IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 31, 2017, accepted November 27, 2017, date of publication December 8, 2017,
date of current version February 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2017.2781360

Binary Hashing for Approximate Nearest Neighbor Search on Big Data: A Survey

YUAN CAO[1], HENG QI[1,2], WENRUI ZHOU[1], JIEN KATO[2], KEQIU LI[1], (Senior Member, IEEE), XIULONG LIU[1], AND JIE GUI[3], (Senior Member, IEEE)

[1] School of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China
[2]Graduate School of Information Science, Nagoya University, Nagoya 464-0814, Japan
[3]Institute of Intelligent Machines, Chinese Academy of Sciences, Hefei 230000, China

Corresponding author: Heng Qi (hengqi@dlut.edu.cn)

This work was supported in part by the NSFC under Grant 61772112, Grant 61672379, Grant 61370199, and Grant 61572463, in part by the State Key Program of National Natural Science of China under Grant 61432002, in part by the Dalian High-level Talent Innovation Program under Grant 2015R049, and in part by the JSPS KAKENHI under Grant 16F16349. The work of the Y. Cao was supported by the China Scholarship Council during a visit to Rutgers University.

ABSTRACT Nearest neighbor search is a fundamental problem in various domains, such as computer vision, data mining, and machine learning. With the explosive growth of data on the Internet, many new data structures using spatial partitions and recursive hyperplane decomposition (e.g., k-d trees) are proposed to speed up the nearest neighbor search. However, these data structures are facing big data challenges. To meet these challenges, binary hashing-based approximate nearest neighbor search methods attract substantial attention due to their fast query speed and drastically reduced storage. Since the most notably locality sensitive hashing was proposed, a large number of binary hashing methods have emerged. In this paper, we first illustrate the development of binary hashing research by proposing an overall and clear classification of them. Then we conduct extensive experiments to compare the performance of these methods on five famous and public data sets. Finally, we present our view on this topic.

INDEX TERMS Approximate nearest neighbor search, large-scale database, hashing based methods, overview.

I. INTRODUCTION

Nearest Neighbor (NN) Search is to take a query point and accurately find the examples which are most similar to it within a large database. It is a fundamental problem in many applications, such as computer vision [1], [2], information retrieval, data mining and machine learning. However, with the increasing availability of all kinds of data, especially visual data in a variety of domains (e.g. scientific image data, community photo collections on the Web, news photo collections or surveillance archives), fast indexing and search for large database is critical. Hence, Approximate Nearest Neighbor (ANN) Search methods occur to expedite the retrieval process by sacrificing a predictable loss in accuracy.

ANN Search methods are generally broken into two families. One group relies on data structures using spatial partitions and recursively hyperplane decomposition, which includes k-d trees [3], metric trees [4], cover trees [5], and other related techniques. These methods perform well when processing low-dimensional data, however, they degenerate to a linear scan in the worst case, when data are distributed in high dimensions. The other group concentrates on hashing based methods, which map data points to low-dimensional binary codes in Hamming space. This paper focuses on the later one.

Efficient hashing based methods should have several properties. First, the algorithms should map similar points in original space into similar binary codes in Hamming space. This property is to ensure the algorithms to preserve the similarity between the two spaces. Second, the dataset should be encoded into a small number of bits to increase retrieval speed and reduce memory storage. For example, for an ordinary workstation with 16 GB memory, to store 500 million images in memory, we could only use about 32 bits for each image. Finally, the algorithms for both learning parameters of hash functions and encoding a new test point should be very efficient. To simultaneously satisfying the above three requirements makes the hash codes learning problem quite challenging.

After learning the parameters of hash functions on the training dataset offline, the online search process of hashing based methods contains three main steps. First, to compute the binary code of the query point. Second, to find the buckets

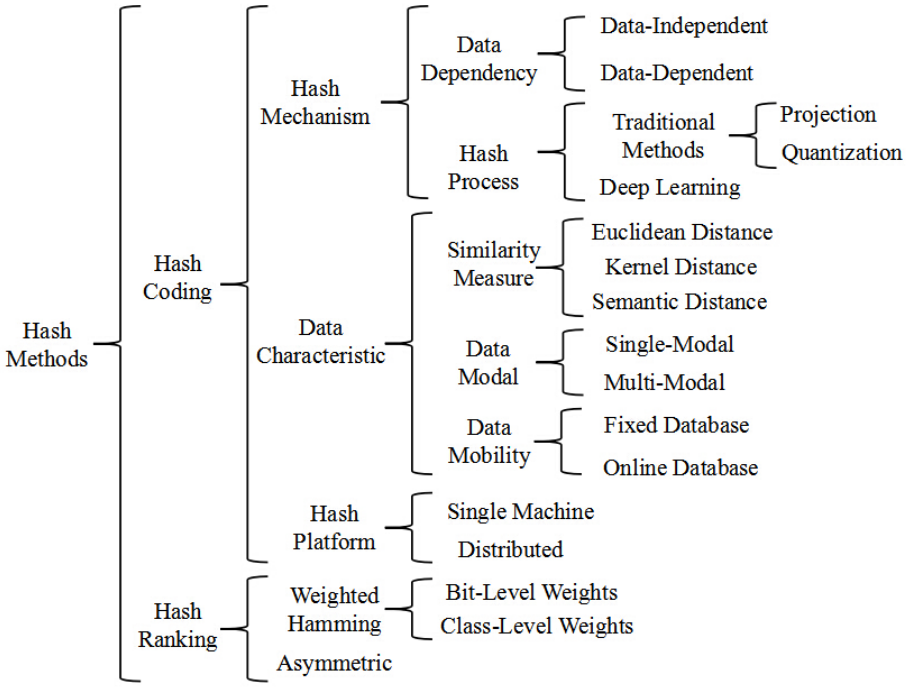

FIGURE 1. Hash Categories.

with small Hamming distances to the query point. Finally, to load the points in the selected buckets into memory and rerank them if necessary.

In view of the above process of hash based ANN search, hash methods can be grossly divided into two categories: Hash Coding and Hash Ranking. Here, we introduce the two categories and more detailed classification in each category as follows (Fig.1). Table 1 shows the characteristics of hash coding methods from three viewpoints, namely hash mechanism, data characteristic and hash platform. Table 2 shows the characteristics of hash ranking methods. Recently, deep learning based hash methods are popular. Liu *et al.* gives Table 3 to summarize existing deep learning based methods [6]. This is not the focus of this paper. We will describe and discuss the hash coding methods and the hash ranking methods in following sections.

A. HASH CODING

Hash Coding based methods aim at how to learn better hash functions to map the data points in original space to Hamming space. We further divide Hash Coding methods into two categories from different angles.

a. Hash Mechanism. These methods take into account how to learn hash functions and whether they are data-dependent or not.
b. Data Characteristic. These hash methods vary according to different similarity measures among data points, data mobility (fixed or floating), data modal (single or multiple) and data distribution (Concentrated or Distributed).

B. HASH RANKING

Hash Ranking based methods pay attention to hash codes ranking, especially for hash codes that share the same hamming distance to the query point.

a. Weighted Hamming Distance. These methods try to learn different weights for each hash code. Some focus on learning bit-wise weight for each hashing bit, some devote to learning overall weight for each class.
b. Asymmetric Distance. This kind of methods only map database points to binary codes but do not map the query point. The distance between this two unequal spaces is defined as asymmetric distance.

The rest of this paper is organized as follows. First, we give an overview and a detailed category on Hash Coding methods in Section 2. Then, we present an introduction of Hash Ranking methods in Section 3. Next, we make extensive experiments on five common datasets to compare the accuracy among several notable hashing based methods in Section 4. Finally, we conclude this paper and propose conclusion and future work on hashing based methods in Section 5.

II. HASH CODING METHODS

A. HASH MECHANISM

1) DATA DEPENDENCY

Hash Based Methods aim at learning hash functions to map high dimensional points to binary codes. Early hash functions are independent of data distribution, which means that the learning process does not rely on any information

TABLE 1. Category of hash coding methods.

Hash Methods	Hash Coding												
	Hash Mechanism				Data Characteristic							Hash Platform	
	Data Dependency		Hash Process		Similarity Measure			Data Modal		Data Mobility		Single	Distributed
	Dependent	Independent	Projection	Quantization	Euclidean	Kernel	Semantic	Single	Multiple	Fixed	Online		
LSH [7]		√	√		√			√		√		√	
SH [8]	√		√		√			√		√		√	
BRE [9]	√		√			√	√	√		√		√	
ITQ [10]	√			√	√			√		√		√	
MLH [11]	√		√		√		√	√		√		√	
SSH [12]	√		√				√	√		√		√	
SPLH [13]	√		√				√	√		√		√	
FSDH [14]	√		√				√	√		√		√	
SDHR [15]	√		√				√	√		√		√	
IH [16]	√		√		√			√		√		√	
Ladahash [17]	√		√		√			√		√		√	
BP [18]	√		√		√			√		√		√	
HH [19]	√		√			√		√		√		√	
DBQ [20]				√									
VBQ [21]				√									
OKH [22]	√		√			√		√		√		√	
KSH [23]	√		√			√	√	√		√		√	
WSH [24]	√		√			√	√	√		√		√	
LSBC [25]		√	√			√		√		√		√	
JOSAT [26]	√		√			√	√	√		√		√	
RMMH [27]	√		√			√		√		√		√	
KLSH [28]		√	√			√		√		√		√	
AGH [29]	√		√		√			√		√		√	
CPH [30]	√		√		√			√		√		√	
CVH [31]	√		√						√	√		√	
IMH [32]	√		√						√	√		√	
CMFH [33]	√		√						√	√		√	
STMH [34]	√		√						√	√		√	
OH [35]	√		√		√		√	√			√	√	
OSH [36]	√		√				√	√			√	√	
OSKH [37]	√		√		√			√			√	√	
DH [38]	√		√		√			√		√			√

TABLE 2. Category of hash ranking methods.

Hash Methods	Hash Ranking		
	Weighted Hamming		Asymmetric
	Bit-Level	Class-Level	
QsRank[39]	√		
WhRank[40]	√		
QARank[41]		√	
AsyE[42]			√
AsyE3[43]			√

of the data. These hash functions are mainly made up of a few random projections that follow certain distribution (e.g. Gauss distribution). The most notably data-independent hashing based method is Locality-Sensitive Hashing (LSH) [7]. Here, We will introduce LSH method briefly as follows.

LSH method offers probabilistic guarantees of retrieving items within $(1+\varepsilon)$ times the true nearest neighbor distance, with query process in constant or sub-linear time respect to n. LSH does this by computing a number of randomized hash functions that guarantee a high probability of collision for similar examples. It has been verified that with high probability, points that are close in original space will have similar hashing codes, and this fact is utilized for efficiently finding approximate nearest neighbors.

Despite its success, it has been observed that the theoretical guarantees are asymptotic as the number of random projections grows. Thus, it needs long codes to achieve acceptable accuracy. However, since the collision probability decreases exponentially with the length of the code, more hash tables are needed to get a good recall. Hence, it would be hard to scale up to millions of points in consideration of time and memory. Therefore, many recent works [8]–[11] focus on generating data-dependent short compact hashing codes.

A well-known data-dependent hashing based method to learn compact codes is Spectral Hashing (SH) [8], which has been shown to attain much higher accuracy than LSH in practical applications. SH constructs an objective function on the training data points to preserve the similarity between original space and Hamming space, which means that the points which are similar in original space can be mapped to similar binary codes with low Hamming distance. With

TABLE 3. Category of deep learning based hash methods (cited from [6]).

Deep Learning based Hash Methods	Category	Network	Layer	Feature+Hash
Deep Learning of Binary Hash[44]	Supervised	Single	8	Independent
Deep Learning for Compact Binary Codes[45]	Unsupervised/Supervised	Single	3	Independent
Unsupervised Deep Neural Networks Hashing[46]	Unsupervised	Single	16	Independent
Supervised Deep Hashing[47]	Supervised	Single	5	Associative
Semantics-Preserving Deep Hashing[48]	Supervised	Single	8	Associative
Deep Semantic Ranking Hashing[49]	Supervised	Single	8	Associative
Binary Deep Neural Network Hashing[50]	Unsupervised/Supervised	Single	5	Associative
Bit-Scalable Deep Hashing[51]	Triplet-wise	Parallel	10	Associative
One-stage Deep Hashing[52]	Triplet-wise	Parallel	10	Associative
Deep Pairwise-Supervised Hashing[53]	Pairwise	Parallel	8	Associative
Deep Hashing Network[54]	Pairwise	Parallel	8	Associative

this objective and a few constraints, the learnt hashing codes of database points are the c (code length) eigenvectors of a Laplacian matrix with minimal eigenvalues (excluding 0).

Since SH method utilizes the data distribution to learn hashing codes, it always performs better than data-independent hashing based methods like LSH. However, there are also some shortcomings in SH method. First, the optimal hashing code formulation in SH suffers from the inability to handle new data points, because no explicit hash function is produced. Second, its behavior is not easy to characterize from the theory pespective. In fact, it is not clear whether Hamming distance between hash codes converges to similarity measure in original space as the number of hash bits increases. Third, in practice, the projected data points are often distributed on a few dimensions which results in SH working well for only relatively small codes. In the next section, we will describe the objective function and hashing codes learning process in detail.

2) HASH PROCESS

- Traditional Methods. Before 2014, most hashing based methods [8], [9], [12], [13], [16]–[19] follow the traditional hash framework composed of two processes: Projection and Quantization. In the Projection process, the data points in original high dimensional space are mapped to a low dimensional projection space by a few hash functions. Then, the data points in the projection space are quantized to binary codes with several thresholds during the Quantization process. In order to illustrate the two processes more clearly, we limn them as follows:

  Let us first introduce our notations. We assume that the data points in original space are denoted as $\{x_1, x_2, \ldots x_n\}, \forall x_i \in R^d$, which form the rows of the data matrix $X \in R^{n\times d}$. The object of hashing based methods is to find a projection matrix $W \in R^{d\times c}$, where c denotes the code length. For $j = 1 \ldots c$, the hash function is defined by

$$\begin{aligned} y_{ij} &= x_i w_j \\ B_{ij} &= Q(y_{ij}) \end{aligned} \tag{1}$$

  where $w_j \in R^d$ is one column of the matrix P, $B \in R^{d\times c}$ is the resulted binary code matrix and

$$Q(y_{ij}) = \begin{cases} 1, & if\ y_{ij} \geqslant t_j; \\ 0, & otherwise. \end{cases} \tag{2}$$

  in which t_j is the corresponding threshold determined by hash based methods. In this way, we can formulate the whole hash coding process as $B = Q(XW)$.
  From the formula (1), we can see that one hash function can map a data point to one bit, therefore, we need c hash functions to learn c bits. The difference among various hashing based methods is the computation of projection matrix W [10]. Now, we show how hashing based methods learn projection matrixes and how to quantize the data points in projection space.

  – Projection. We define the Projection process as $P = XW$. Now, the most common hashing based methods learn hash functions by constructing an objective function and then minimize or maximize it to preserve the similarity between original space and Hamming space. Meanwhile, in order to increase the performance of hashing based methods, the hash functions should also meet some constraints, such as: Balance and Independence.
    Take SH [8] as an example, we present the objective function and constraints in SH method as follows:

$$\begin{aligned} minimize: &\quad \sum_{kij} S_{i,j}(y(i,k) - y(j,k))^2 \\ suject\ to: &\quad Y(i,j) \in \{-1, 1\} \\ &\quad Y^T 1 = 0 \\ &\quad Y^T Y = I \end{aligned} \tag{3}$$

    in which, $y(i,k)$ and $y(j,k)$ denote the k-th bit of points i and j respectively in projection space. S_{ij} represents the similarity between points i and j in original space with values in $[0, 1]$. It is clear to observe that, if two points are similar in original space, which means S_{ij} is large, the value of $(y(i,k) - y(j,k))^2$ should be small. Hence,

SH learns hash functions by minimizing the above objective function and meanwhile meeting the two constraints: Balance and Independence.

From the information theoretic point of view, we would like to maximize the information provided by each bit. According to maximum entropy principle, a balance partitioning of data space provides maximum information. Thus, it is desired to have $Y^T 1 = 0$, where $Y \in B^{n*c}$ denotes the binary codes of n data points. On the other hand, both the worst case and the average case of time complexity can be minimized if the hash buckets are perfectly balanced. In the meanwhile, each hash function should also be mutual independent, $Y^T Y = I$, to reduce the information redundancy produced by similar data space partitioning.

- Quantization. In the Quantization process, the data points in the projection space are quantized into binary codes by thresholding. Currently, most of the hashing based methods quantize each projection dimension with one single threshold which is called Single-Bit Quantization (SBQ). The thresholds are often computed by meeting the balance constraint: $(Y^T 1 = 0)$.

  However, one problem with SBQ is that the thresholds always lie in the region of the highest point density, which leads to many neighboring points close to the thresholds quantized into totally different codes. Therefore, a new quantization method dubbed Double-Bit Quantization (DBQ) [20] is proposed to solve the problem of SBQ by quantizing each projection dimension into double bits with adaptively learned thresholds. It attempts to avoid setting thresholds among data points with similar projected values.

  In addition, since a subset of hash functions may be more informative than others, a dubbed Variable-Bit Quantization (VBQ) [21] method is introduced to locate a variable number of bits for each LSH hyperplane. This is done by constructing such an objective function that Hyperplanes which better preserve the neighbourhood structure of data points in original space are awarded more bits. In comparison with SBQ and DBQ [20], VBQ [21] can get a much higher accuracy. However, it can only be applied on LSH hash method, which makes VBQ [21] limited in real applications.

- Deep Learning Based Methods. Hashing is a widely-used approximate nearest neighbor search approach for large-scale image retrieval. In the existing hashing based methods for image retrieval, an input image is usually denoted by a vector of hand-crafted visual features that do not necessarily preserve the accurate semantic similarities of images pairs, which leads to poor performance of hash function learning. Therefore, deep learning based hashing methods [44]- [54] are proposed to simultaneously learn a good image representation and a set of hash functions automatically.

  To our knowledge, Semantic Hashing [55] is the first work on deep learning techniques for hashing. They learn hashing codes for images by stacked Restricted Boltzmann Machines (RBMs). Similarly, Torralba *et al.* [56] model a deep network by using multiple layers of RBMs. However, these models are complex and surpassed by many recent hashing based methods.

  The aforementioned two deep learning based hashing methods are both unsupervised which do not utilize the label information of training points. Until 2014, Xia et al. proposed a supervised hashing based method called Convolutional Neural Network Hashing (CNNH) [47]. They first learn approximate hashing codes from pairwise similarity matrix decomposition and then simultaneously learn image features and hash functions with the raw image pixels as input using CNNs to fit the learnt hash codes. However, there are some limitations in CNNH algorithm. Therefore, a few improved deep learning based hashing methods are proposed in 2015.

  First, the learnt image features can not give feedback for hash codes learning, although the learnt approximate hash codes can guide the learning of image features, which means that CNNH can not learn image features and hash codes simultaneously. Therefore, Deep Pairwise-Supervised Hashing (DPSH) [53] is proposed to perform simultaneous feature learning and hash code learning for applications with pairwise labels. They integrate all components into the same deep architecture to map the images from pixels to pairwise labels in an end-to-end way. Therefore, different components can give feedback to each other, which leads to better codes learning.

  Second, in the image features learning stage, a matrix-decomposition algorithm is used which requires the input of a pairwise similarity matrix. It consumes considerable storage and computational time when the data size is large. Therefore, Lin *et al.* [44] present an efficient deep learning approach to learn a set of effective hash-like functions.

  Third, CNNH does not explicitly impose the ranking constraint on the deep models, which can not figure out the multi-level similarity problem. Zhao *et al.* [49] learn a joint optimization of feature representation and mapping them to hash codes. Hash functions are learnt with semantic ranking supervision which is the order of a ranking list derived from shared class labels between query and database images.

B. DATA CHARACTERISTIC

1) SIMILARITY MEASURE

- Euclidean Distance. In general, the similarity between two points x and y in original space is measured by

Euclidean distance which is defined as follows:

$$E_d(x, y) = ||x - y||_2 = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \quad (4)$$

Many hash functions are leant based on this similarity measure, such as LSH [7] and SH [8] mentioned before.

- Kernel Distance. The aforementioned hashing based methods have one important limitation: they assume the points in original space are in a vector format. However, in many real applications such as multimedia, biology or Web, data types are always in the forms of graphs, trees, sequences, sets, or other formats. For such general data types, a number of complex kernels are defined to compute data similarities.

  Moreover, even if the data are stored in the vector format, for computer vision applications such as image retrieval, kernel distance can show better semantic similarities between images than Euclidean distance. In addition, many machine learning applications benefit from the use of domain specific kernel functions, for which the data in original space is not known explicitly, namely only the pairwise kernel function is computable. Therefore, many hash based methods [9], [22]–[27] are developed dealing with the kernel functions applicable to both vector and non-vector inputs.

  Kernel function includes linear kernel function, polynomial kernel function, Gauss kernel function and so on, in which Gauss kernel function is the most common one and is also called Radial Basis Function (RBF), a kind of scalar function along the radial symmetry. It is defined as follows:

$$k(||x - xc||) = e^{-\frac{||x-xc||^2}{2\sigma^2}} \quad (5)$$

  where xc is the center of kernel function, σ is the parameter of width. We can see that Gauss kernel function is an antidependence function of Euclidean distance.

  Now, let's introduce Optimized Kernel Hashing OKH [22] in detail. In order to attain efficient hash codes, they start with the same objective function with SH [8]. However, SH has two important limitations: First, since there are no explicit hash functions in SH method, it is hard to encode a novel input point. Second, SH can not provide effective solutions for kernel similarities between points in original space.

  OKH adopts the following formulation:

$$\begin{aligned} \min_{A,b} \quad & \frac{1}{2} \sum_{i,j=1}^{N} S_{ij} ||Y_i - Y_j||^2 + \lambda \sum_{m=1}^{M} ||V_m||^2 \\ s.t. \quad & \sum_{i=1}^{N} Y_i = 0 \\ & \frac{1}{N} \sum_{i=1}^{N} Y_i Y_i^T = I \\ & Y_i \in \{-1, 1\}^M \end{aligned}$$

$$\begin{aligned} Y_{mi} &= h_m(X_i) = sign(V_m^T \varphi(X_i) - b_m) \\ V_m &= \sum_{p=1}^{P} A_{pm} \varphi(Z_p), \quad i = 1, ..., N, \ m = 1, ..., M \end{aligned} \quad (6)$$

  By observing Eq. 3 and Eq. 6, we can see the main difference between SH and OKH algorithms is that, hash functions $Y_{mi} = h_m(X_i) = sign(V_m^T \varphi(X_i) - b_m)$ represented in the kernel form are included in the objective function. Here, Y_{mi} is the mth bit for Y_i. There are M hash functions $\{h_m, m = 1, ...M\}$ in total, each of which is for one hash bit. In Eq. 6, V_m is the hyperplane vector in the kernel space, φ is the function for embedding the points in original space to kernel space, and b_m is the threshold on m-th bit. Since it is infeasible to define the hyperplane vector V_m directly in the kernel space, they represent V_m as a linear combination of landmarks in the kernel space with combination weights denoted as A_{pm}, and $\{Z_p, p = 1, ...P\}$ are landmark points which can be "basis" vectors. In addition, there is an additional item in OKH's objective function $\lambda \sum_{m=1}^{M} ||V_m||^2$ which is utilized to a regularized term as control the smoothness of the kernel function.

- Semantic Distance. Since unsupervised methods [8], [9], [19], [28]–[30] do not require any labeled training points, their objective functions are always constructed with the pre-specified similarity matric. However, in image retrieval applications, sometimes similarity between points is not defined with a simple metric, because this kind of metric may not preserve semantic similarity. Therfore, semantic similarity is usually given in terms of labeled pairs of images. In other words, similar images in original space share at least one common label. For such pairwise labeled data, one would like supervised hashing methods [12], [13], [23], [24] to automatically generate codes that preserve this semantic similarity.

  There exist a few supervised hashing methods that can handle semantic similarity, however, they are prone to overfitting when labeled data is small or noisy. What's more, these methods are usually slow to train. Hence, a kind of Semi-Supervised Hashing (SSH) [12] is proposed to learn hash codes by minimizing empirical error on the labeled data while maximizing variance and independence of hash bits over the labeled and unlabeled data. Here is the objective function with constraints in SSH method:

$$\begin{aligned} H^* = \ & arg \max_H J(H) \\ & subject\ to \quad \sum_{i=1}^{n} h_k(x_i) = 0, k = 1, ..., K \\ & \frac{1}{n} H(X) H(X)^T = I \end{aligned} \quad (7)$$

where, $J(H)$ measures the empirical accuracy on the labeled data for a group of hash functions H:

$$J(H) = \sum_{k}\{\sum_{(x_i,x_j)\in M} h_k(x_i)h_k(x_j) - \sum_{(x_i,x_j)\in C} h_k(x_i)h_k(x_j)\} \quad (8)$$

in which, M contains pairs of points with the same label and C contains that with different labels. Since the objective function $J(H)$ itself is non-differentiable and the balancing constraint makes the problem NP hard, the authors relax the objective function as well as the constraints to obtain an approximate solution for hash functions. Furthermore, there are also other two interesting semantic hashing methods: FSDH [14] and SDHR [15]. Fast supervised discrete hashing (FSDH) [14] does not only outperform other methods, but also very fast, which is very useful in practical applications. SDHR [15] learns a real-value label matrix R to get higher accuracy than many other semantic hashing methods.

2) DATA MODAL

Nowadays, most of the existing hashing researches pursue the binary codes on homogeneous similarity assessment, which means that the data entities in the database are all of the same type. However, with the rapid development of the Internet and multimedia devices, tremendous amounts of data have been easily obtained, such as document, images and videos. As the data sizes increase, the density of similar objects also increases. In practice, some databases include data objects with multiple views, or some relevant multimedia data from different media types may have semantic correlations. For example, an image can be associated with related content-based texts, descriptors, tags, or links. Since the heterogeneous entities and relationships are also ubiquitous in real applications, there is an emerging need to search similar data entities from multiple heterogeneous domains.

Recent researches focus on multi-modal hashing methods to address cross-modality similarity search problem by mapping heterogeneous data points in original space into a common Hamming space. The core problem in multi-modal hashing is how to simultaneously construct the underlying correlations among multiple modalities and preserve similarity relationships in each individual modality. One of the attempts aims at translating each of the modalities of a multi-modal object to one of the modalities and employing single-modal similarity search techniques. However, this kind of methods suffers from two main weaknesses [57], [58]. First, there is approximate loss in the translation stage and it is application dependent [57]. Second, the translation stage is always very slow, which makes it not practical to many real applications [58].

Hence, a number of effective and efficient multi-modal hashing methods appear continuously since 2010. Cross-View Hashing (CVH) [31] learns hash codes meeting the conditions that the binary codes of different views of the same object should be similar if not identical and that similar data objects should also be encoded to similar binary codes. CVH does this by minimizing the weighted average Hamming distance of the binary codes for the training objects over all the views through solving a generalized eigenvalue problem. However, CVH ignores the differences between the modalities which results in poor performance [34].

Inter-Media Hashing(IMH) [32] explores the correlations among multiple media types from different data sources and transforms multimedia data from heterogeneous data sources into a common Hamming space. They use linear regression with regularization model to learn view-specific hash functions so that the hash codes for new data points can be efficiently generated. However, IMH can not be applied to large-scale data set, because it needs to construct the similarity matrix for all the data points, which leads to a large computational complexity [34].

The aforementioned two hashing methods need to store independent hash codes to implement cross-view search, which increases the cost of storage and search. Collective Matrix Factorization Hashing(CMFH) [33] learns unified hash codes by collective matrix factorization with latent factor model from different modalities of one instance. CMFH avoids the large scale graph construction and eigen-decomposition problems which are very time consuming. However, CMFH assumes restrictive constraint that each view of one instance generates identical hash codes. The consistence of pairwise data points is guaranteed, however, the cross correlationship between different pairwise data points is ignored.

Although these multimodal hashing methods perform well in multimodal applications, they all discard the discrete constrains to get hash codes, which will cause large quantization error and get poor search performance for long codes. Semantic Topic Multimodal Hashing(STMH) [34] focuses on the binary nature of hashing to facilitate the large scale cross-media retrieval for multimedia data sources with texts and images by learning latent topics from texts and the unified hash codes simultaneously.

3) DATA MOBILITY

Existing popular hashing based methods usually employ a batch-learning strategy. However, two critical problems that are rarely mentioned before have to be taken into consideration: First, in real-world applications, the data often comes sequentially. For instance, a search engine company like Baidu has numerous new web pages including texts and images continuously arriving at the data centers each day. The existing hashing based methods do not have the ability to adapt to the changes as a dataset grows and diversifies, because they have to retrain new hash functions, which is apparently a less efficient learning manner for streaming data. Second, when the dataset becomes huge, the computational cost and memory requirement may be intractable and

infeasible. In fact, data is usually stored in a distributed manner and it is too large to be read into memory of one workshop.

Therefore, a few online hashing based methods appear in succession. Online Hashing [35] is known as the first attempt to develop a kernel mapping based passive-aggressive learning strategy for accommodating a new pair of data aiming at mapping the data points in original space to finite number of hash codes, meanwhile appropriately preserving the similarity relationship between data points.

Online Supervised Hashing [36] is the first supervised hashing method that allows the label space to grow. Given an incoming stream of training data with corresponding labels, they learn and adapt their hashing functions in a discriminative manner. The method can also adapt to new classes presented in a incoming data stream.

Online Sketching Hashing [37] proposes a novel online hashing approach based on the idea of data sketching. A sketch of one dataset preserves the main property of interest but with a significantly smaller size. With a small size sketch, the method can learn hash functions in an online fashion with low computation and storage complexities. It performs well because the computations performed on the sketch rather than the whole dataset do not lose much information.

C. HASH PLATFORM

All the hash based methods we discussed above are designed for the centralized setting, that is to say, are single machine approaches. However, in many real-word applications, the size of datasets is more and more large. Therefore, the data is often stored in distributed machines. One intuitive way is to gather all data together to a fusion center before training, and learn hash functions like in a single machine. However, it is easy to observe that it is prohibitively expensive in computation, both time and space. Hence, Hashing for Distributed Data [38] develops a hashing model to learn hash functions in a distributed manner. Since there is no exchange of training data across the nodes in the learning process, the communication cost is very small, which makes it easy to adapt to arbitrary large-scale distributed applications.

III. HASH RANKING METHODS

Most of the recent researches devote to increasing the accuracy of hashing based methods by learning more effective hash functions, however, the hash ranking problem which is also very important is rarely studied. In traditional hashing based methods, data points in original space are embedded to Hamming space and the similarity between two binary codes is evaluated by Hamming distance. However, since the Hamming distances are discrete integer values, there are often more than one hash code sharing the same Hamming distance with the query point, which makes it ambiguous and leads to a critical issue for approximate nearest neighbor search when ranking is important. Hence, two groups of hash ranking methods are proposed recently to solve this problem. One group focuses on learning Weighted Hamming Distance by learning different weights for each bit or each class [39]–[41]. The other group aims at learning Asymmetric Distance which only binarizes the database points but not the query point [42], [43].

A. WEIGHTED HAMMING DISTANCE

1) BIT-LEVEL WEIGHTS

These hash codes ranking methods learn different weights for each hash bit. Query-sensitive Hash Code Ranking (QsRank) [39] takes the target neighborhood radius ϵ and the raw query point as input, learns weights for each bit, and then computes ranking score for each hash code.

In theory, they can calculate the scores by computing the probabilities of the points mapped to each hash code, which equals the percentages of the ϵ-neighbors of the query point mapped to each hash code.

In practice, to efficiently compute the scores, they propose an approximation algorithm. First, they only utilize the points in the ϵ-neighbors of the query point in a few top dimensions, which is reasonable because the data points in the original space are often first dimension-reduced by PCA before the projection stage which minimizes the reconstruction error with a few top dimensions. Then, since PCA dimensions are mutual uncorrelated, they approximate the score formulation by computing weights for each bit independently and multiply them together at last.

Another Weighted Hamming Ranking algorithm (WhRank) is proposed to rank the returned binary codes at a finer-grained binary code level [40]. The weights are composed of two parts: data-adaptive weights and query-sensitive weights. First, in the Offline stage, they assign different bit-level weights to different hash bits to give hashing based methods the ability to distinguish between the relative importance of different bits. Then, in the Online stage, they give an algorithm to learn a set of dynamic bit-level weights of hash bits for a given query.

In [40], they introduce a term 'discriminating power' to denote the ability of a hash function mapping similar data points to the same bit. The more discriminative the hash function is, the more important the corresponding bit is. Therefore, a larger weight will be assigned to it. Based on the definition of discriminating power, they use the distribution of database points around the training points to reveal how discriminative the hash bit is. As a result, the weight should be monotonically non-increasing w.r.t. standard deviations of the nearest neighbors' distributions around the training points.

Meanwhile, for a specific query point, the discriminating powers of different hash functions are also different. Intuitively, if the query point is near to the threshold, then after adding a random noise to it, it's more likely that the noisy query point flips on the opposite side of the threshold. Therefore, the weight should be monotonically non-decreasing w.r.t. the distance between the query point and the corresponding threshold.

2) CLASS-LEVEL WEIGHTS

These hash codes ranking methods learn different weights for each hash code or each class. [41] is the most representative one. They learn the suitable weights for each class by proposing a formulation that not only minimizes intra-class sample similarity but also maintains inter-class proximity. The objective function can be solved as a quadratic programming problem. Then, the query-adaptive weights are rapidly computed by evaluating the proximity between a query point and the concept categories.

B. ASYMMETRIC DISTANCE

As far as I know, [42] is the first method that proposes the asymmetric distance for hash codes ranking. They state that the asymmetric scheme which binarizes the database points but not the query point is still efficient but may provide superior accuracy, because they take advantage of more precise position information of the query point. Of course, it can also distinguish the hash codes with the same Hamming distance to the query point, since the asymmetric distance space is divided more densely than Hamming space.

The whole process can be briefly summarized as follows. They first compute two query-independent points that represent hash code 0 and 1 on each bit. Since they generally do not have access to the distribution of the database points, they simply approximate them by sample average. Then, for a query point, they first calculate the distances between the query point and the two query-independent points on each bit and then store them in look-up tables. At last, the asymmetric distance between the query point and the database points can be calculated by adding up some items of the tables according to their hash codes, which is very efficient.

IV. EXPERIMENTS

A. EXPERIMENTAL SETUP

We conduct many experiments on five common datasets: SIFT-10K, CIFAR-10, NUS-WIDE, MNIST and Caltech-256. We make an overview of the performance among some notable hash coding methods and make a large amount of experiments to compare the performance among several typical hash coding and hash ranking methods whose codes are open online. Now, let's describe these five datasets as well as three evaluation criteria used in the experiments(Fig. 4 and Fig. 6-Fig. 20) in detail.

SIFT-10K is a public benchmark for the evaluation of hash based ANN search, which is published by Hervĺę Jĺęgou [59].[1] It contains 35,000 128-dimensional vectors. The dataset is partitioned into a base set of 10,000 vectors and a training set of 25,000 vectors. Query set that contains 100 samples is selected from training set and the accurate 100 nearest neighbors of them are provided in a 100×100 matrix ('groundtruth'). We simply use the training set to learn the parameters and compute the average result of the 100 queries searching in the base set.

[1] You can download from http://corpus-texmex.irisa.fr/.

CIFAR-10 [60] is a subset of the Tiny Images dataset [61].[2] It consists of 60,000 images of size 32×32 pixels which have been grouped into 10 classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck). All of the images are identified with labels which can be used to evaluate the performance of hash based methods. Since the original images are 32×32 color images, they can be represented with grayscale GIST descriptors computed at 3 different scales (8,8,4), resulting in 320-dimensional vectors.

NUS-WIDE is a real-world web image database from National University of Singapore [62].[3] It includes 269,648 images with a total number of 5,018 unique tags. There are six types of low-level features extracted from these images: BoW, CH, CM55, CORR, WT and EDH. The images are manually assigned with some of the 81 concept tags. Since images are mostly associated with more than one label, one image is considered as the true nearest neighbor of the query if they contain at least one same label.

MNIST is a subset of a larger set of handwritten digits available from NIST [63].[4] It contains 70,000 images of size 28×28 pixels in which 10,000 are used as query set and the remaining 60,000 images are used as training set and base set. All of the images are represented by 784-dimensional signature vectors and marked with digital labels from 0 to 9.

Caltech-256 [64][5] is a collection of 30,607 images that can be divided into 257 classes (the last one is 'clutter'). We select 5 images from each class to form the query set with 1280 images. The remaining images are partitioned into two sets in which 5140 are randomly selected as the training set and the others are used as the database. In our experiments, we represent the images with 320-dimensional grayscale GIST descriptors [17]. We utilize the image labels to tell if the candidate is the true neighbor of the query.

Here, let us introduce the three evaluation criteria used to verify the accuracy of hashing based methods in the experiments: Precision@1, Precision and Mean Average Precision (MAP):

$$\text{Precision@1} = \frac{1}{\text{Number of the first true Neighbor}} \tag{9}$$

$$\text{Precision} = \frac{\text{Number of True Neighbors}}{N} \tag{10}$$

$$\text{MAP} = \frac{\sum_M \text{Precision}}{M} \tag{11}$$

where N is the number of points returned back and M is the number of true neighbors in the first N returned points.

[2] You can download from https://www.cs.toronto.edu/ kriz/cifar.html.

[3] You can download from http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm.

[4] You can download from http://yann.lecun.com/exdb/mnist/.

[5] You can download from https://authors.library.caltech.edu/7694/.

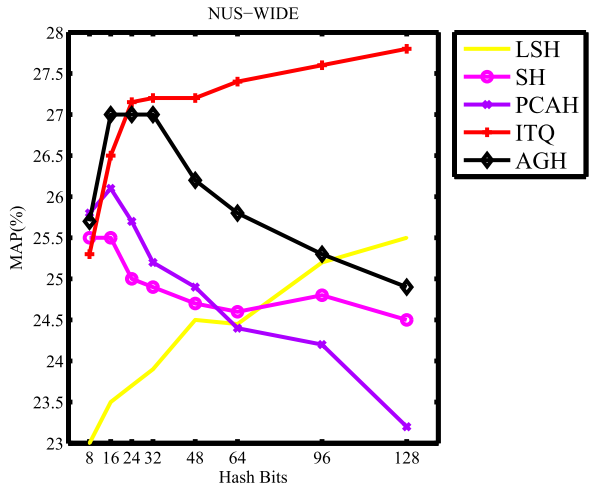

FIGURE 2. Euclidean-MAP.

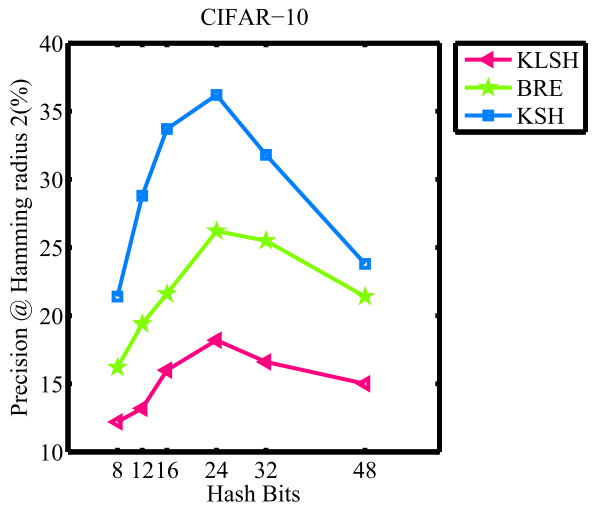

FIGURE 3. Kernel-Precision.

B. EXPERIMENTAL RESULTS AMONG HASH CODING METHODS

Fig. 2, Fig. 3 and Fig. 5 present the accuracy comparison among several notable hash coding methods in Euclidean space, Kernel space and Semantic space. The data in these three figures are from [23], [62], and [65]. The experimental settings in each figure are the same, so the results are convincing. The details of the experimental settings can be found in [23], [62], and [65]. We also conduct many experiments on almost all hashing methods whose codes are open online and create Fig. 4 and Fig. 6-Fig. 14.

1) EXPERIMENTAL ANALYSIS IN EUCLIDEAN SPACE

LSH [7] learns randomized hash functions to guarantee a high probability of collision for similar points. Since the theoretical guarantees are asymptotic as the number of random projections grows, the accuracy of LSH method is low with short codes and increases slowly with code length (Fig. 2, 4- 10). SH [8] learns data-dependent hash functions

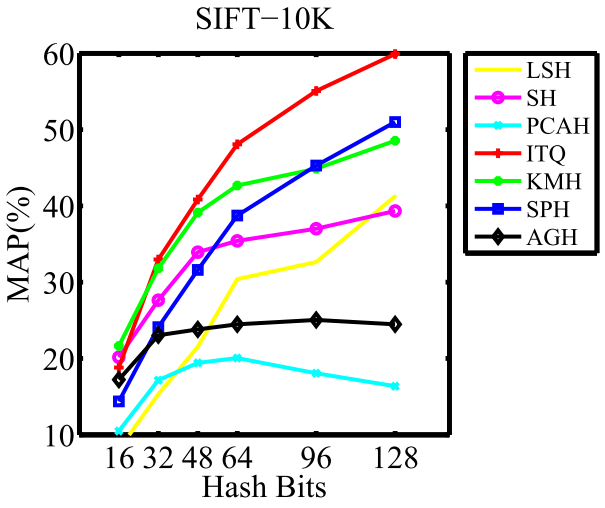

FIGURE 4. Euclidean-MAP.

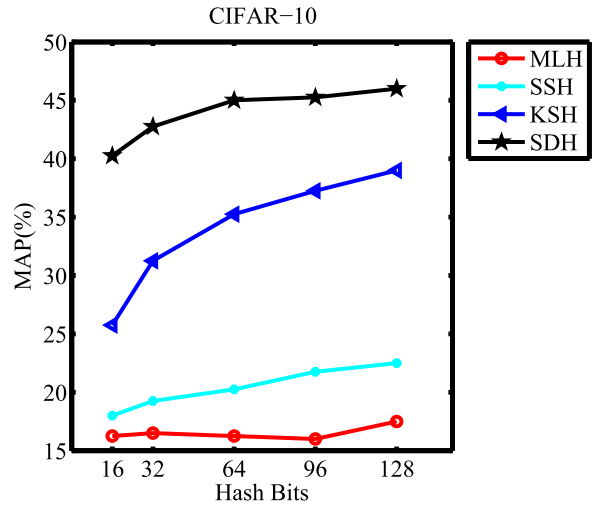

FIGURE 5. Semantic-MAP.

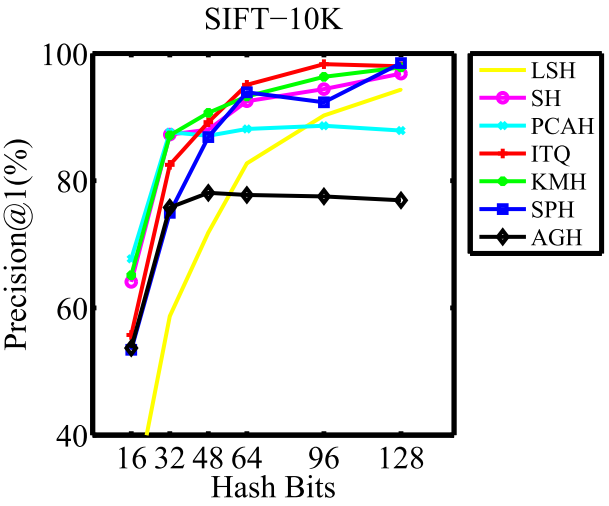

FIGURE 6. Euclidean-Precision@1.

and performs better than LSH method for small codes. From Fig. 2, we can see that longer codes can not increase the accuracy, because the projected data points are often distributed on

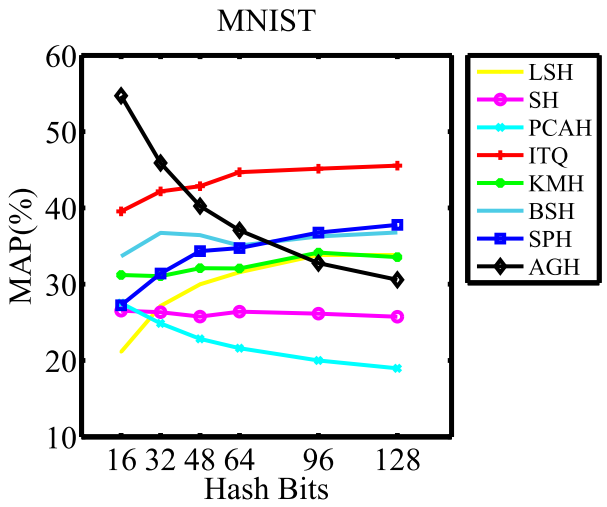

FIGURE 7. Euclidean-MAP.

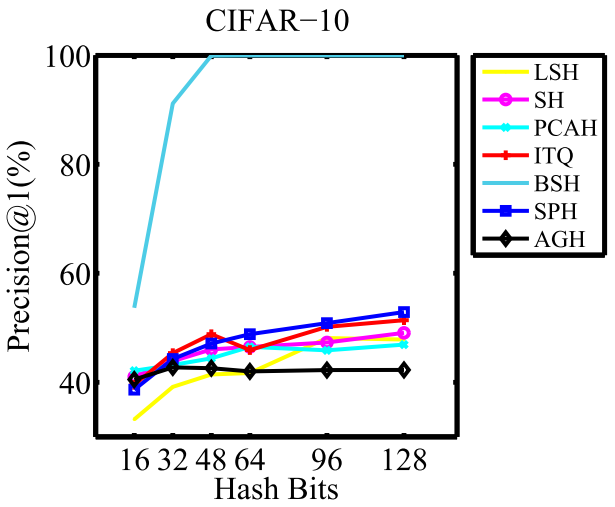

FIGURE 10. Euclidean-Precision@1.

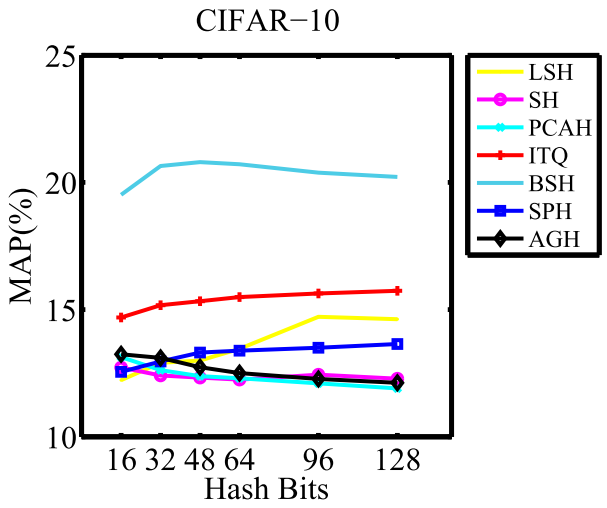

FIGURE 8. Euclidean-MAP.

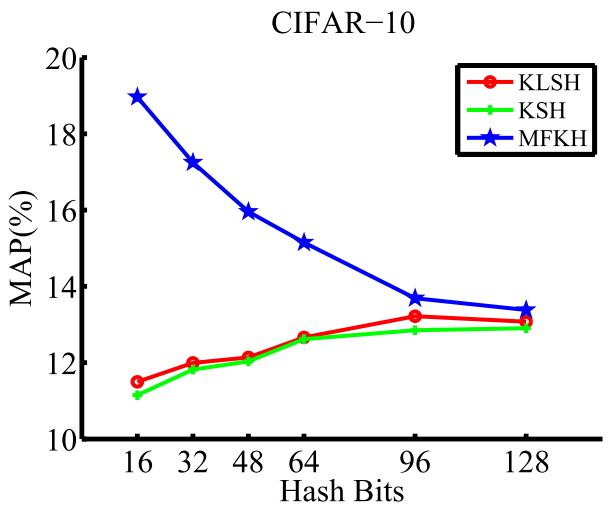

FIGURE 11. Kernel-MAP.

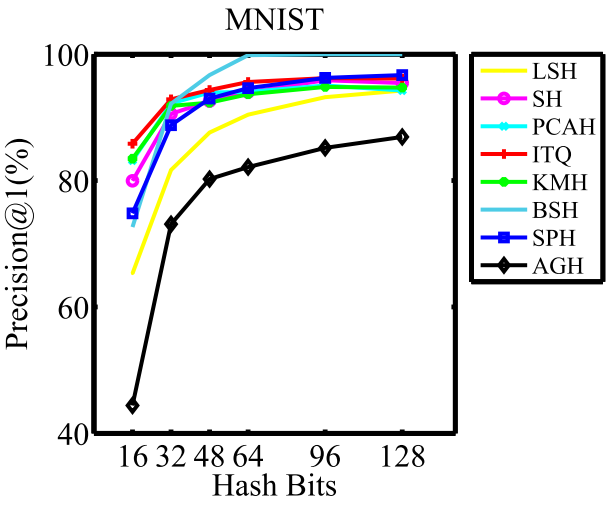

FIGURE 9. Euclidean-Precision@1.

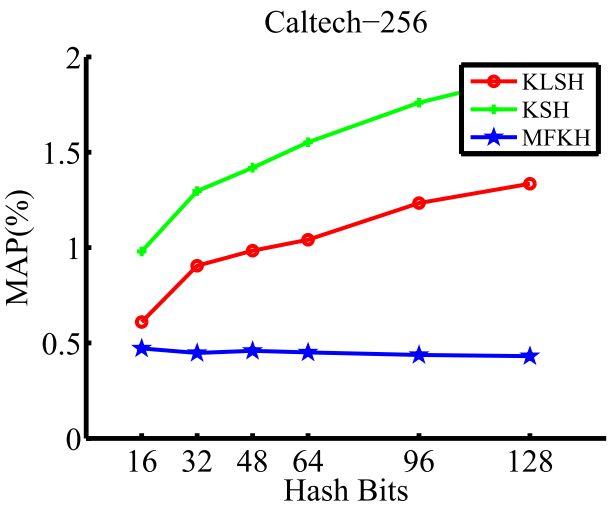

FIGURE 12. Kernel-MAP.

a few dimensions. PCAH [10] learns hash codes by randomly rotating data. With the rotation, the variance is balanced and quantization error is lower. However, PCAH [10] always gives low accuracy(Fig. 4- 10). Therefore, ITQ [10] learns an optimized rotation on its basis to let the quantization error be the lowest with an efficient alternating minimization scheme.

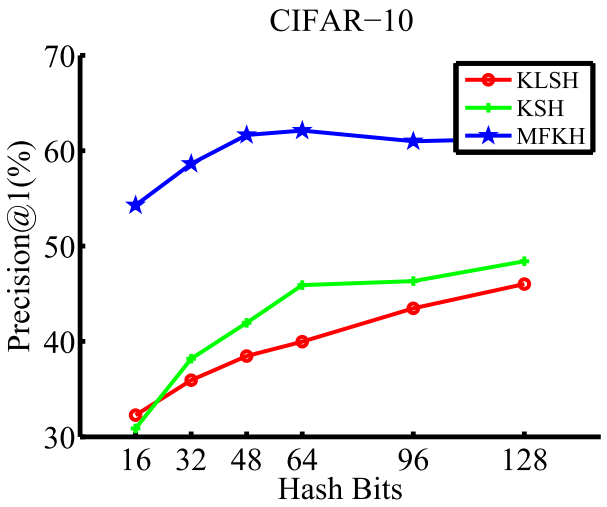

FIGURE 13. Kernel-Precision@1.

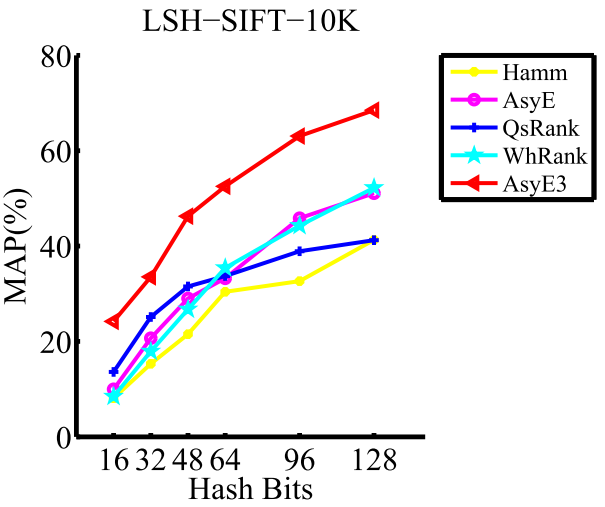

FIGURE 15. LSH-MAP.

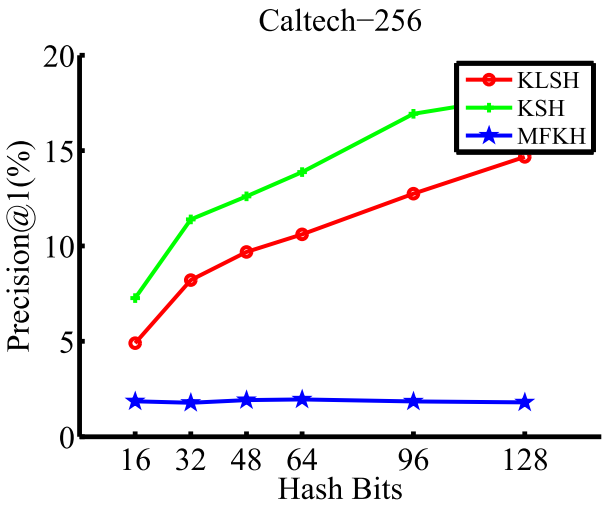

FIGURE 14. Kernel-Precision@1.

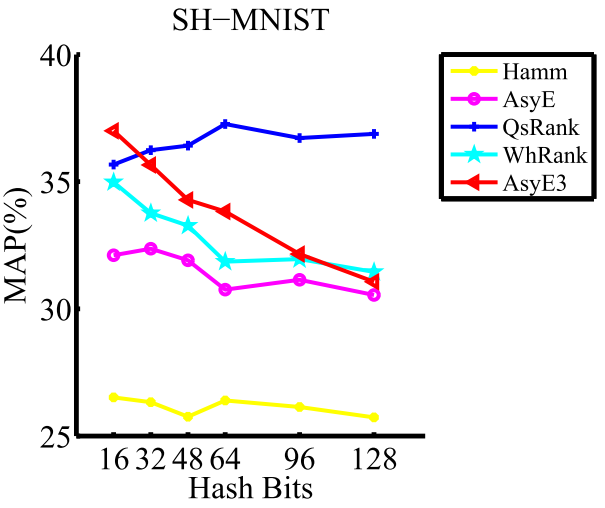

FIGURE 16. SH-MAP.

We can see ITQ always attain higher accuracy than PCAH (Fig. 2, 4,6- 10). AGH [29] aims at learning effective short codes by building an approximate neighborhood graph using Anchor Graphs to solve the objective function similar to that in SH. Therefore, we can see that AGH performs well for small codes(Fig. 2, 4, 6, 7- 9). SPH [66] proposes a novel hypersphere-based hashing function rather than hyperplane-based hashing functions to map more spatially coherent data points into a binary code. We can see from Fig. 4, 6- 10 that SPH [66] can always outperform hash coding methods mentioned above especially with longer code length. From Fig. 7- 10, we can clearly see BSH [67] provides much higher accuracy in image datasets such as CIFAR-10 and MNIST. This is because BSH [67] develops a Boosting-style algorithm for simultaneously optimizing the label subset and hashing function in a unified framework. BSH [67] designs each hashing function especially for a subset of labels, which can further enhance hashing efficiency for multi-label data. KMH [68] proposes a novel Affinity-Preserving K-means algorithm which simultaneously performs k-means clustering and learns the binary indices of the quantized cells. From Fig. 4, 6, 7, 9, we can find that KMH performs well in SIFT dataset but not well in image dataset(MNIST). It is possible that KMH is more applicable to non-image datasets.

2) EXPERIMENTAL ANALYSIS IN KERNEL SPACE

From Fig. 3, 11- 14, We can see KLSH [28] has low accuracy in general because it is based on LSH technique which formulates random projections as hash functions. However, KLSH can be applied to high-dimensional kernelized data when the underlying feature embedding for the kernel is unknown. BRE [9] learns hash functions by minimizing the reconstruction error between the original distances and the Hamming distances of the corresponding binary embeddings. It is also easily kernelized, but outperforms KLSH on all hash bits(Fig. 3), because BRE forms an objective function utilizing the distribution of training data and learns

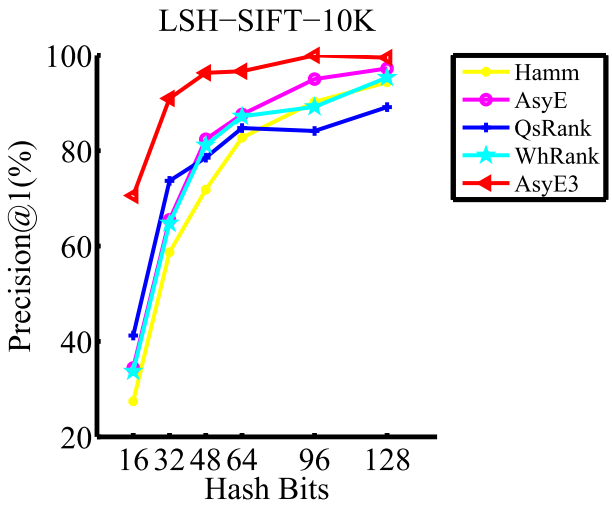

FIGURE 17. LSH-Precision@1.

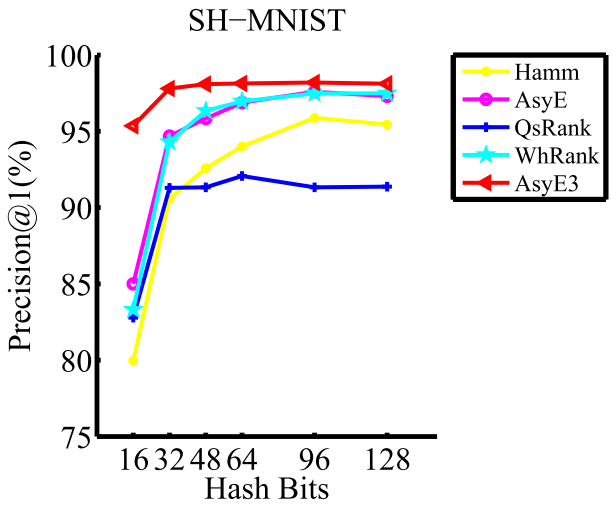

FIGURE 18. SH-Precision@1.

data-dependent hash functions rather than random projections in KLSH. KSH [23] learns hash functions in a more effective and efficient manner. We can see it attains dramatically higher accuracy compared with the other two hash coding methods in most instances(Fig 3, 11- 14). From Fig. 11 and Fig. 13, we can see MFKH [67] can achieve obvious performance gains over other kernel based hash coding methods(KLSH [28] and KSH [23]) because MFKH [67] proposes a novel hashing approach which utilizes the information conveyed by different features.

3) EXPERIMENTAL ANALYSIS IN SEMANTIC SPACE

MLH [11] learns hash functions based on structured prediction with latent variables and a hinge-like loss function. We can see MLH does not perform much better with more hash bits. However, it is efficient to be used for large datasets and scales well to large code lengths. SSH [12] utilizes the

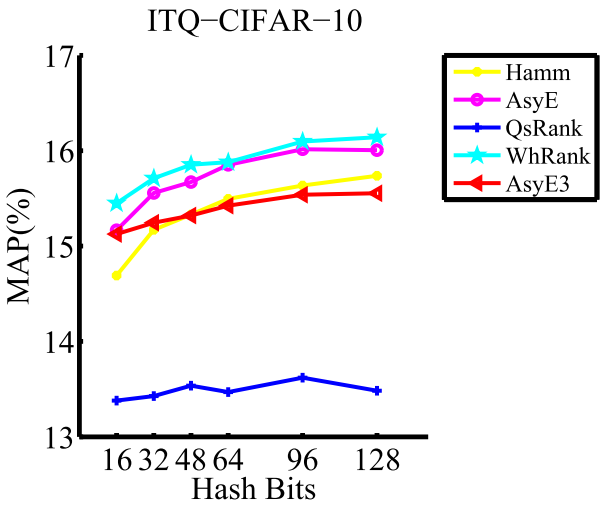

FIGURE 19. ITQ-MAP.

label information of the data points and learns hash codes that can maintain semantic similarity. SSH can achive superior performance over many unsupervised methods and some supervised methods such as MLH. KSH [23] is another effective supervised hash coding method. It can sequentially and efficiently train hash functions one bit at a time, yielding very short yet discriminative codes. We can see KSH outperforms MLH and SSH methods on all hash bits. SDH [62] proposes a novel supervised hashing framework, aiming at directly optimizing the binary hash codes efficiently and efficiently. SDH solves the discrete constraints imposed on the pursued hash codes by solving a sub-problem associated with NP-hard binary optimization. We can see from Fig. 4, SDH attains superior results over the other three supervised hashing methods.

C. EXPERIMENTAL RESULTS AMONG HASH RANKING METHODS

We conduct our experiments on SIFT-10K, MNIST and CIFAR-10 to show the performance comparison among typical hash ranking methods including Hamming distance (Hamm), Asymmetric Distance [42] (AsyE), Query-sensitive Ranking [39] (QsRank), Weighted Hamming distance [40] (WhRank) and Multiple Asymmetric Ranking [43] (AsyE3). From Fig. 15- 20, We can see AsyE and QsRank perform analogously and both get obvious higher accuracy than Hamm. This is because both of them can rerank the points sharing the same Hamming distance with the query point. WhRank can get higher accuracy than QsRank in most circumstances because WhRank learns bit-level weight in consideration of the distribution of data. AsyE3 [43] is a new asymmetric distance calculation method which learns more than two representative points for each hash bit. It performs much more better especially in non-image dataset because it divides the distance space more densely than AsyE, QsRank and WhRank(Fig. 15, 17).

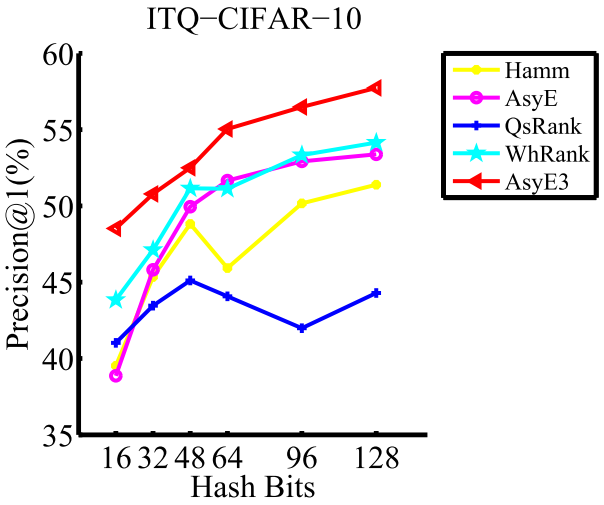

FIGURE 20. ITQ-Precision@1.

D. EXPERIMENTAL OVERVIEW ON HASH METHODS

In general, there are three main streams to increase hashing accuracy. One stream tries to construct more efficient hash functions, one stream aims at learning new quantization method subsequent to projection process, the other stream concentrates on reranking hash codes with new distances. There is also a new trend that many people try to lean hash codes with deep learning framework. In my opinion, there are still lots of details to be discovered in traditional hashing methods, which can keep the balance of time and accuracy of hashing methods skilly. Furthermore, new deep learning framework can be specially constructed for hashing methods which is a very promising topic. As for hash ranking methods, I think it is a very interesting topic, because it is easy to realize much more accurate ANN search with very simple operations. There fore, in our future work, we will also search for more efficient and fast hash ranking approaches.

V. CONCLUSION AND FUTURE WORK

In this paper, we illustrate the process of development of hashing based methods and propose an overall and a novel and clear category for them. We think this paper is suited for people to know hash quickly and find which branch they are interested in. We also conduct a set of experiments to compare the performance among almost all the existing hashing based methods whose codes are open online. Besides, we will continue our research in learning more efficient hash coding and hash ranking methods. Since approximate nearest search problem is involved in many domains, we will also explore to propose appropriate hash algorithms to various practical applications in our future work.

REFERENCES

[1] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *Comput. Surveys*, vol. 40, no. 2, 2008, Art. no. 5.
[2] J. Hayes and A. Efros, "Scene completion using millions of photographs," in *Proc. SIGGRAPH*, 2007, Art. no. 4.
[3] J. H. Friedman, J. L. Bentley, and R. A. Finkel, "An algorithm for finding best matches in logarithmic expected time," *ACM Trans. Math. Softw.*, vol. 3, no. 3, pp. 209–226, 1977.
[4] J. K. Uhlmann, "Satisfying general proximity/similarity queries with metric trees," *Inf. Process. Lett.*, vol. 40, pp. 175–179, Nov. 1991.
[5] A. Bygelzimer, S. Kakade, and J. Langford, "Cover trees for nearest neighbor," in *Proc. ICML*, 2006, pp. 97–104.
[6] W. Liu, H. Ma, H. Qi, D. Zhao, and Z. Chen, "Deep learning hashing for mobile visual search," *EURASIP J. Image Video Process.*, vol. 2017, p. 17, Dec. 2017.
[7] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. STOC*, 1998, pp. 604–613.
[8] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2008, pp. 1753–1760.
[9] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embedding," in *Proc. NIPS*, 2009, pp. 1042–1050.
[10] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. CVPR*, 2011, pp. 817–821.
[11] M. Norouzi and J. D. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.
[12] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.
[13] J. Wang, S. Kumar, and S.-F. Chang, "Sequential projection learning for hashing with compact codes," in *Proc. ICML*, 2010, pp. 1127–1134.
[14] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, in press, doi: 10.1109/TPAMI.2017.2678475.
[15] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Supervised discrete hashing with relaxation," *IEEE Trans. Neural Netw. Learn. Syst.*, in press, doi: 10.1109/TNNLS.2016.2636870.
[16] W. Kong and W.-J. Li, "Isotropic hashing," in *Proc. NIPS*, 2012, pp. 1646–1654.
[17] C. Strecha, A. Bronstein, M. Bronstein and P. Fua, "LDAHash: Improved matching with smaller descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, 2012.
[18] Y. Gong, S. Kumar, A. Henry Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proc. CVPR*, 2013, pp. 484–491.
[19] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, and D. Cai, "Harmonious hashing," in *Proc. IJCAI*, 2013, pp. 1820–1826.
[20] W. Kong and W.-J. Li, "Double-bit quantization for hashing," in *Proc. AAAI*, 2012, pp. 634–640.
[21] S. Moran, V. Lavrenko, and M. Osborne, "Variable bit quantisation for lsh," in *Proc. ACL*, 2013, pp. 753–758.
[22] J. He, W. Liu, and S.-F. Chang, "Scalable similarity search with optimized kernel hashing," in *Proc. KDD*, 2010, pp. 1129–1138.
[23] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, 2012, pp. 2074–2081.
[24] Y. Mu, J. Shen, and S. Yan, "Weakly-supervised hashing in kernel space," in *Proc. CVPR*, 2010, pp. 3344–3351.
[25] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. NIPS*, 2009, pp. 1509–1517.
[26] J. He, R. Radhakrishnan, S.-F. Chang, and C. Bauer, "Compact hashing with joint optimization of search accuracy and time," in *Proc. CVPR*, 2011, pp. 753–760.
[27] A. Joly and O. Buisson, "Random maximum margin hashing," in *Proc. CVPR*, Jun. 2011, pp. 873–880.
[28] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. CVPR*, 2009, pp. 2130–2137.
[29] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.
[30] Z. Jin *et al.*, "Complementary projection hashing," in *Proc. ICCV*, 2013, pp. 257–264.
[31] S. Kumar and R. Udupa, "Learning hash functions for cross-view similarity search," in *Proc. IJCAI*, 2011, pp. 1360–1365.
[32] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen, "Inter-media hashing for large-scale retrieval from heterogeneous data sources," in *Proc. SIGMOD*, 2013, pp. 785–796.
[33] G. Ding, Y. Guo, and J. Zhou, "Collective matrix factorization hashing for multimodal data," in *Proc. AAAI*, 2014, pp. 2075–2082.
[34] D. Wang, X. Gao, X. Wang, and L. He, "Semantic topic multimodal hashing for cross-media retrieval," in *Proc. CVPR*, 2015, pp. 3890–3896.

[35] L.-K. Huang, Q. Yang, and W.-S. Zheng, "Online hashing," in *Proc. IJCAI*, 2013, pp. 1422–1428.
[36] F. Cakir and S. Sclaroff, "Online supervised hashing," in *Proc. ICIP*, 2015, pp. 2606–2610.
[37] C. Leng, J. Wu, J. Cheng, X. Bai, and H. Lu, "Online sketching hashing," in *Proc. CVPR*, 2015, pp. 2503–2511.
[38] C. Leng, J. Wu, J. Cheng, X. Zhang, and H. Lu, "Hashing for distributed data," in *Proc. ICML*, 2015, pp. 1642–1650.
[39] X. Zhang, L. Zhang, and H. Shum, "QsRank: Query-sensitive hash code ranking for efficient ϵ-neighbor search," in *Proc. CVPR*, 2012, pp. 2058–2065.
[40] L. Zhang, Y. Zhang, J. Tang, K. Lu, and Q. Tian, "Binary code ranking with weighted Hamming distance," in *Proc. CVPR*, 2013, pp. 1586–1593.
[41] Y.-G. Jiang, J. Wang, and S.-F. Chang, "Lost in binarization: Query-adaptive ranking for similar image search with compact codes," in *Proc. ICMR*, 2011, Art. no. 16.
[42] A. Gordo, F. Perronnin, Y. Gong, and S. Lazebnik, "Asymmetric distances for binary embeddings," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 33–47, Jan. 2014.
[43] Y. Cao, H. Qi, K. Li, and M. Stojmenovic, "Multiple query-independent values based asymmetric ranking for approximate nearest neighbor search," in *Proc. ISPA*, Aug. 2016, pp. 1628–1635.
[44] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. CVPR*, 2015, pp. 27–35.
[45] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. CVPR*, 2015, pp. 2475–2483.
[46] K. L. A. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. CVPR*, 2016, pp. 1183–1192.
[47] R. Xia, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI*, 2014, pp. 2156–2162.
[48] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hashing via deep neural networks for large-scale image search," *CoRR*, vol. abs/1507.00101, 2015.
[49] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. CVPR*, 2015, pp. 1556–1564.
[50] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *Proc. ECCV*, 2016, pp. 216–234.
[51] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4766–4779, Dec. 2015.
[52] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. CVPR*, 2015, pp. 3270–3278.
[53] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. IJCAI*, 2016, pp. 1711–1717.
[54] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *Proc. AAAI*, 2016, pp. 2415–2421.
[55] R. Salakhutdinov and G. Hinton, "Semantic hashing," in *Proc. CVPR*, 2009, pp. 969–978.
[56] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proc. CVPR*, Jun. 2008, pp. 1–8.
[57] R. Udupa and M. Khapra, "Improving the multilingual user experience of Wikipedia using cross-language name search," in *Proc. NAACLHLT*, 2010, pp. 492–500.
[58] J. C. Platt, K. Toutanova, and W.-T. Yih, "Translingual document representations from discriminative projections," in *Proc. EMNLP*, 2010, pp. 251–261.
[59] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.
[60] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009.
[61] C. Redondo-Cabrera, R. J. López-Sastre, J. Acevedo-Rodriguez, and S. Maldonado-Bascon, "Surfing the point clouds: Selective 3D spatial pyramids for category-level object recognition," in *Proc. CVPR*, 2012, pp. 3458–3465.
[62] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. CVPR*, 2015, pp. 37–45.
[63] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[64] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," in *Proc. CVPR*, 2013.
[65] P. Zhang, W. Zhang, and W.-J. Li, "Supervised hashing with latent factor models," in *Proc. SIGIR*, 2014, pp. 173–182.
[66] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. CVPR*, 2012, pp. 2957–2964.
[67] X. Liu, J. He, B. Lang, and D. Liu, "Compact kernel hashing with multiple features," in *Proc. ACM MM*, 2012, pp. 881–884.
[68] 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.

YUAN CAO received the B.Sc. and B.E degrees in computer science from Dalian Maritime University, Dalian, China, in 2013. She is currently pursuing the Ph.D. degree in computer application technology with the Dalian University of Technology. Her research interests include large scale approximate nearest neighbor search and image retrieval.

HENG QI received the bachelor's degree from Hunan University in 2004 and the master's and Ph.D. degrees from the Dalian University of Technology, in 2006 and 2012, respectively. He is currently a JSPS Post-Doctoral Research Fellow with Nagoya University, Japan and also an Associate Professor with the School of Computer Science and Technology, Dalian University of Technology, China. His research interests include multimedia computing, big data computing, and cloud computing.

WENRUI ZHOU received bachelor's degree from Dalian Maritme University in 2015. She is currently pursuing the master's degree with the School of Computer Science and Technology, Dalian University of Technology, China. Her research interests include big data and hashing algorithms.

JIEN KATO received the M.E. and Ph.D. degrees in information engineering from Nagoya University in 1990 and 1993, respectively. She is currently an Associate Professor with the Graduate School of Informatics, Nagoya University. Her research interests include visual tracking, object/action recognition and detection, people re-identification, video summarization, and machine learning. She is a member of the IEICE, IPSJ, JSAI and ACM. She is a Senior Member of the IEEE Computer Society.

KEQIU LI (SM'12) received the bachelor's and master's degrees from the Department of Applied Mathematics, Dalian University of Technology, in 1994 and 1997, respectively, and the Ph.D. degree from the Graduate School of Information Science, Japan Advanced Institute of Science and Technology, in 2005. He also has two-year postdoctoral experience with The University of Tokyo, Japan. He is currently a Professor with the School of Computer Science and Technology, Dalian University of Technology, China. He has authored or co-authored over 200 technical papers, such as the IEEE/ACM TON, the IEEE TMC, the IEEE TPDS, and the ACM TOIT. His research interests include internet technology, data center networks, cloud computing, and wireless networks.

XIULONG LIU received the B.E. degree from the School of Software Technology, Dalian University of Technology, China, in 2010, where he is currently the Ph.D. degree with the School of Computer Science and Technology. He served as a Research Assistant with The Hong Kong Polytechnic University in 2014, and a Visiting Scholar with Temple University in 2015. His research interests include RFID systems and wireless sensor networks.

JIE GUI (SM'16) received the B.S. degree in computer science from Hohai University, Nanjing, China, in 2004, the M.S. degree in computer applied technology from the Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei, China, in 2007, and the Ph.D. degree in pattern recognition and intelligent systems from the University of Science and Technology of China, Hefei, China, in 2010.He is currently an Associate Professor with the Hefei Institute of Intelligent Machines, Chinese Academy of Sciences. He has been a Post-Doctoral Fellow with the National Laboratory of Pattern Recognition, Institute of Automation Chinese Academy of Sciences. His research interests include machine learning, pattern recognition, data mining, and image processing.