv Kumar. 2020. Accelerating Large-Scale Inference with Anisotropic Vector Quantization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020 (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 3887–3896.

[19] D. Frank Hsu and Isak Taksa. 2005. Comparing Rank and Score Combination Methods for Data Fusion in Information Retrieval. *Inf. Retr.* 8, 3 (2005), 449–480.

[20] Piotr Indyk and Rajeev Motwani. 1998. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing*. ACM, 604–613.

[21] Wenqi Jiang, Marco Zeller, Roger Waleffe, Torsten Hoefler, and Gustavo Alonso. 2024. Chameleon: a Heterogeneous and Disaggregated Accelerator System for Retrieval-Augmented Language Models. *Proc. VLDB Endow.* 18, 1 (2024), 42–52.

[22] Xinke Jiang, Rihong Qiu, Yongxin Xu, Wentao Zhang, Yichen Zhu, Ruizhe Zhang, Yuchen Fang, Chu Xu, Junfeng Zhao, and Yasha Wang. 2024. RAGraph: A General Retrieval-Augmented Graph Learning Framework. In *NeurIPS 2024*.

[23] Omar Khattab and Matei Zaharia. 2020. ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. In *SIGIR*. ACM, 39–48.

[24] Piyush Kumar, Joseph S. B. Mitchell, and E. Alper Yildirim. 2003. Approximate minimum enclosing balls in high dimensions using core-sets. *ACM J. Exp. Algorithmics* 8 (2003).

[25] Govinda D. Kurup. 1992. *Database Organized on the Basis of Similarities with Applications in Computer Vision*. Ph. D. Dissertation.

[26] Weiwei Li. 2025. Enhanced automated art curation using supervised modified CNN for art style classification. *Scientific Reports* 15, 1 (2025), 7319.

[27] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2020. Approximate Nearest Neighbor Search on High Dimensional Data - Experiments, Analyses, and Improvement. *IEEE Trans. Knowl. Data Eng.* 32, 8 (2020), 1475–1488.

[28] Mugeng Liu, Siqi Zhong, Qi Yang, Yudong Han, Xuanzhe Liu, and Yun Ma. 2025. WebANNS: Fast and Efficient Approximate Nearest Neighbor Search in Web Browsers. In *SIGIR 2025*. ACM, 2483–2492.

[29] Yury A. Malkov and Dmitry A. Yashunin. 2020. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. *IEEE Trans. Pattern Anal. Mach. Intell.* 42, 4 (2020), 824–836.

[30] Javier Marín, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. [n. d.]. Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. *IEEE Trans. Pattern Anal. Mach. Intell.* 43, 1 ([n. d.]), 187–203.

[31] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *27th Annual Conference on Neural Information Processing Systems 2013*. 3111–3119.

[32] Jason Mohoney, Anil Pacaci, Shihabur Rahman Chowdhury, Ali Mousavi, Ihab F. Ilyas, Umar Farooq Minhas, Jeffrey Pound, and Theodoros Rekatsinas. 2023. High-Throughput Vector Similarity Search in Knowledge Graphs. *Proc. ACM Manag. Data* 1, 2 (2023), 197:1–197:25.

[33] Nasser M Nasrabadi and Robert A King. 1988. Image coding using vector quantization: A review. *IEEE Transactions on communications* 36, 8 (1988), 957–971.

[34] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *Proc. ACM Manag. Data* 1, 1 (2023), 54:1–54:27.

[35] Liudmila Prokhorenkova and Aleksandr Shekhovtsov. 2020. Graph-based Nearest Neighbor Search: From Practice to Theory. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020 (Proceedings of Machine Learning Research, Vol. 119)*. PMLR, 7803–7813.

[36] Amaia Salvador, Nicholas Hynes, Yusuf Aytar, Javier Marín, Ferda Ofli, Ingmar Weber, and Antonio Torralba. 2017. Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. In *2017 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE Computer Society, 3068–3076.

[37] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW 10*. ACM, 285–295.

[38] Naoya Sogi, Takashi Shibata, and Makoto Terao. 2024. Object-Aware Query Perturbation for Cross-Modal Image-Text Retrieval. In *ECCV 2024*, Vol. 15137. Springer, 447–464.

[39] Gilbert Strang. 2000. *Linear algebra and its applications*.

[40] Philip Sun, David Simcha, Dave Dopson, Ruiqi Guo, and Sanjiv Kumar. 2023. SOAR: Improved Indexing for Approximate Nearest Neighbor Search. In *Thirty-seventh Conference on Neural Information Processing Systems*.

[41] Jianguo Wang, Xiaomeng Yi, Rentong Guo, Hai Jin, Peng Xu, Shengjun Li, Xiangyu Wang, Xiangzhou Guo, Chengming Li, Xiaohai Xu, Kun Yu, Yuxing Yuan, Yinghao Zou, Jiquan Long, Yudong Cai, Zhenxiang Li, Zhifeng Zhang, Yihua Mo, Jun Gu, Ruiyi Jiang, Yi Wei, and Charles Xie. 2021. Milvus: A Purpose-Built Vector Data Management System. In *SIGMOD '21*. ACM, 2614–2627.

[42] Mengzhao Wang, Xiaoliang Xu, Qiang Yue, and Yuxiang Wang. 2021. A Comprehensive Survey and Experimental Comparison of Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 14, 11 (2021), 1964–1978.

[43] Wenjie Wang, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2022. User-controllable Recommendation Against Filter Bubbles. In *SIGIR '22*. ACM, 1251–1261.

[44] Stephen J. Wright. 1997. *Primal-Dual Interior-Point Methods*. SIAM.

[45] Jiadong Xie, Jeffrey Xu Yu, and Yingfan Liu. 2025. Graph Based K-Nearest Neighbor Search Revisited. *ACM Trans. Database Syst.* (May 2025).

[46] Jiadong Xie, Jeffrey Xu Yu, Siyi Teng, and Yingfan Liu. 2025. Beyond Vector Search: Querying With and Without Predicates. *Proc. ACM Manag. Data* 3, 6 (2025), 1–26.

[47] Yuexuan Xu, Jianyang Gao, Yutong Gou, Cheng Long, and Christian S. Jensen. 2024. iRangeGraph: Improvising Range-dedicated Graphs for Range-filtering Nearest Neighbor Search. *Proc. ACM Manag. Data* 2, 6 (2024), 239:1–239:26.

[48] Ming Yang, Yuzheng Cai, and Weiguo Zheng. 2024. CSPG: Crossing Sparse Proximity Graphs for Approximate Nearest Neighbor Search. In *NeurIPS 2024*.

[49] Shuo Yang, Jiadong Xie, Yingfan Liu, Jeffrey Xu Yu, Xiyue Gao, Qianru Wang, Yanguo Peng, and Jiangtao Cui. 2025. Revisiting the Index Construction of Proximity Graph-Based Approximate Nearest Neighbor Search. *Proc. VLDB Endow.* 18, 6 (2025), 1825–1838.

[50] E. Alper Yildirim. 2008. Two Algorithms for the Minimum Enclosing Ball Problem.

*SIAM J. Optim.* 19, 3 (2008), 1368–1391.

[51] Ziqi Yin, Jianyang Gao, Pasquale Balsebre, Gao Cong, and Cheng Long. 2025. DEG: Efficient Hybrid Vector Search Using the Dynamic Edge Navigation Graph. *Proc. ACM Manag. Data* 3, 1 (2025), 29:1–29:28.

[52] Qianxi Zhang, Shuotao Xu, Qi Chen, Guoxin Sui, Jiadong Xie, Zhizhen Cai, Yaoqi Chen, Yinxuan He, Yuqing Yang, Fan Yang, Mao Yang, and Lidong Zhou. 2023. VBASE: Unifying Online Vector Similarity Search and Relational Queries via Relaxed Monotonicity. In *17th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2023*. USENIX Association, 377–395.

[53] Shaoting Zhang, Ming Yang, Timothée Cour, Kai Yu, and Dimitris N. Metaxas. 2015. Query Specific Rank Fusion for Image Retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 37, 4 (2015), 803–815.

[54] Jiongli Zhu, Yue Wang, Bailu Ding, Philip A Bernstein, Vivek Narasayya, and Surajit Chaudhuri. 2025. MINT: Multi-Vector Search Index Tuning. *arXiv preprint arXiv:2504.20018* (2025).

# A Theorems and Proofs

**Proof of Theorem 4.2.**

**Proof Sketch:** Since the all/any-radius $\dot{r}^\star(u) = \max/\min_{i\in\{1,\cdots,m\}} \delta(q_i,u)$, and $\delta(q_i,u) \le \delta(q_i,v) + \delta(u,v)$ always holds, we can derive $\dot{r}^\star(u) = \max/\min_{i\in\{1,\cdots,m\}} \delta(q_i,u) \le \delta(u,v) + \max/\min_{i\in\{1,\cdots,m\}} \delta(q_i,v) = \delta(u,v) + \dot{r}^\star(v)$. Similarly, as $\delta(q_i,u) \ge \delta(q_i,v) - \delta(u,v)$ always holds, we have $\dot{r}^\star(u) = \max/\min_{i\in\{1,\cdots,m\}} \delta(q_i,u) \ge -\delta(u,v) + \max/\min_{i\in\{1,\cdots,m\}} \delta(q_i,v) = \dot{r}^\star(v) - \delta(u,v)$. □

**Theorem A.1:** *Given a dataset $D$ and a query $q = [q_1,\cdots,q_m]$, let $\mathcal{R}_i$ be the k-NN of query vector $q_i$, $\bar{\mathcal{R}} = \bigcup_{i=1}^m \mathcal{R}_i$, and $\bar{\mathcal{R}}_r^\exists = \{u \in \bar{\mathcal{R}} | \exists i \in \{1,\cdots,m\}, \delta(u,q_i) \le r\}$, denote $r^\star$ is the smallest value such that $\left|\bar{\mathcal{R}}_{r^\star}^\exists\right| \ge k$, we have $\bar{\mathcal{R}}_{r^\star}^\exists$ is the any-k of $q$.*

**Proof Sketch:** Assume that one vector $v \in \bar{\mathcal{R}}_{r^\star}^\exists$ is not one of the any-$k$ NN of $q$, then there exist a vector $w \notin \bar{\mathcal{R}}_{r^\star}^\exists$ is one of the any-$k$ NN of $q$. Hence $\exists i \in \{1,\cdots,m\}, \delta(v,q_i) \le r^\star$ but $\forall i \in \{1,\cdots,m\}, \delta(w,q_i) > r^\star$. Since $w$ is one of any-$k$ NN and $v$ is not any-$k$ NN, there is a $r$ that $w \in \mathcal{R}_r^\exists$ and $v \notin \mathcal{R}_r^\exists$, which means $\exists q_i \in q, \delta(w,q_i) \le r$ but $\forall q_i \in q, \delta(v,q_i) > r$, which leads to a contradiction since no such $r$ exists compared to $r^\star$. □

**Theorem A.2:** *Given a dataset $D$ and a query $q = [q_1,\cdots,q_m]$, let $\mathcal{R}_i$ be the k-NN of query vector $q_i$, $\bar{\mathcal{R}} = \bigcup_{i=1}^m \mathcal{R}_i$, and $\bar{\mathcal{R}}_r^\forall = \{u \in \bar{\mathcal{R}} | \forall i \in \{1,\cdots,m\}, \delta(u,q_i) \le r\}$, denote $r^\star$ is the smallest value such that $\left|\bar{\mathcal{R}}_{r^\star}^\forall\right| \ge k$, we have $\bar{\mathcal{R}}_{r^\star}^\forall$ is the all-k of $q$ when $\bar{\mathcal{R}}_{r^\star}^\forall \subseteq \mathcal{R}_i$ holds.*

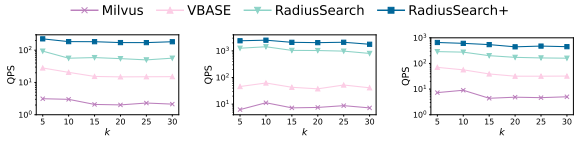
**Proof Sketch:** Assume that one vector $v \in \bar{\mathcal{R}}_{r^\star}^\forall$ is not all-$k$ NN, then there exist a vector $w \notin \bar{\mathcal{R}}_{r^\star}^\forall$ is one of all-$k$ NN. Since $\bar{\mathcal{R}}_{r^\star}^\forall \subseteq \mathcal{R}_i$ holds, we have $\forall i \in \{1,\cdots,m\}, \delta(v,q_i) \le r^\star$, but $\exists i \in \{1,\cdots,m\}, \delta(w,q_i) > r^\star$. Since $v$ is not all-$k$ NN and $w$ is all $k$ NN, there exist a $r$ that $\forall i \in \{1,\cdots,m\}, \delta(w,q_i) \le r$ but $\exists i \in \{1,\cdots,m\}, \delta(v,q_i) > r$. There is a contradiction in the value of $r$ and $r^\star$. □

# B Additional Experimental Results

**Statistics of Datasets.** The summary of datasets conducted in the experiments can be found in Table 1, with the number of dimensions (dim.), the number of vectors in the base dataset (#vectors), and the number of vectors provided for queries from the query dataset (#queries).

**Table 1: Statistics of Datasets**

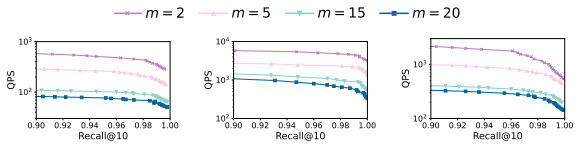
| Dataset | dim. | #vectors | #queries | Type |
|---|---|---|---|---|
| Recipe | 2,048 | 887,536 | 1,000 | Image |
| Msong | 420 | 992,272 | 200 | Audio |
| SIFT | 128 | 1,000,000 | 10,000 | Image |
| GIST | 960 | 1,000,000 | 1,000 | Image |
| Crawl | 300 | 1,989,995 | 10,000 | Text |




(a) Recipe (All-$k$ ANN) (b) Msong (All-$k$ ANN) (c) GIST (All-$k$ ANN)

(d) Recipe (Any-$k$ ANN) (e) Msong (Any-$k$ ANN) (f) GIST (Any-$k$ ANN)

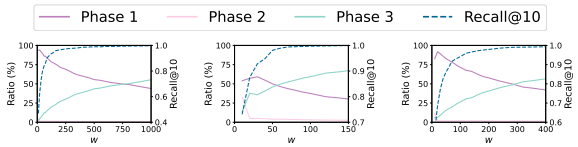
**Figure 10: QPS at Recall@$k$ = 0.99 When Varying $k$ (Exp. 3)**




(a) Recipe (All-$k$ ANN) (b) Msong (All-$k$ ANN) (c) GIST (All-$k$ ANN)

(d) Recipe (Any-$k$ ANN) (e) Msong (Any-$k$ ANN) (f) GIST (Any-$k$ ANN)

**Figure 11: Search Performance When Varying the $m$ (Exp. 4)**



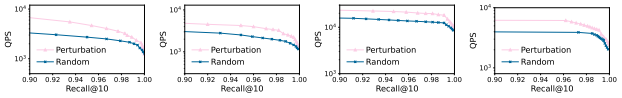

(a) Recipe (All-$k$ ANN) (b) Msong (All-$k$ ANN) (c) GIST (All-$k$ ANN)

(d) Recipe (Any-$k$ ANN) (e) Msong (Any-$k$ ANN) (f) GIST (Any-$k$ ANN)

**Figure 12: Time Decomposition of** RadiusSearch+ **(Exp. 5)**

**Exp. 6: Varying the Selection Strategy of Vectors in Each Query.** In this part, we evaluate the search performance of RadiusSearch+ by varying the vector selection strategy for each query. First, we generate query vectors by randomly selecting a

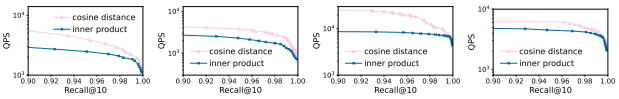
vector from the query vector dataset and then generating 5 distinct random noise perturbations of this vector as query vectors. This strategy is referred to as Perturbation. Second, we evaluate the strategy where each query vector is randomly chosen from the query vector datasets, labeled as Random. As depicted in Fig. 13, the Perturbation queries demonstrate even higher QPS performance compared to those generated using the selection strategy outlined in the "datasets" part. The Random approach showcases slightly slower yet still efficient results.




(a) SIFT (All) (b) Crawl (All) (c) SIFT (Any) (d) Crawl (Any)

**Figure 13: Varying Query Selection Strategy (Exp. 6)**

**Exp. 7: Extend to Other Distance Metrics.** In real applications, cosine similarity and inner product are two other commonly used distance metrics. Here, we extend our approach RadiusSearch+ to support these metrics, with an evaluation of its performance. Embedding normalization is a common practice, e.g., [30]; when embeddings are normalized, inner product, cosine distance, and Euclidean distance are equivalent. To further support non-normalized inner product, following [7], we adjust the pruning strategy for the index, i.e., modifying a small portion of edges in HNSW. The results of our RadiusSearch+ on cosine distance and inner product for SIFT and Crawl datasets are illustrated in Fig. 14, demonstrating its consistent performance across these distance metrics.




(a) SIFT (All) (b) Crawl (All) (c) SIFT (Any) (d) Crawl (Any)

**Figure 14: Performance on Other Distance Metrics (Exp. 7)**