Neural Computing and Applications (2024) 36:2303–2313
https://doi.org/10.1007/s00521-023-08920-3

S.I. : MACHINE LEARNING AND BIG DATA ANALYTICS FOR IOT SECURITY AND PRIVACY (SPIOT 2022)

Quantization to speedup approximate nearest neighbor search

Hao Peng[1,2]

Received: 15 March 2023 / Accepted: 24 July 2023 / Published online: 8 August 2023

Abstract

The quantization-based approaches not only are the effective methods for solving the problems of approximate nearest neighbor search, but also effectively reduce storage space. However, many quantization-based approaches usually employ fixed nprobes to the search process for each query. This will lead to extra query consumption. Additionally, we observed that as the number of points in each cluster center of product quantization increases, the query cost also increases. To address this issue, we propose an acceleration strategy based on the IVF-HNSW framework to further speed up the query process. This strategy involves introducing an adaptive termination condition for queries and reducing the number of data points accessed by building HNSW results. Through extensive experiments, we have shown that our proposed method significantly accelerates the nearest neighbor search process.

Keywords Approximate nearest neighbor search · Quantization-based approach · Termination conditions

1 Introduction

Nearest neighbor search (NNS) is a fundamental problem and has been played a significant effect to different fields, such as multimedia data mining, databases, machine learning and information retrieval. At the beginning, NNS can be solved effectively by linear scanning if the size of dataset is general. However, when the size of dataset increases in large, the performance of linear scanning is degenerated distinctly. Actually, someone does not require to find the exact nearest neighbors for many applications. Hence, the notion of approximate nearest neighbor search (ANNS) is raised to deal with the problem of query performance. To this end, many scholars spare no effort to develop various state-of-the-art algorithms for supporting a wonderful trade-off between query time and accuracy.

There is the objective condition that the memory space for a great number of algorithms building index structure is comparatively large. Quantization-based methods for ANNS are the most popular manner to decrease the memory consumption. They achieve that the original data vectors are quantified to the binary codes, such that the problem of high memory footprint could be addressed. Nevertheless, the prominent weakness is that quantization-based methods will bring some errors due to the quantization property, e.g., product quantization (PQ) [1]. To circumvent this drawback, some variations of PQ have been put forward to reduce the quantization error. In addition, quantization-based methods often rely on cluster algorithms to divide the original data space into different subspace. If the subspace divided is finely grained, it is conducive to rapidly obtain more accuracy query results, e.g. [2].

At present, due to the fact that according to the neighbor relationship of data points, graph-based methods can build effective indexing structure, they are fast ANNS algorithms and could reach up to high search accuracy with fewer time. Since graph-based methods have excellent query property for ANNS, they have already been employed into other algorithms. For example, HNSW [3] is adopted to build the indexing structure of coarse centroids for IVF-PQ

✉ Hao Peng
H.Peng7@newcastle.ac.uk

[1] School of Computing, Newcastle University, Newcastle NE1 7RU, UK

[2] School of Mathematics and Physics, Qingdao University of Science and Technology, Qingdao 266061, China

[1, 4]. For this case, one can rapidly obtain first nprobe centroids for each query. In addition, other variations of HNSW [3] have been proposed, and the focus is to optimize the neighbor relationship of data points by the selective connection of edges. The purpose is to further speedup query processing and also achieve higher query accuracy.

Due to the fact that IVF-PQ [1, 4] is a popular technique for indexing high-dimensional data that involves the generation of many coarse centroids based on the learned dataset. The original data is then quantized into different coarse centroids, and an inverted index is constructed according to the number of centroids. However, as the scale of the original data increases, the size of each centroid in the inverted list may also increase, leading to increased query cost if a greedy search algorithm is used to search for entries in these centroids. To address this issue, we propose using graph-based methods such as HNSW (Hierarchical Navigable Small World) to construct an indexing structure for those centroids containing many entries. This approach not only speeds up query processing but also allows higher query accuracy to be achieved with lower query cost. In addition, when a query arrives, IVF-PQ requires comparing the entries of the first nprobe centroids to return the approximate nearest neighbors. The value of nprobe is typically set as a uniform constant, which may not be optimal for all queries. To address this limitation, our proposed method learns different nprobe values for various queries, which enables effective acceleration of query processing and improves the overall performance of the system.

The main contributions of the paper are concluded as follows:

- We propose to build the small-scale various HNSW for those centroids containing large entries to further answer the ANNS. It is to verify that the small-scale various HNSW is more appropriate to enhance the query performance in comparison with the original method IVF-PQ.
- We propose the improved early termination method under the framework of IVF-HNSW to predict the termination condition. This method can effectively give the number of nprobe for each query. Also, the training process of the method is simpler compared with [5].
- Extensive experiments are conducted to show that the proposal implements preferable search performance for different real datasets with various scales. In comparison with IVF-HNSW, it can reach up to at least 30% acceleration with respect to the index of time performance verified by a variety of datasets.

The rest of this paper is organized as follows. Section 2 introduces the related work. Some preliminaries are reviewed in Sect. 3. The learned early termination method is introduced in Sect. 4. Experimental results and analysis are reported in Sect. 5. Finally, a conclusion is made in Sect. 6.

2 Related works

Approximate nearest neighbor search (ANNS) always is the study hotspots of scholars in the recent twenty years. There are many works to find an effective method solving the ANNS problem.

Initially, space partitioning methods for tree structure usually [6–9] divide the entire data space into multiple regions with a recursive manner, such that the index can be constructed by a tree or a forest. According to this partition strategy, they are divided into two classes of structures, i.e., hierarchical structures and reference point-based structures. Although these methods are effective as the data dimension is low, their performance sharply reduced, when the data dimension increases obviously. If the dimensionality of datasets is more than 10, the efficiency of space partitioning methods for tree structure is slower than the brute-force linear-scan method. To this end, many methods to address the problem of "curse of dimensionality" have been proposed, e.g., the pyramid technique [10] and iDistance [11].

Hash-based methods were primordially proposed to address the problem of ANNS under Hamming space [12], and then was extended to Euclidean space [13]. The essence is to project the high-dimension data points as low-dimension data by a family of hash function, so that one can devise an effective method to find the nearest neighbors in the projection space. Moreover, hash-based methods often need to build a great number of hash tables to implement satisfactory search results. Hence, many scholars had proposed many hash-based variants, e.g., Multi-Probe LSH [14], QALSH [15], VHP [16], LCCS-LSH [17], PMLSH [18] and R2SLH [19].

Graph-based methods [20–23] are effective for high-dimensional ANNS than other methods. The reason is that Graph-based methods can build an effective indexing structure. However, the prominent weakness is that Graph-based methods need to construct exact *K*NN graph, whose construction complexity presents exponential increment with the amount of nodes increasing. To solve this problem, the approximated *K*NN graph has been studied. Actually, the effectiveness of approximated *K*NN graph equals to the exact one in certain application. Nevertheless, the construction of Graph is limited to memory. If we want to construct billion-scale *K*NN graph, then the space consumption is unacceptable. To achieve more effective query, the original data is addressed by cluster algorithms for building simpler subgraph, such that query processing can be conducted in sparse subgraph [24]. As machine learning and deep learning have developed, scholars have naturally thought of applying deep learning to graph-based nearest neighbor search. In "Path Learning in Similarity Graphs for Nearest Neighbor Search", [25] proposed using greedy paths to execute search on similarity graphs through path learning in similarity graphs. Spatial partitioning is a fundamental technique for fast nearest neighbor search algorithms, and a new framework for constructing spatial partitions was proposed in [26]. This framework simplifies the problem into balanced graph partitioning and then performs supervised classification. [27] introduced a new principled algorithm based on optimizing adjacency matrices, which significantly improves search efficiency. Specifically, they proposed a probability model of the similarity graph defined by its edge probabilities and demonstrated how to learn these probabilities as a reinforcement learning task from data. These algorithms can effectively solve nearest neighbor search problems. Recently, scholars [28, 29] develop the graph structures by learning methods for better denoting the neighbor relationship of data points. However, these methods are performed only in single core machine, [30] thus extends it to GPU, which is to exploit the hardware resources adequately. Actually, the purpose of these methods is to speedup the query processing.

Product quantization (PQ) [1] is regarded as the most common of quantization-based methods, which plays an important role in accelerating search processing. PQ provides several attractive properties: (1) PQ compresses an input vector into a short code (e.g., 64-bits), such that the billion-scale dataset processed in memory is possible; (2) PQ develops effective manners (ADC and SDC) to compute the approximate distance between a raw vector and a compressed PQ code, which is a good estimation of the original Euclidean distance; (3) the data structure and coding algorithm are simple, which allow it to hybridize with other indexing structures. Since the quantization of PQ for the original vectors results in certain error influencing the accuracy of distance calculation, a great number of quantization-based approaches have been put forward. The purpose is to further reduce the quantization error for obtain more accuracy query results, e.g. [31–34].

3 Preliminaries

3.1 Problem definition

For a high-dimensional data space with n points and the dimension of each point being d, any two points are selected in the data space. Then the Euclidean distance for the two points $X = (x_1, ..., x_d)$ and $Y = (y_1, ..., y_d) \in \mathbb{R}^d$ is represented as $d(X, Y) = \sqrt{\sum_{i=1}^{d} (x_i - y_i)^2}$. When a query point q and the distance index $d(\cdot, \cdot)$ are given, it is naturally to define the problem of the following form:

Definition of nearest neighbor search. If the distance measure $d(\cdot, \cdot)$ and any given query are determined, the problem of the nearest neighbor search is to answer a point o^* in a data space that has the minimal distance to the query by building data structure in advance.

Definition of approximate nearest search 2. If the distance measure $d(\cdot, \cdot)$ and any given query q are determined, the problem of the approximate nearest neighbor search is to answer a point o with $d(o, q) \leq (1 + \varepsilon)d(o^*, q)(\forall \varepsilon > 0)$ in the data space by building data structure in advance. In addition, Approximate k Nearest Neighbor Search is to answer k points with $d(o_i, q) \leq (1 + \varepsilon)d(o_i^*, q)(i \in \{1, 2, ..., k\})$.

3.2 Product quantization and inverted index

The basic idea behind Product Quantization (PQ) is to split a high-dimensional vector into multiple sub-vectors and quantize each sub-vector separately. This is done by applying K-means clustering on each sub-vector to obtain a codebook or dictionary of centroids. The original vector is then quantized by finding the closest centroid in each sub-vector's codebook and replacing it with the index of the corresponding centroid. This process results in a compact representation of the high-dimensional vectors by encoding them as sequences of indices of the centroids. More specifically, PQ [1] first partitions data space with d-dimension into m different subspaces, and the dimension of each subspace is $\frac{d}{m}$. After that, a part of original vectors is selected and partitioned into $\frac{d}{m}$ dimensions. Then the cluster algorithm k-means is used to learn various k centroids in the subspace, such that each subvector is quantified into corresponding centroid. Namely each subvector is represented by a PQ-code, where the PQ-code is an integer between $[1, k]$. So that one can use a tuple of m PQ-code to denote the entire code of the original vectors. Finally, the codebook with size of $m \cdot k$ centroids is set up with respect to this product quantizer.

For IVF-PQ, the original vectors in the data space are divided into different centroids by k-means, where each centroid contains some original vectors. Then the difference between the original vectors and its centroid to obtain the residual vectors. After that, PQ is conducted with respect to the residual vectors to form PQ-code. Finally, an inverted index is constructed, which is the combination of centroids and PQ-code.

3.3 Graph-based methods

The indexing structure of graph-based methods [3, 20, 21, 35] contains nodes and edges, in which each node denotes a point of dataset and each edge denotes the neighbor relationship between any two points. The structure can effectively retain the neighbor relation among nodes. One of the state-of-the-art graph-based methods is the hierarchical navigable small world graphs (HNSW) [3]. Its indexing structure contains multiple layers of small world graphs [35], which is illustrated in Fig. 1. More specifically, the top and base layer contain single vector and all database vectors, respectively, while the intermediate layer contains a subset of database vectors in the next layer. When constructing the index, database vectors are inserted one by one, where each vector is inserted into multiple layers form the base layer to a certain layer determined by an exponentially decaying probability distribution. For a newly vector arrived, it is connected to some nearest nodes that are at most permitted to connect a fixed number of vectors. When a query arrives, the search is started in the top layer to find its nearest node, and then this node is regarded as the started point to perform search processing in the next layer. If this process is converged to the base layer, then a nearest point of query is returned.

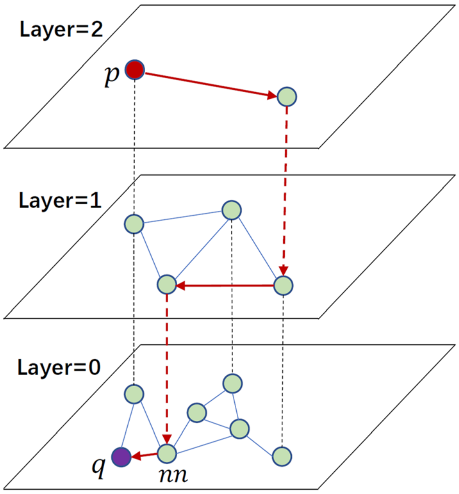


Fig. 1 A three-layer HNSW index. The search starts from the red circle of the top layer. The red arrow indicates the greedy search path. The search is terminated until finding the nearest neighbor of q in the 0-th layer

Graph-based methods are important in HNSW-IVF because they enable efficient nearest neighbor search in high-dimensional spaces. The graph structure provides a way to navigate through the data and find nearby points without having to calculate distances between all pairs of data points. Instead, the search algorithm can follow the edges of the graph to quickly traverse the space and locate the nearest neighbors. In the paper, it explains that the use of a graph structure allows for efficient indexing of large datasets. It also shows how the graph-based approach can be combined with an inverted index to further improve search performance. By dividing the dataset into smaller subspaces and building separate indexes for each subspace, the search process can be accelerated even further. Overall, graph-based methods are a key component of the HNSW-IVF approach, enabling fast and accurate nearest neighbor search in high-dimensional spaces.

4 The learned early termination under IVF-HNSW

4.1 Weakness on IVF-HNSW

Although IVF-HNSW uses HNSW to speed up the approximate nearest neighbor, it is only focusing on the partition of original data space, i.e., k-means is employed to partition the original data space to different centroids and then HNSW is built with these centroids. However, IVF-HNSW needs to search its top-k centroids for each query. That is, the algorithm is terminated as the fixed nprobe has been conducted. So, the method of learned early termination for nprobe is used. In addition, since the dataset is mapped to different centroids, it is possible that one centroid contains many original data. If one directly checks those points, then it will increase extra query cost. To this end, HNSW is constricted to accelerate the query process.

4.2 Algorithm description

```
Algorithm 1: Offline Training (K-means and prediction model)
        K-means.
        Input: train set of the base dataset: train_s,
        test set of the base dataset: test_s,
        number of k-means cluster centroids: $k$
    Output: $k$ cluster centroids
    Initialize Num. cluster centroids be queried: $n$
    Initialize Num. features: $f$
    for each vector $v_i$ in train_s, $v_j$ in test_s do
        for $m=1$ to $n$ do
            $d$ ( $v_i$ , $m$ th nearest centroid),
            $d$ ( $v_j$ , $m$ th nearest centroid)
        end
        for $i_1=1$ to $f$ do
            $r_{i_1}^{i}$ = $d$ ( $v_i$ , $5i_1$ th nearest centroid) /
            $d$ ( $v_i$ , 1st nearest centroid),
            $r_{i_1}^{j}$ = $d$ ( $v_j$ , $5i_1$ th nearest centroid) /
            $d$ ( $v_j$ , 1st nearest centroid),
        end
    end
      nprobe evaluations: $n\_p$ , perform actual ANNS until find the ground truth,
      Num. $n\_p$ for each $v_i$ in train_s,
      Num. $n\_p$ for each $v_j$ in test_s
      Training.
    Input: training feature: $r_{i_1}^{i}$ , where $i_1=1$ to $f$ ,
        $n\_p$ for each $v_i$ in train_s,
        testing feature: $r_{i_1}^{j}$ , where $i_1=1$ to $f$ ,
        $n\_p$ for each $v_j$ in test_s,
    return prediction model
```

In this section, we provide a detailed description of how to use k-means to obtain the input features for our model and train a prediction model that can give termination conditions for different queries. In addition, we address the issue of greedy search during query processing in product quantization, especially when centroids contain many points. To reduce the number of points accessed during query processing, we propose using an HNSW structure specifically for these clusters. This approach allows us to achieve results comparable to the original method by setting a fixed number of access points. By doing so, we can significantly reduce the query cost and improve the overall efficiency of the system. This process is presented in Fig. 2.

Our approach is implemented by the framework IVF-HNSW. Its construction contains two fractions, which are indexing process and query process. Meanwhile, the early termination method to predict the number of nprobe includes the offline training and online prediction.

Indexing process. First, the original data space is partitioned to different subspaces by k-means. When one data is mapped to a centroid, the difference between the original data and the centroid is computed to obtain the residual vector. With these residual vectors, Product Quantization is exploited to generate pq-codes. Moreover, if each centroid containing many points is large than given threshold, then HNSW is constructed for these points in the centroid.

To accelerate the query processing, an early termination method is used, which is the combination of offline training and prediction. The concrete process is described as follows:

(1) *Offline training*. The offline training process includes both K-means and prediction model, and the pseudo-code is presented in Algorithm 1. Due to the fact that the size of training set impacts the training time and result, selecting appropriate size of training set is particularly important. To this end, a tenth of the original vectors are randomly selected as the training set. Also, it does not exceed one million regardless of the scale of the original vectors. Additionally, 10 thousand original vectors are selected as the testing set.

Inspired by [5], if the feature of the learning termination condition is the nearest neighbor of the query, the learning prediction model can effectively predict the termination condition of the query, that is, quickly find the nearest neighbor. This reveals that the performance of evaluating the prediction model is whether the nearest neighbor of query has been found as the input feature. In order to improve the efficiency and accuracy of the prediction model, we use K-means as an effective method for extracting the input feature. The main consideration is to approximate the nearest neighbor feature of the query through the distance of K-means centroids. While K-means

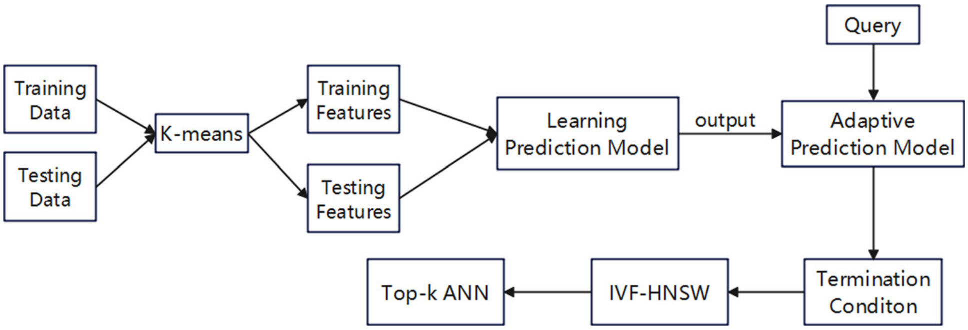


Fig. 2 Process of proposal

is unable to exactly ensure the nearest neighbor contained in the first centroid, so we select many centroids. The advantage is that K-means does not need to determine the nearest neighbor, which is only an alternative by a centroid, such that the input feature is obtained quickly. Hence, K-means [36] is performed to output k centroids by the training set. After that, for each vector in the training and testing set, the distance between the one and its m-th centroid and the distance ratios between the one and its 5i-th centroid $(i \in \{1, 2, ..., 10\})$ are calculated (Line 4–12), where the distance ratios are the input features for the prediction model. Note that we may be select the first 30 centroids to obtain the distance ratios when the true nprobe is comparatively small. Then ANNS for each vector in the training and testing set is conducted on IVF-HNSW to collect the number of nprobe (Line 13–15). Finally, some features are used to predict the model (Line 16–17).

The inputs. Compared with the adaptive online early termination method [5], the complex runtime features are not necessary in the offline method, in which the features required are a relatively small constant. Hence, the number of the input features are lower, which are shown in Table 1. As discussed above, K-means has been exploited to generate the input features for the prediction model. If the queries are allocated into the same cluster, then the Euclidean distance between the queries and centroid is similar. This means that the centroid is most important feature to the queries. The main focus is that inspired by [5], the importance of features for any given query is close to their neighbors. Also, most of query do not need to conduct large nprobe. Namely, many query could find their nearest neighbor in the near centroids. Therefore, we propose that an alternative for the feature of various queries is the first coarse centroid, and its neighbors are other coarse centroids. Then the distance ratios between the first coarse centroid and others are denoted as the input features. The process is to generate some approximated features in advance, such that the prediction model can be performed offline. In fact, the number of features required by the model varies. The reason for using 50 centroids as features is that the nearest neighbor of the query can generally be found in the first 50 clustering centers, and this number of features is enough to train a good model. If the number of centroids is increased, it will increase the training time of the model. This approach is aimed at obtaining a good model while reducing the training time required.

The output. The learned model that is similar to the housing price prediction model is employed to predict the nprobe for any given query. This output is not the value of termination condition, while they are closely dependent. The relation is presented in the part of prediction and search.

Training. To obtain uniformed training model, the search is first performed under the framework of IVF-HNSW to find ANN for some given queries, in which the primary objective is to find their exact nearest neighbors. If the one has been found, then it is nature to yield the number of nprobe in the search process. Due to the fact that the quantization property of indexing construction for IVF-HNSW, it is unable to report the exact nearest neighbors for certain queries. Then these queries should be neglected in the training procedure. With the input features and the number of nprobe, the training model can be generated with offline manner.

In sum up, the training procedure of proposal is simpler compared against the online learning procedure [5]. This method is different from the data structure faced by [5], and it is difficult to compare these two methods in this paper. In fact, the termination conditions for training queries using two different learning models are similar, but the method used in this paper is more suitable for nonlinear prediction. Furthermore, when the offline training procedure is terminated, the number of nprobe has already been obtained in advance, such that online prediction generating many runtime features can be circumvented. These are the distinguished difference for the one in the online learning procedure.

Table 1 Input features

Feature	Description	Num. of features
F_i: $i$th_cc_to_1th cc	$d(q,i$th nearest coarse cluster centroid$)/d(q,1$th nearest coarse cluster centroid$)$ where $i \in \{5, 10, 15, ..., 45, 50\}$	10

Algorithm 2: Prediction and Query Process

```
Input:   query vector: $q$ , $F_i$ : $i$th_cc_to_1th cc ($i \in \{5,10,15,...,45,50\}$)
              number of neighbors to return: $k$ ,
              threshold: $r$ , IVF-HNSW index:
$IVF-HNSW$
Output:  top-$k$ nearest neighbors
$n\_p \leftarrow$predict ( $q$ )
$mh \leftarrow$ empty max heap with size $k$
$Cs \leftarrow$ HNSW to find $n\_p$ centroids of $q$
$Tab \leftarrow$ precompute distance table

for each $i = 1$ to $n\_p$ do
    N ← compute the entry number of $i$-th centroid
    if $N > r$ then
        search in HNSW to find candidates
        compute the distance between query and candidates
        store the distance in $mh$
    end
    else
        greedy search in the current centroid
        compute the distance between query and entry by $Tab$
        store the distance in $mh$
    end
end
return top-$k$ nearest neighbors in $mh$
```

(2) *Prediction and query process*. With the prediction model above, the number of nprobe for any given query can be predicted effectively. The pseudo-code of prediction and search is illustrated in Algorithm 2. When a query arrives, the number of nprobe denoted as $\hat{p}$ is obtained by the prediction model. The number of nprobe is designed to be $p = c \times 2^{\hat{p}}$, which is regarded as the termination condition (Line 1), where c is a constant and its role is to stretch or shrink $\hat{p}$ for better performing search procedure. In comparison with [5], we found that [5] design a binary search to determine the nprobe for total queries, when the target accuracy is given. But there exists a drawback, namely some queries might share common nprobe as it conducts to stretch or shrink $\hat{p}$. Hence, we hope that each query should be different for obtaining the nprobe during the binary search. Next, our method first judge whether or not the entries of current centroid are higher than given threshold r, where r is an important constant for the proposal. Assume that the number of centroids is set to be K, then the average of each centroid is *n/K*. In practice, as the number of points in each centroid is large than *t(n/K)*, the index for these points is constructed by HNSW, thereby the threshold r is designed as *t(n/K)*, where t is an user-specified positive integer. In this setting, the main focus is to consider those centroids that contain many points. If there exists at least *t(n/K)* points quantized in the current centroid, then the search is conducted in the HNSW (Line 6–10). Otherwise, the greedy search is conducted in the current centroid on the framework of IVF-HNSW (Line 11–14). Finally, top-k approximate nearest neighbors are returned (Line 15).

5 Experiments

5.1 Datasets

In order to validate the reasonability of the proposal, we select some real-life datasets used to the database in the experiments. For the queries, we also select the query set given by the database to be our query. The corresponding statistics of datasets and queries are summarized in Table 2. Note that the proposal is also extended to address large-scale data, where the experiments are conducted in the regular machine with memory 32 GB.

- *SIFT 10 M*. This dataset is randomly selected from SIFT 1 billion, where the total number of points selected are 10 million, and the dimension of each point is 128.
- *DEEP 10 M*. This dataset is randomly selected from DEEP 1 billion, where the total number of points selected are 10 million, and the dimension of each point is 96.
- *SPACEV 10 M*. The total number of SPACEV dataset is 10 million data points, and the dimension of each data point is 100.
- *TURING 10 M*. The total number of Turing dataset is 10 million data points, and the dimension of each data point is 100.
- *GIST*. The GIST is 1 million data points with 960-dimensions image extracted from handwritten numbers.

5.2 Evaluation metric

In this paper, the two metrices are used to further verify the effectiveness of our proposal.

- *Recall*. First, recall is used as an evaluation index of accuracy to measure whether or not the search results returned are competitive. For the k-NNS of our algorithm reported, we define the recall as the ratio on how many the k points returned by our methods are contained in the one array that has k true nearest neighbors for a given query. Hence, recall can be naturally generated by the following form:

$$\text{Recall} = \frac{|R' \cap R|}{|R|}$$

In which R' is a set that contains k points reported by our method with respect to a given query, while R is

Table 2 Summary of datasets

Dataset	Vector dimension	Num. base vectors	Num. learned vectors	Num. query vectors
DEEP	96	10 M	1 M	10,000
SPACEV	100	10 M	1 M	10,000
SIFT	128	10 M	1 M	10,000
TURING	100	10 M	1 M	10,000
GIST	960	1 M	0.5 M	1000

also a set that contains k true nearest neighbors with respect to the given query. For simplicity, here Recall is evaluated by $R1@100$, which is shown as recall@100 below.

- *Query Answering Time*. The average answering time is another evaluation metrics to measure our search performance. It is often denoted the wall-clock time of the computer for the algorithm to return query results. In order to obtain more accuracy time for our algorithm, the average value for multiple recall and answering time is used in our experiments. Namely, the average values are at least more than 5 times.

5.3 Baseline method

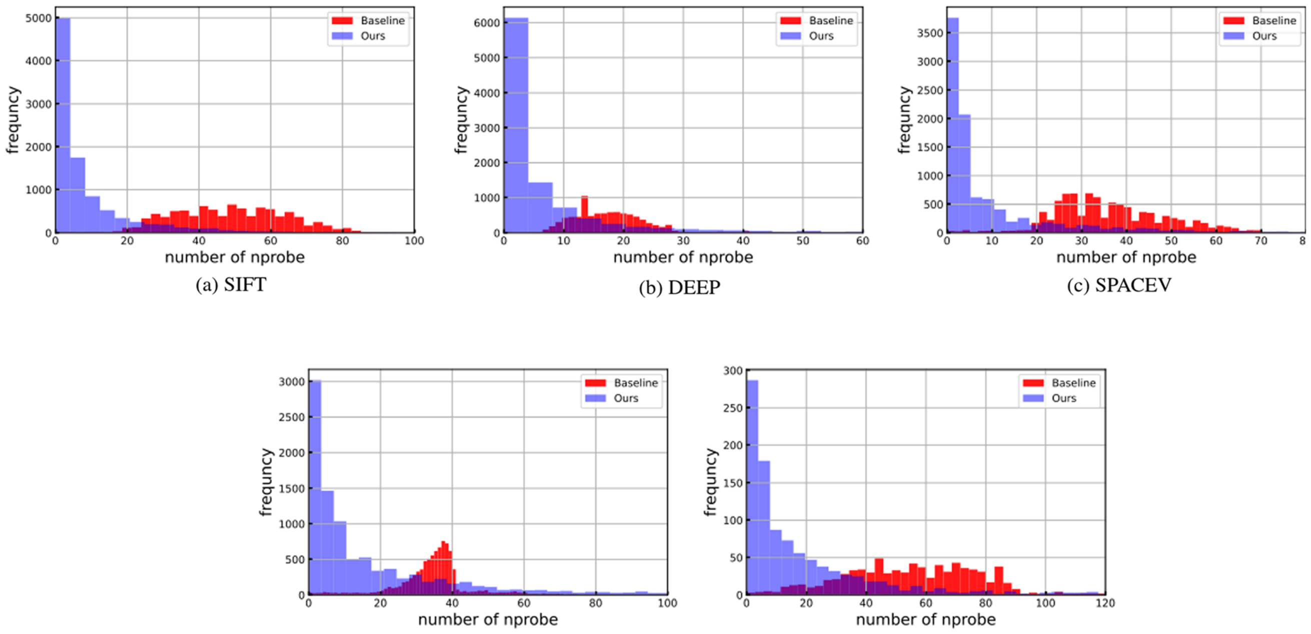
In practice, there exists a great number of the stats-of-the-art algorithms for ANNS, such as PQ [1], OPQ [31], PQBF [37] and IVF-HNSW. Since IVF-HNSW performs better for ANNS compared against PQ, OPQ and PQBF [2], we choose IVF-HNSW as the baseline for comparison. To achieve the best search performance of IVF-HNSW, we set some parameters described in [2]. Readers can refer to [2] for more details. Also, to make a fair comparison, the experiment in our method is same as IVF-HNSW. The purpose is to reveal that based on the framework of IVF-HNSW, our method with the adaptive early termination method is superior than IVF-HNSW. Although there exist many other methods for ANNS, we mainly focus on the quantization-based methods for comparison because of the quantization property of indexing structure of IVF-HNSW. Also, it should be note that IVF-HNSW do not conduct any pruning process.

5.4 Results and analysis

The main purpose of this paper is to accelerate nearest neighbor search based on the IVF-HNSW framework, targeting the problems of uniform query termination conditions and a large number of data points in each centroid. Such problems can be verified by experimental methods to determine their effectiveness. The experimental results are presented later. Based on these results, our proposed method outperforms the baseline method. Due to the fact that our algorithm is achieved in the framework of IVF-HNSW, the indexing consumption contains the cost that is same as IVF-HNSW and the one of building HNSW for some centroids with many quantized points. Since the index cost of HNSW is *O(mn)*, where m is a user-specified constant and n is the number points for index. In the experiments, as the number of points in w centroids is larger than *t(n/K)*, in which w is a data-dependent constant. Hence the additional index cost for the algorithm is *O(wmn)*. Here we mainly analyze the trade-off between the running time and accuracy.

Next, we will analyze why our method can effectively speedup query processing. On the one hand, our method is designed to learn the numbers of nprobe for different queries, such that the uniformed nprobe for each query is avoided, which negatively impacts the query cost. The reason is that the approximate nearest neighbors of many queries can be found as the number of nprobe is small. On the other hand, when the number of coarse centroids is set as 2^{14}, then the average number of entries in each centroid is $\frac{n}{2^{14}}$, where n is the number of original data points. As n increases, the average will be large. For this case, the total number of entries in the original methods is checked to be $nprobe \times \frac{n}{2^{14}}$. Then it will need to check a great number of entries, thereby the query burden is increased. To verify the effectiveness of our method, the numbers of nprobe for each query are given, where the statistics are collected when high accuracy is achieved, as illustrated in Fig. 3. One can be found that the numbers of nprobe for most of queries are small, in comparison with the baseline [5]. Also, the numbers of nprobe for many queries are lower than 20. It is the reason why our method can speed up query processing, i.e., our method could check fewer points.

In addition, we conduct many experiments over different datasets to reveal the superiority of our method, and the results are shown in Fig. 4. It can be observed that compared with the baseline, our method achieves more higher query performance as the common accuracy is obtained. Since our method can learn more superior nprob for each query. Also, it is accelerated by the HNSW construction for some centroids containing many points. Hence, the query performance is effectively enhanced. Actually, our method to learn the number of nprobe relies on the features of



(a) SIFT

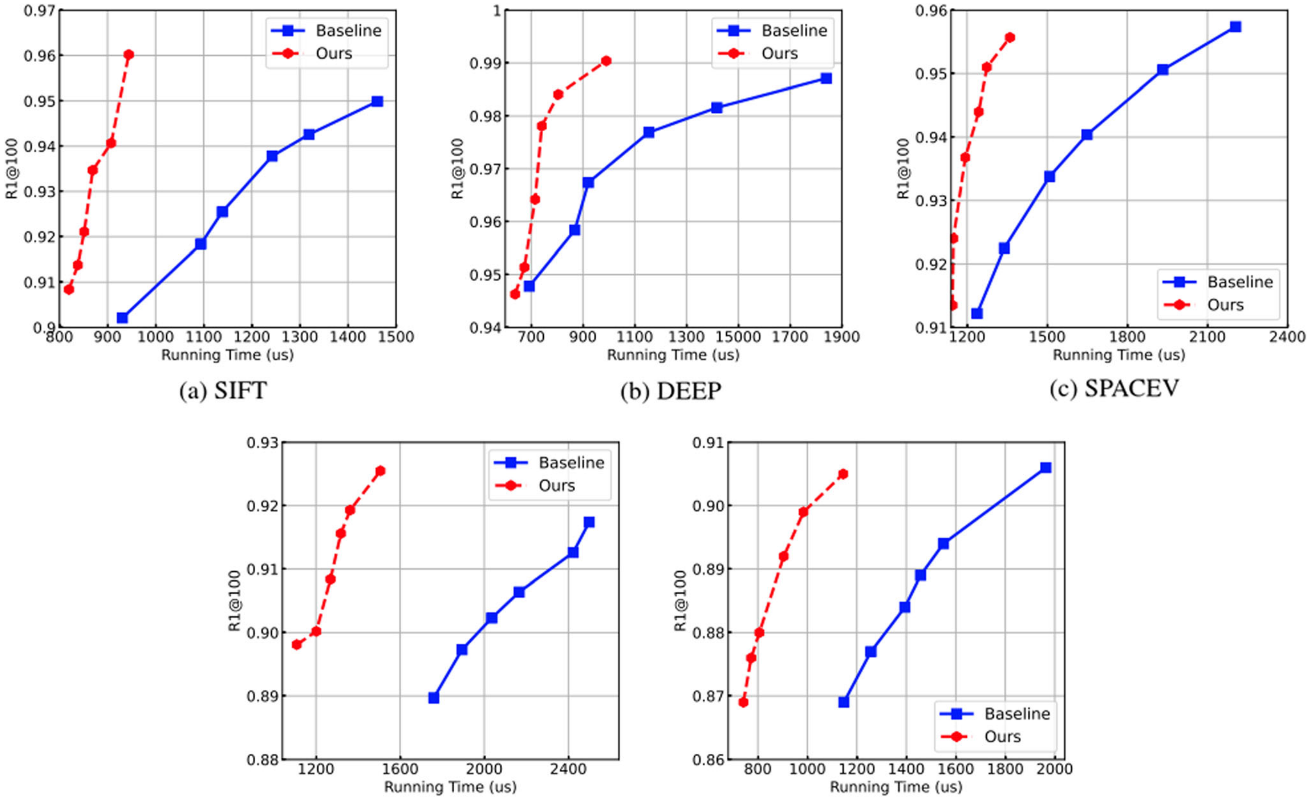
(b) DEEP

(c) SPACEV

(d) TURING

(e) GIST

Fig. 3 Histogram on numbers of nprobe for queries



(a) SIFT

(b) DEEP

(c) SPACEV

(d) TURING

(e) GIST

Fig. 4 Average time versus recall@100

k-means. If one can propose other method to learn the numbers of nprobe for each query, then the query performance will be effectively improved.

6 Conclusion

In this paper, we propose an efficient method to solve the problem of approximate nearest neighbor search. We employ the learned method the predict the numbers of nprobe for each query and also speedup the query processing by building HNSW for some centroids containing many entries. We conduct many experiments to validate the effectiveness of our method by different datasets, and the results illustrate that our method has prominent superiority than other methods as the same search index is set. Approximate nearest neighbor search is widely used in practice, yet it still faces challenges when dealing with large datasets and slow query processing. Our goal is to speed up this process and meet practical demands. The research presented in this paper is an extension of existing nearest neighbor research and represents a way to enrich current technologies. We aim to provide a more efficient and accurate approach for nearest neighbor search that can be applied in various fields, such as image recognition, recommendation systems and natural language processing. By improving the performance of these systems, we hope to contribute to the development of advanced technologies and benefit society as a whole. In addition, the main research challenge of this paper is how to obtain effective input features, as the quality of input features directly affects the predictive ability of the learning model. Therefore, our future work will focus on finding better quality input features to improve the training effectiveness of the model. In experiments, we have found that having more query nearest neighbors as features leads to better predictive performance of the model. Therefore, we can explore ways to find better query nearest neighbors as input features.

Data availability statement This article does not cover data research. No data were used to support this study.

Declarations

Conflict of interest These no potential competing interests in our paper. And all authors have seen the manuscript and approved to submit to your journal. We confirm that the content of the manuscript has not been published or submitted for publication elsewhere.

References

1. Jegou H, Douze M, Schmid C (2010) Product quantization for nearest neighbor search. IEEE Trans Pattern Anal Mach Intell 33(1):117–128
2. Baranchuk D, Babenko A, Malkov Y (2018) Revisting the inverted indices for billion-scale approximate nearest neighbors. In: Proceedings of the ECCV, pp 202–216
3. Malkov Y-A, Yashunin D-A (2018) Efcient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Trans Pattern Anal Mach Intell 42(4):823–836
4. Sivic J, Zisserman A (2003) Video Google: a text retrieval approach to object matching in videos. In: Proceedings of the CVPR, pp 1470–1477
5. Li C, Zhang M, Andersen D-G, He, Y (2020) Improving approximate nearest neighbor search through learned adaptive early termination. In: Proceedings of the ACM SIGMOD, pp 2539–2554
6. Bentley J-L (1975) Multidimensional binary search trees used for associative searching. ACM Commun 18(9):509–517
7. Yianilos P-N (1993) Data structures and algorithms for nearest neighbor search in general metric spaces, pp 311–323
8. Guttmann R (1984) A dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD, pp 47–57
9. Sebastian TB, Kimia BB (2002) Metric-based shape retrieval in large databases. In: Proceedings of the pattern recognition, pp. 291–296
10. Chen L, Gao Y, Li X, Jensen C, Chen G (2017) Efficient metric indexing for similarity search. IEEE Trans Knowl Data Eng 556–571
11. Jagadish H, Ooi B, Tan K, Yu C, Zhang R (2005) iDistance: an adaptive B+-tree based indexing method for nearest neighbor search. ACM Trans Database Syst 30(2):361–397
12. Gionis A, Indyk P, Motwani R (1999) Similarity search in high dimensions via hashing. In: Proceedings of the VLDB, pp 518–529
13. Datar M, Immorlica N, Indyk P, Mirrokni V (2004) Locality-sensitive hashing scheme based on p-stable distributions. In: Proceedings of the SCG, pp 253–262
14. Lv Q, Josephson W, Wang Z, Charikar M, Li K (2007) Multi-probe LSH: Efficient indexing for high-dimensional similarity search. In: Proceedings of the VLDB, pp 950–961
15. Huang Q, Feng J, Zhang Y, Fang Q, Ng W (2015) Query-aware locality-sensitive hashing for approximate nearest neighbor search. In: Proceedings of the VLDB, pp 1–12
16. Lu K, Wang H, Wang W, Kudo M (2020) VHP: approximate nearest neighbor search via virtual hypersphere partitioning. In: Proceedings of the VLDB, pp 1443–1455
17. Lei Y, Huang Q, Kankanhalli M et al (2020) Locality-sensitive hashing scheme based on longest circular co-substring. In: Proceedings of the SIGMOD, pp 2589–2599

18. Zheng B, Xi Z, Weng L et al (2020) PM-LSH: a fast and accurate LSH framework for high-dimensional approximate NN search. In: Proceedings of the VLDB, pp 643–655
19. Lu K, Kudo M (2020) R2LSH: A nearest neighbor search scheme based on two-dimensional projected spaces. In: Proceedings of the ICDE, pp 1045–1056
20. Fu C, Xiang C, Wang C-X, Cai D (2019) Fast Approximate nearest neighbor search with the navigating spreadingout graph. In: Proceedings of the VLDB, pp 461–474
21. Fu C, Cai D (2016) Efanna: an extremely fast approximate nearest neighbor search algorithm based on knn graph. arXiv:1609.07228
22. KGraph. https://github.com/aaalgo/kgraph
23. Li W, Zhang Y, Sun Y, Wang W, Li M, Zhang W, Lin X (2019) Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. IEEE Trans Knowl Data Eng 32(8):1475–1488
24. Gallego AJ, Rico-Juan JR, Valero-Mas JJ (2022) Efficient k-nearest neighbor search based on clustering and adaptive k values. Pattern Recognit 122:108356
25. Baranchuk D, Persiyanov D, Sinitsin A et al (2019) Learning to route in similarity graphs. In: Proceedings of the ICML, pp 475–484
26. Dong Y, Indyk P, Razenshteyn I et al (2019) Learning space partitions for nearest neighbor search. In: Proceedings of the ICLR, pp 1–20
27. Baranchuk D, Babenko A (2019) Towards similarity graphs constructed by deep reinforcement learning. In: Proceedings of the CoRR
28. Lee N, Lee J, Park C (2022) Augmentation-free self-supervised learning on graphs. In: Proceedings of the AAAI conference on artificial intelligence, pp 7372–7380
29. Oyamada RS, Shimomura LC, Barbon S Jr et al (2023) A meta-learning configuration framework for graph-based similarity search indexes. Inf Syst 112:102123
30. Groh F et al (2022) Ggnn: graph-based gpu nearest neighbor search. IEEE Trans Big Data 9(1):267–279
31. Ge T, He K, Ke Q, Sun J (2013) Optimized product quantization. IEEE Trans Pattern Anal Mach Intell 36(4):744–755
32. Babenko A, Lempitsky V (2014) Additive quantization for extreme vector compression. In: Proceedings of the CVPR, pp 931–938
33. Babenko A, Lempitsky V (2015) Tree quantization for large-scale similarity search and classification. In: Proceedings of the CVPR, pp 4240–4248
34. Zhang T, Du C, Wang J (2014) Composite quantization for approximate nearest neighbor search. In: Proceedings of the ICML, pp 838–846
35. Malkov Y, Ponomarenko A, Logvinov A, Krylov V (2014) Approximate nearest neighbor algorithm based on navigable small world graphs. Inf Syst 45:61–68
36. Nister D, Stewenius H (2006) Scalable recognition with a vocabulary tree. In: Proceedings of the CVPR, pp 2161–2168
37. Arora A, Sinha S, Kumar P, Bhattacharya A (2018) Hd-index: pushing the scalability-accuracy boundary for approximate knn search in high-dimensional spaces. In: Proceedings of the VLDB, pp 906–919

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.