Density Peak Hashing

Zhen Wang [#1], Fuzhen Sun [#2], Baomin Shao [#3]

[#] *School of Computer Science and Technology, Shandong University of Technology*
Zibo, Shandong Province, China

Abstract

Hashing algorithms can map the floating-point data into compact binary code, and it can fast respond to the ANN search task according to the Hamming distance. The main idea of hashing algorithms is to cluster the data points into different groups and assign binary codes. Generally, many existing hashing algorithms adopt the K-means clustering algorithm to divide the data points into different clusters. K-means clustering algorithm learns the clustering results according to the Euclidean distances among data points, and clusters the data points with small distances to a center into the same group. Therefore, the K-means clustering algorithm is not applicable to the data with the non-spherical distribution. To solve this problem, this paper proposes to compute the clustering groups based on density peaks and learns the binary codes according to the obtained cluster groups, which can make the encoding results well adaptive to data distribution. Furthermore, a two-step mechanism is adopted to learn the linear hashing functions which can recompute the above binary encoding results. To effectively reduce the training time complexity in this paper, only cluster centers are involved in the training process. While learning the hashing functions, the cluster centers' binary codes are demanded to preserve the Hamming space's Euclidean similarity relationship. Thus, the data pairs' Hamming distances can approximate their Euclidean distance. The comparative ANN search experiments in three image datasets show that the proposed density peak hashing (DPH) can achieve an excellent performance.

Keywords **—***Hashing algorithms, Approximate nearest neighbor search, Density peak, Binary code*

I. INTRODUCTION

Generally, the approximate nearest neighbor (ANN) search task is achieved according to the Euclidean distance among floating-point data. For example, the image content is usually represented as a high dimensional feature descriptor (such as GIST and SIFT). The Euclidean distances among these feature descriptors are utilized to fulfill the image search task. However, the time complexity of Euclidean computing distance is too high, and it's difficult to fast respond to large scale ANN search tasks. In addition, with the development of Internet technology, the number of images is rapidly increasing. Thus, how to fast respond to the ANN search task becomes a hot problem. More and more people have recently focused on hashing algorithms [1-8], which can map floating point data into compact binary codes. Furthermore, the computer hardware instruction XOR can be utilized to compute the Hamming distance among binary codes, which can effectively reduce the ANN search task's time complexity.

The classical method, local sensitive hashing (LSH) [9], firstly proposes to map floating point data into compact binary code. LSH can map similar data points into the same compact binary code with a higher probability. Thus, LSH can directly utilize Hamming distance to achieve ANN search task. To guarantee the ANN search performance obtained in the Hamming space, LSH demands that the generated binary codes satisfy the local sensitive restriction. In other words, the Hamming distances among nearest neighbors should be smaller than those among dis-similar data points.

According to whether the training samples are utilized to learn hashing functions, existing hashing algorithms are roughly divided into data-independent algorithms [9] and data-dependent methods [10-12]. As no training process is involved in LSH [9], LSH [9] belongs to the data-independent method, randomly generating linear hashing functions. When the number of binary bits increases, the ANN search performance of LSH [9] method cannot obviously improve. To fix the problem caused by the random hashing functions in LSH [9] method, the data dependent hashing algorithms utilize machine learning mechanisms to learn the compact binary codes which can well preserve the floating data points' original Euclidean distance relationship. Spectral hashing [10] utilizes a similarity graph to represent the Euclidean distance relationship among data points and generate binary codes by graph partition mechanism. For a large scale dataset, the time complexity of establishing a spectral graph [10] is unacceptable. To avoid this problem, anchor graph hashing (AGH) [13] firstly adopts the K-means clustering method to learn clustering centers. Then only the centers are involved in establishing the spectral graph. Thus, AGH [13] method can effectively reduce the training time complexity. During the training process, both spectral hashing [10] and anchor graph hashing [13] demand the training

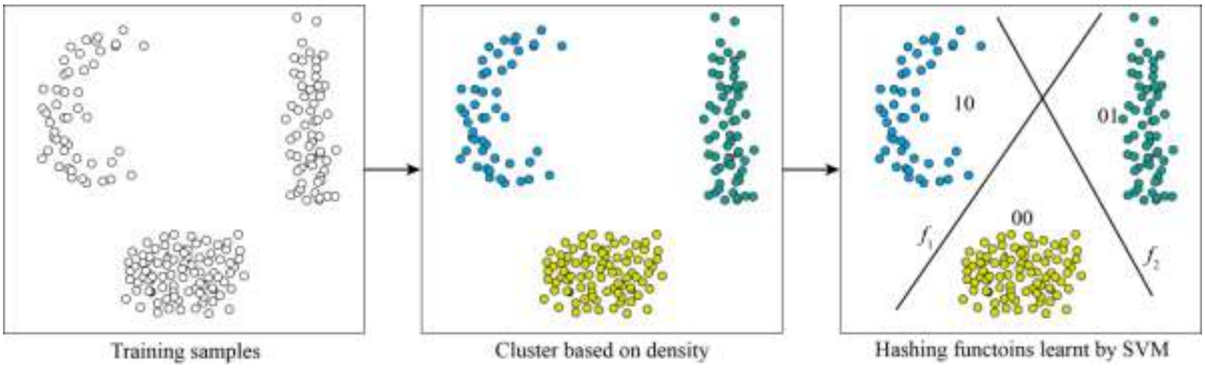

Fig 1: The flowchart of the proposed density peak hashing (DPH)

Samples should obey the uniform distribution. However, in practice, the real dataset does not satisfy the above assumption. So, an excellent hashing method should get rid of the distribution requirement. Principal component analysis hashing (PCAH) [14] considers the linear functions with large eigenvalues as hashing functions. PCAH [14] encodes the data according to their projection signs obtained based on the hashing functions. Unfortunately, many data points with small distances may be separated into different sides of the hashing functions, and they are encoded as different binary codes. To avoid the above problem, the RR method proposes to randomly rotate the PCA-projected data. To further improve the performance of obtained hashing functions, iterative quantization hashing iteratively learns the rotation matrix to minimize the similarity loss. In ITQ [11], the data points are mapped into the vertices of a fixed cubic. In contrast, K-means hashing [12] learns the encoding centers with minimal quantization error, which makes the encoding results adapt to data distribution.

According to the way of generating binary codes, existing hashing algorithms can be divided into two kinds, the lookup hashing [12] and the projection hashing [9, 11]. The lookup hashing (such as K-means hashing) encodes the data as the same binary code as its nearest center, and it needs to compute and compare the Euclidean distance between the data and all encoding centers. For generating M-bit binary code, the time complexity of lookup hashing method is $O(2^M)$. Correspondingly, the projection hashing just needs to compute M projection results, and the time complexity is only $O(M)$. Thus, in this paper, the proposed density peak hashing (DPH) adopts a two-step mechanism to learn the hashing functions and the flowchart is shown in Fig. 1. Firstly, the training samples are clustered into different groups according to the density distribution, and the density peaks are considered the cluster centers. The data points located in the same group are encoded as the same binary code. Secondly, the linear functions which can separate different clustering groups into different sides are learned to encode the training samples.

The main contributions of the proposed density peak hashing (DPH) are concluded as follows.

(1) The traditional K-means clustering method only adapts to the data with a spherical distribution. To avoid the above problem in this paper, the initial clustering groups are obtained based on the density value, which is more adaptive to the data distribution.

(2) DPH method demands a data pair's Hamming distance, and Euclidean distance have an identical value after transforming them into the same scale space. Thus, the Hamming lengths can approximate the corresponding Euclidean distances, which can improve the ANN search performance.

(3) During the training process, only the density peaks are involved in learning hashing functions, which can effectively reduce the training complexity.

II. DENSITY PEAK HASHING

The basic idea for mapping floating point data into binary codes includes two steps. (1) The similar data points should be clustered into the same group to encode the same binary code. (2) The binary codes of different cluster groups should satisfy the similarity preserving restriction. Correspondingly, two novel mechanisms are proposed in this paper, and the details are described below.

A. *Learning Cluster Centers Based on Density Value*

Traditionally, many hashing algorithms (such as K-means hashing [12] and anchor graph hashing [13]) employ the K-means clustering method to solve the problem (1). K-means clustering method utilizes Euclidean distance to measure the similarity degree among data points, and the data points are assigned to its nearest cluster center. Thus, the K-means clustering method is only applicable for the data points with a spherical distribution. In this paper, to get excellent clustering results for the data with any distribution, the clustering method based on density peak [15] is employed to learn the initial clustering centers. The learning process is introduced below.

The clustering method based on density peaks considers the clustering centers having the following features. Firstly, the clustering centers have a higher density value. Secondly, the distances among the data points with higher density values should be more considerable. Thus, the cluster centers can be found by finding the data points which own the above two characteristics.

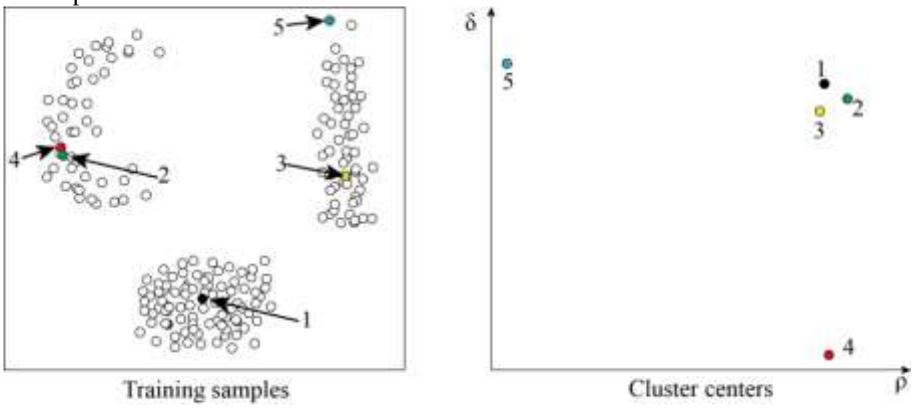

Fig 2: The cluster centers obtained based on density values

$X = \{x_1,\cdots, x_N\}$ represents the training dataset which contains N samples, and the distance between the i-th and j-th sample can be computed as d_{ij} = dist (x_i, x_j). Here, dist(·,·) means the Euclidean distance.

Usually, the cluster centers are located at the center of the group, and they are surrounded by the samples that belong to the same clustering group. The value of density can describe this feature. The local density value of the data point x_i can be computed by Eq. (1).

$$\rho(x_i) = \sum_{x_j \in X} \chi(d_{ij} - d_t) \tag{1}$$

In Eq. (1)$_{lt}$ is the threshold value. $\chi(\cdot\cdot)$ is the judicial function, and the definition is shown in Eq. (2).

$$\chi(x) = \begin{cases} 1, & x < 0 \\ 0, & x \geq 0 \end{cases} \tag{2}$$

According to the definitions of Eq. (1) and (2), the density value means the number of the data points which have smaller distance value than the threshold, and this definition is consistent with the characteristic of cluster centers.

Usually, more than one data point with higher density value is obtained. To further distinguish which data point is the right cluster center, the distances among the data points with high-density value should be computed, and its definition is shown in Eq. (3).

$$\delta_i = \min_{j \in C}(d_{ij}) \tag{3}$$

C is the dataset, which includes the data points with higher density value. δ_i returns the minimal distance value between the i-th sample and the other samples with higher density value.

Only the data points have a large density value during the training process, and δ_i values are considered cluster centers. To explain this phenomenon, one example is given in Fig. 2.

In Fig. 2，it is clear that all data points can be classified into three groups. By computing the density value ρ and the distance value δ, the number 1, 2, 3 data points with higher density value and maximum minimal distance value are considered cluster centers. The number 5 data has a higher density value, but its δ value is small. In contrast, the number 4 data has a large distance value and a minimum density value. Therefore, both numbers 4 and 5 data points can not be considered as a cluster center.

When the cluster centers are found, the remaining data points are assigned to its closest point with a higher density value.

B. Normalized Distance Similarity Preserving Hashing Functions

The above section has successfully clustered the data points $X = \{x_1\cdots, x_N\}$ into different groups. The next mission is to find their binary codes and the linear functions that can separate the data from the clustering results. The linear functions that map the data with different binary codes into different sides can be considered the hashing functions. To illustrate this situation, an example is given in Fig. 3.

In Fig. 3, the linear function f_1 can separate the data points with different binary codes on the first bit. Therefore, f_1 can be considered as the first hashing function. Correspondingly, f_2, which can separate the data with different binary codes on the second bit, is taken as the second hashing functions.

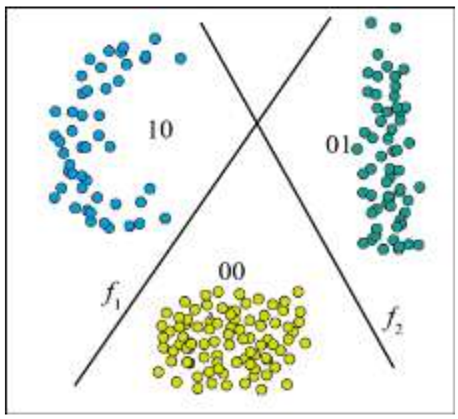

Fig 3: The linear hashing functions separate the data with different binary code into the other side.

As described above, the binary encoding process can be defined as in Eq. (4). The floating-point data are mapped into binary codes according to the sign of projection results.

$$\begin{aligned} B(x_i) &= \{b_1(x_i), \cdots, b_m(x_i)\} \\ &= \{sign(f_1(x_i)), \cdots, sign(f_m(x_i))\} \end{aligned} \tag{4}$$

The aim of mapping floating point data into binary code is to fast respond to the ANN search task. Thus, the generated binary codes should approximate the corresponding Euclidean distance, and this requirement is termed as similarity preserving restriction, which is defined in Eq. (5).

$$L(X) = \sum_{i=1}^{N} \sum_{j=i}^{N} (N(d(x_i, x_j)) - d_h(x_i, x_j))^2 \tag{5}$$

$d(\cdot, \cdot)$ represents the Euclidean distance and $d_h(\cdot, \cdot)$ computes the Hamming distance. As the value range of the Euclidean distances is different from that of Hamming distance, the normalization process is employed to make the different kinds of distance values have the same scale range. $N(\cdot)$ represents the normalization procedure.

In Eq. (5), the binary codes are involved in the process of computing Hamming distance. Unfortunately, the $sign(\cdot)$ function makes the binary codes have discrete integer values. As a result, it would be a NP-hard problem to directly optimize the objective function in Eq. (5).

In this paper, the tanh(·) function is utilized to relax the binary encoding function to a continuous form, as defined in Eq. (6).

$$b_m(x_i) = \tanh(w_m^T x_i) \tag{6}$$

Therefore, the Hamming distance of a data pair is re-defined as in Eq. (7).

$$d_h(x_i, x_j) = \frac{1}{2}(M - \tan(W^T x_i)\tan(W^T x_j)) \tag{7}$$

For the m-th hashing function, the partial gradient descent of the objective function is defined in Eq. (8).

$$\begin{aligned} \frac{\partial L(X)}{\partial w_m} &= \frac{\partial \sum_{i=1}^{N} \sum_{j=i}^{N} (d(x_i, x_j) - M \cdot d_h(x_i, x_j))^2}{\partial w_m} \\ &= \sum_{i=1}^{N} \sum_{j=i}^{N} (d(x_i, x_j) - M \cdot d_h(x_i, x_j)) \frac{\partial d_h(x_i, x_j)}{\partial w_m} \end{aligned} \tag{8}$$

For the Hamming distance of a data pair, its partial gradient descent is shown as in Eq. (9).

$$\begin{aligned} \frac{\partial d_h(x_i, x_j)}{\partial w_m} &= x_i(1 - \tanh(w_m^T x_i)\tanh(w_m^T x_j)) \\ &\quad + x_j(1 - \tanh(w_m^T x_j)\tanh(w_m^T x_i)) \end{aligned} \tag{9}$$

Finally, the gradient descent algorithm can be utilized to learn the m-th hashing functions and the parameter w_m is updated as in Eq. (10).

$$w_m = w_m - \frac{\partial L(X)}{\partial w_m} \tag{10}$$

During the training process, only the cluster centers obtained according to the density value are utilized to learn the hashing functions, and this measure has two merits. (1) It guarantees the encoding results are consistent with the clustering results obtained based on density peaks. (2) It reduces the number of data points involved in the training process which can effectively reduce the training time complexity.

III. EXPERIMENTS AND RESULTS

A. *Experimental Setting and Evaluation Measure*

The global feature descriptor GIST [16] is usually utilized to fulfill the image search task. Thus, the ANN search performance of the proposed DPH method and the other comparative methods can be evaluated by searching the nearest neighbor of GIST descriptor in the Hamming space. In this paper, three large scale image datasets including NUS-WIDE [17], 22K LabelMe [18] and ImageNet 100 are adopted to achieve the comparative experiments. The GIST descriptors [16] of the images in these three datasets are extracted and their nearest neighbors are defined based on Euclidean distance. To learn hashing functions and test their ANN search performance, each dataset is further divided into three parts including the training dataset, query dataset and test dataset. The samples in training dataset are utilized to learn hashing functions, and the nearest neighbors of the samples in query dataset are retrieved from the test dataset.

The images in NUS-WIDE [17] dataset are picked from Flickr dataset, and the number is 270 thousand. For NUS-WIDE dataset, its training dataset, query dataset and test dataset separately has 50 thousand, 50 thousand and 190 thousand images. The total number of the images in 22K LabelMe [18] is 22 thousand, and 20 thousand images are randomly picked as test dataset. Correspondingly, the other 2 thousand images in 22K LabelMe is considered as query samples and the training dataset has 5 thousand images. ImageNet 100 dataset is the sub-set of ImageNet dataset and it contains 100 categories of images. For ImageNet 100 dataset, 130 thousand images are randomly selected as the test data, and the training dataset has 30 thousand images. Correspondingly, the total number of the images in query dataset is 10 thousand.

In this paper, the evaluation standard precision is adopted to measure the ANN search performance of the proposed DPH method and the other comparative hashing methods. The definition of precision (pr) is shown in Eq. (11).

$$pr = \frac{\#(retrieved\ positive\ samples)}{\#(retrieved\ samples)} \tag{11}$$

pr represents the ratio of the positive samples to all retrieved samples. #(*retrieved positive samples*) counts the number of the nearest neighbors to the query samples in retrieval result. #(*retrieved samples*) is the number of retrieved samples. A higher precision value means the ANN search performance is better.

B. Comparative Experiments and Results

To prove the proposed density peak hashing (DPH) can well improve the ANN search performance, 7 classical hashing algorithms including local sensitive hashing (LSH) [9], spectral hashing (SH) [10], principal component analysis hashing (PCAH) [14], anchor graph hashing (AGH) [13], random rotation (RR) hashing, iterative quantization (ITQ) [11] hashing and K-means hashing [12] are utilized as the baseline methods.

The ANN search comparative experiments are separately conducted in three widely used large scale image datasets including NUS-WIDE [17], 22K LabelMe [18] and ImageNet 100. To achieve the image search task in the Hamming space, the GIST descriptors [16] of the images in these three datasets are separately mapped into 32-bit, 64-bit and 128-bit binary code. Then, the images with small Hamming distance to the query image are returned as the nearest neighbors. The precision curves of all methods in three datasets are shown in Figs. 4, 5, 6.

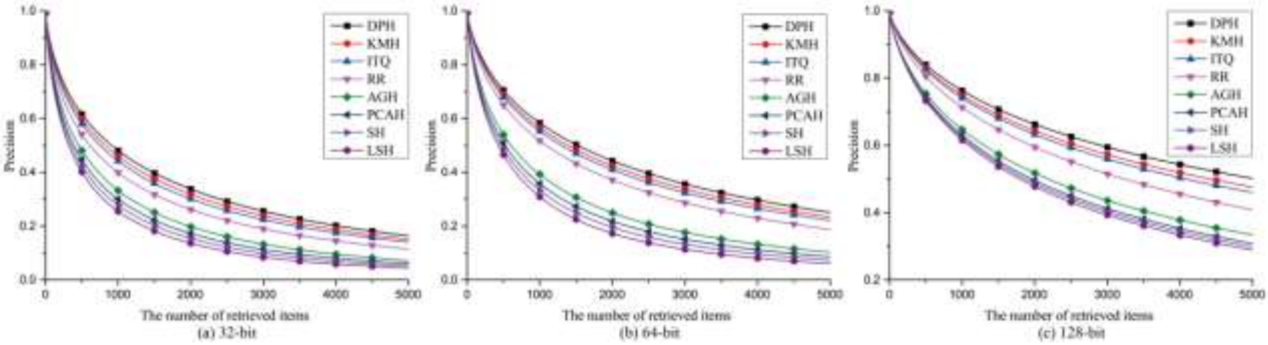

Fig 4: The precision curves of ANN search results in 22K LabelMe dataset.

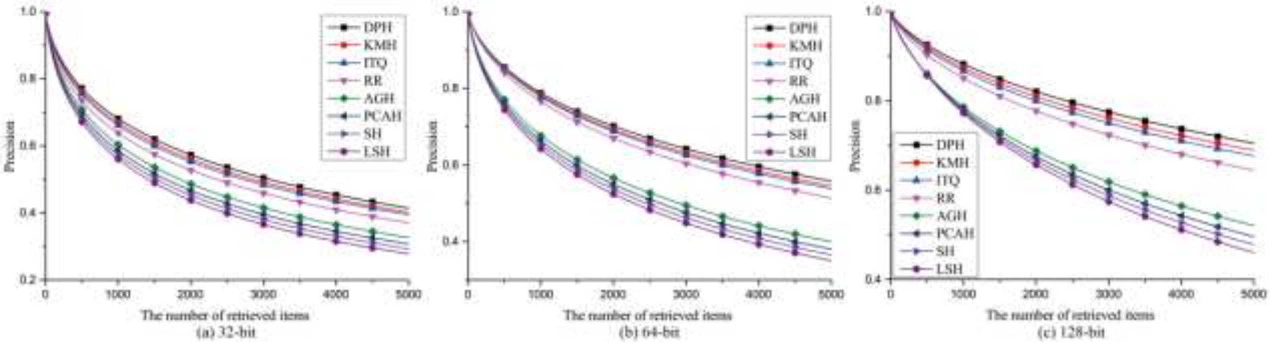

Fig 5: The precision curves of ANN search results in NUS-WIDE dataset.

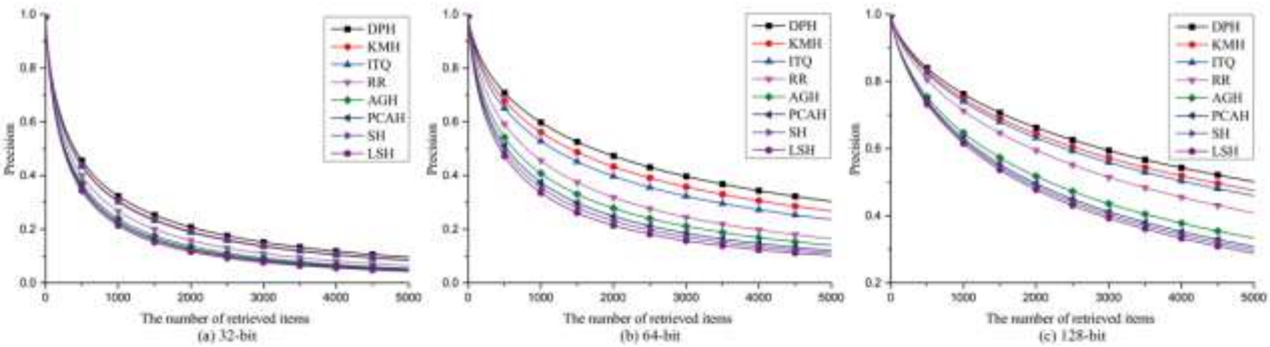

Fig 6: The precision curves of ANN search results in ImageNet 100 dataset.

LSH [9] method firstly proposes to achieve the ANN search task in the Hamming space. However, the hashing functions in LSH are independent from the training samples and they are randomly generated without a training procedure. As a result, the ANN search performance of LSH method is not stable and it can not obviously improve as the number of binary bit increasing. In contrast, the other 6 hashing algorithms and DPH utilize machine learning mechanisms to learn hashing functions which are well adaptive to the data distribution. Thus, the machine learning based methods can achieve a better ANN search performance with compact binary code. To further guarantee the ANN search performance obtained in the Hamming space, the similar data points should map into the same binary code. Therefore, SH [10] method establishes a similarity spectral graph and maps the nearest neighbors into the same binary code by graph partition mechanism. Correspondingly, the graph partition mechanism is also adopted by AGH method. But, not all samples are involved in the training process of AGH method. AGH [13] method adopts the K-means clustering method to learn the cluster centers. During the training procedure, the obtained cluster centers are considered as the vertices of anchor graph which can effectively reduce the training complexity. Both SH [10] and AGH [13] can achieve an excellent ANN search performance for the data points under the uniform distribution. Unfortunately, the real dataset does not obey the above assumption, and their ANN search performances are relative inferior in these three large scale dataset in this paper. In contrast, PCAH [14], RR, ITQ [11], KMH [12] and DPH have got rid of the restriction of data distribution. PCAH [14] considers the linear projection functions with high eigenvalues as hashing functions. Then, the high dimensional floating point data can be encoded as compact binary code according to the sign of the projection results. As many similar data points distribute around the hashing functions, they would be assigned as different binary code. This would lead an inferior retrieval result. To avoid the above problem, RR method rotates the PCA-projected data before generating binary code. However, RR method employs a random rotation matrix. To further improve the ANN search performance, ITQ [11] method adopts a machine learning mechanism to optimize the rotation matrix and it maps the data to the vertices of a hyper cubic. During the training procedure, ITQ [11] method aims to minimize both the quantization loss and the similarity loss, which can make the similar data points map to the same vertex. In ITQ [11] method, the hyper cubic is fixed and it may lead the encoding results not adapt to the data distribution. In contrast, KMH [12] method learns encoding centers which can minimize the sum of the distance values between the center and its nearest neighbors. Furthermore, KMH [12] assigns binary codes to the encoding centers by minimizing the similarity loss. KMH [12] adopts an optimization mechanism which likes that in the k-means clustering method to iteratively learn the encoding centers and binary codes. However, the centers obtained by k-means clustering method only adapt to the data with spherical distribution. In practice, the distribution of the dataset is non-spherical. In this paper, the proposed DPH method learns the encoding centers based on the training samples' density value. The final obtained clustering results are more adaptive to the data distribution. DPH requires the values of data pairs' Hamming distances and Euclidean distances are identical in the same scale space. Therefore, the Hamming distance can well approximate the Euclidean distance. Initially, the encoding mechanism of DPH method is as the same as that of K-means hashing. KMH encodes the data as the same binary code as its nearest center and it needs to compute and compare 2^M distance values. With the assistance of two-step mechanism, DPH further learns linear hashing functions based on the obtained clustering centers, and the encoding mechanism based on linear projection function is adopted. Then, DPH just needs to compute the projection results with M linear hashing functions to achieve the encoding procedure. Correspondingly, the encoding time complexity is only $O(M)$. Thanks to the above innovative measures, DPH can fast respond the ANN search task in large-scale dataset and have an excellent ANN search performance as shown in the experimental results.

IV. CONCLUSIONS

In this paper, a novel binary encoding mechanism termed as density peak hashing (DPH) is proposed to map the floating point data with high dimension into compact binary code. The nature task of binary encoding mechanism is clustering the data into different groups and assigning them binary codes. For the first stage, many hashing algorithms adopt the classical K-means clustering method to learn the classify results. As K-means clustering method divides the data points which have small Euclidean distance to the identical cluster center into the same group, the hashing algorithm based on K-means clustering mechanism only adapt to the data with spherical distribution. In practice, not all real datasets obey the above assumption. To solve this problem, DPH learns the clustering results according to the density value. During the training process, the data points which have higher density value and larger distance to its nearest density peak are considered as the cluster centers. To guarantee the data pairs' Hamming distances can approximate their Euclidean distances, the binary codes should satisfy the similarity preserving restriction. DPH transforms the Hamming distances and Euclidean distances into the same scale space and demands the two kinds of distances of each data pair are identical. There are two optional mechanisms, such as the lookup mechanism and projection based mechanism, to encode the raw data. When generate M-bit data, the lookup mechanism should compute and compare the distances between the raw data and 2^M centers, and its complexity is $O(2^M)$. In contrast, the projection based mechanism just needs to compute the projection results with *M* linear hashing functions and its complexity is $O(M)$. Therefore, DPH adopts the two-step mechanism to learn the linear hashing functions according to the obtained density peaks. As described above, DPH can encode large scale raw data points with a low time complexity and preserve the data pairs' Euclidean distance relationship in the Hamming space. Therefore, DPH can fast respond the ANN search task in the Hamming space. The final experimental results in three large scale image datasets have also shown that DPH can achieve an excellent ANN search performance.

ACKNOWLEDGMENT

This work is supported by the Nature Science Foundation of Shandong Province (No. ZR2018PF005) and the National Nature Science Foundation of China (No. 61841602).

REFERENCES

[1] Z. Wang, F. Sun, L. Zhang and P. P. Liu, "*Minimal residual ordinal loss hashing with an adaptive optimization mechanism*," EURASIP Journal on Image and Video Processing, vol. 2020, pp.1-11, Mar. 2020.

[2] Z. Wang, F. Sun, L. Zhang and L. Wang, "*Top position sensitive ordinal relation preserving bitwise weight for image retrieval*," Algorithms, vol. 13, pp. 18, Jan. 2020.

[3] Z. Wang, L. Zhang, F. Sun, L. Wang and S. Liu, "*Relative similarity preserving bitwise weights generated by an adaptive mechanism*," Multimedia Tools and Applications, vol. 78, pp. 24453-24472, Jan. 2019.

[4] R. Kang, Y. Cao, M. Long, J. Wang and P. S. Yu, "*Maximum-Margin Hamming Hashing*," in Proc. ICCV'19, 2019, pp. 8251.

[5] X. Luo, P. F. Zhang, Z. Huang, L. Nie, and X. S. Xu, "*Discrete Hashing with Multiple Supervision*," *IEEE Transactions on Image Processing*, vol. 28, pp. 2962-2975, Jan. 2019.

[6] G. Wu, J. Han, Y. Guo, L. Liu, and G. Ding, "*Unsupervised Deep Video Hashing via Balanced Code for Large-Scale Video Retrieval*," *IEEE Transactions on Image Processing*, vol. 28, pp. 1993-2007, 2019.

[7] C. Yue, B. Liu, M. Long, and J. Wang, "*HashGAN: Deep Learning to Hash with Pair Conditional Wasserstein GAN*," in Proc. CVPR, 2018, pp. 1287-1296.

[8] H. Liu, R. Ji, J. Wang, and C. Shen, "Ordinal Constraint Binary Coding for Approximate Nearest Neighbor Search," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, vol. PP, pp. 1-1, 2018.

[9] M. Datar, N. Immorlica, P. Indyk and V. S. Mirrokni, "*Locality-sensitive hashing scheme based on p-stable distributions*" in Proc. [C]. SCG'20, 2004, pp. 253-262.

[10] Y. Weiss, A. Torralba and R. Fergus, "*Spectral hashing*," in Proc. NIPS, 2008, pp. 1753-1760.

[11] Y. Gong and S. Lazebnik, "*Iterative Quantization: A procrustean approach to learning binary codes*," in Proc. CVPR, 2011, pp. 817-824.

[12] K. He, F. Wen and J. Sun, "*K-means hashing: an affinity-preserving quantization method for learning binary compact codes*," in Proc. CVPR, 2013, pp. 2938-2945.

[13] W. Liu, J. Wang, S. Kumar and S. F. Chang, "*Hashing with graphs*," in Proc. ICML' 28, 2011, pp. 1-8.

[14] Y. Matsushita and T. Wada, "*Principal component hashing: An accelerated approximate nearest neighbor search*," in Proc. PSIVT, 2019, pp. 374-385.

[15] A. Rodriguez and A. Laio, "*Clustering by fast search and find of density peaks*," Science, vol. 344, pp. 1492-1496, Jun 2014.

[16] A. Oliva and A. Torralba, "*Modeling the shape of the scene: A holistic representation of the spatial envelope*," International journal of computer vision, vol. 42, pp. 145-175, May, 2001.

[17] X. Wang, Y. Shi, and K. M. Kitani, "*Deep Supervised Hashing with Triplet Labels*," in Proc. ACCV, 2016, pp. 70-84.

[18] F. Cakir and S. Sclaroff, "*Adaptive Hashing for Fast Similarity Search*," in Proc. CVPR, 2015, pp. 1044-1052.

[19] Pooja Goyal, Sushil Kumar, Komal Kumar Bhatia, "Hashing And Clustering Based Novelty Detection" SSRG International Journal of Computer Science and Engineering 6.6 (2019): 1-9.