Toward Making Unsupervised Graph Hashing Discriminative

Chao Ma, Chen Gong, *Member, IEEE*, Xiang Li, Xiaolin Huang, *Senior Member, IEEE*, Wei Liu, *Member, IEEE*, and Jie Yang

***Abstract*—Recently, hashing has attracted much attention in visual information retrieval due to its low storage cost and fast query speed. The goal of hashing is to map original high-dimensional data into a low-dimensional binary-code space where the similar data points are assigned similar hash codes and dissimilar points are far away from each other. Existing unsupervised hashing methods mainly focus on recovering the pairwise similarity of the original data in hash space, but do not take specific measures to make the generated binary codes to be discriminative. To address this problem, this paper proposes a novel unsupervised hashing method, named "Discriminative Unsupervised Graph Hashing" (DUGH), which takes both similarity and dissimilarity of original data into consideration to learn discriminative binary codes. In particular, a probabilistic model is utilized to learn the encoding of original data in low-dimensional space, which models the original neighbor structure through both positive and negative edges in the KNN graph and then maximizes the likelihood of observing these edges. To efficiently and accurately measure the neighbor structure for large-scale datasets, we propose an effective KNN graph construction algorithm based on the random projection tree and neighbor exploring techniques. The experimental results on one synthetic dataset and four typical real-world image datasets demonstrate that the proposed method significantly outperforms the state-of-the-art unsupervised hashing methods.**

Manuscript received June 10, 2018; revised November 23, 2018; accepted July 22, 2019. Date of publication July 29, 2019; date of current version February 21, 2020. This work was supported in part by NSFC, China (No: 61602246, 61876107, U1803261, 61603248), in part by Committee of Science and Technology, Shanghai, China (No. 19510711200) and 973 Plan, China (No. 2015CB856004), in part by 1000-Talent Plan (Young Program), in part by NSF of Jiangsu Province (No: BK20171430), in part by the Fundamental Research Funds for the Central Universities (No: 30918011319), in part by the open project of State Key Laboratory of Integrated Services Networks (Xidian University, ID: ISN19-03), in part by the Summit of the Six Top Talents Program (No: DZXX-027), in part by the Innovative and Entrepreneurial Doctor Program of Jiangsu Province, in part by the Young Elite Scientists Sponsorship Program by Jiangsu Province, and in part by the Young Elite Scientists Sponsorship Program by CAST (No: 2018QNRC001). The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mohammed Daoudi. *(Corresponding author: Jie Yang; Xiaolin Huang.)*

C. Ma and X. Li are with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: sjtu_machao@sjtu.edu.cn; xx.lee@sjtu.edu.cn).

C. Gong is with the PCA Lab, the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: chen.gong@njust.edu.cn).

X. Huang and J. Yang are with the Institute of Image Processing and Pattern Recognition, Institute of Medical Robotics, Shanghai Jiao Tong University, Shanghai 200240, China (e-mail: xiaolinhuang@sjtu.edu.cn; jieyang@sjtu.edu.cn).

W. Liu is with the Tencent AI Lab, Shenzhen 518172, China (e-mail: wliu@ee.columbia.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2019.2931808

***Index Terms*—Unsupervised hashing, graph-based method, discrimination, probabilistic model.**

I. Introduction

RECENTLY, with the advance of computer technology and the development of the World Wide Web, a huge amount of digital data including texts, images and videos, are generated, stored, analyzed, and accessed every day. Nearest neighbor search is one of the critical techniques in many fields of information processing and analysis, such as data mining, information retrieval, and pattern recognition [1]–[5]. However, greedily searching the nearest neighbor in a large-scale dataset is infeasible. Due to this problem, approximate nearest neighbor (ANN) search has attracted much attention in a variety of areas, and many researches have been developed to handle the ANN search tasks. Hashing [1], [4], [6]–[27] is a widely-studied solution to ANN search, which tries to map original high-dimensional data into a low-dimensional binary-code space where the neighboring structure is preserved. Specifically, the Hamming distance between the binary codes of two data points should be small if they are similar, and the Hamming distance should be large if the two points are dissimilar.

Existing hashing methods can be divided into two categories: data-independent and data-dependent methods. For data-independent methods, such as Locality-Sensitive Hashing (LSH) and its variants [1], [28], [29], the hash functions are generated by using the random projection. Although LSH takes low computation complexity and is ensured to achieve high collision probability for similar data points, it requires relatively long hash codes to achieve high precision in practice, which leads to the increase of storage space and retrieval time cost.

The other category, data-dependent methods have been developed rapidly in recent years, as they can effectively index large-scale data with very compact binary codes. Unlike data-independent hashing methods that randomly select projection functions, data-dependent hashing methods attempt to learn parameters of different projection functions from a training set. Data-dependent methods can be unsupervised or supervised. Benefiting from the utilization of label information, supervised hashing methods have demonstrated to be promising in some applications [12], [13], [17]. However, it is often the case that the semantic labels are not available in many real-world applications, hence only unsupervised hashing can be performed, which is also the focus of this paper.

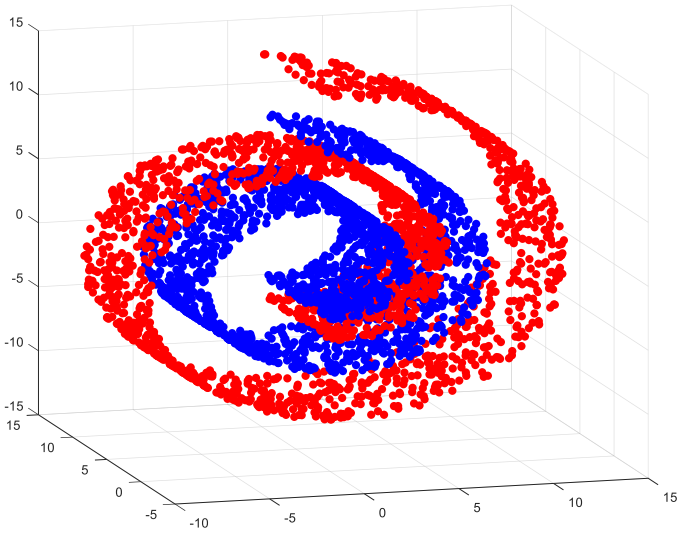


Fig. 1. Visualization of Two-SwissRolls dataset in 3-dimensional space.

Existing unsupervised hashing methods mainly utilize the data distribution or the underlying manifold structure to design effective indexing schemes [16], [23]. Although these methods have obtained promising results to some extent, they share two common problems. One basic problem is that all previous methods solely focus on recovering the pairwise similarity of original data in hash space, but they do not take specific measures to enforce the generated binary codes of dissimilar points to be dissimilar. In other words, the *discriminability* of the obtained binary codes has never been considered in unsupervised hashing methods, which results in undesirable results. For example, the 16-bit relaxed codes on a synthetic dataset called Two-SwissRolls (the dataset is shown in Fig. 1, and the details of this dataset are shown in Section V-B) learned by four graph-based unsupervised hashing algorithms (i.e., Spectral Hashing (SH) [6], Anchor Graph Hashing (AGH) [30], Inductive Manifold Hashing (IMH) [31], and our method) are visualized in Fig. 2. As we can see, due to the neglect of the dissimilarity relationship between original data points, SH, AGH, and IMH are not able to push dissimilar points far away from each other in hash space, therefore the obtained binary codes cannot accurately reflect the neighboring structure of original data. Therefore, preserving the dissimilarity of the original data is indeed important in hashing task. Another problem of existing hashing methods is how to effectively measure the similarity and dissimilarity relationship between the original data, since directly searching the neighbors of all points for large-scale datasets is too time-consuming. Several existing methods [30], [32] adopt anchor graph [33] technique to approximate the exact graph, where two points are considered to be similar if they share at least one common anchor point. However, it cannot be guaranteed that the nearby points in the original feature space will always have identical nearest anchors [34], and the widely utilized kernel-defined local weights in these methods are also sensitive to the variations of hyper-parameters [35]. As a result, the original neighboring structure may be destroyed in the established anchor graph. The above mentioned two bottlenecks have seriously limited the utilization of existing unsupervised hashing algorithms in practice.

In order to address the above shortcomings, we propose a novel unsupervised hashing method named "Discriminative Unsupervised Graph Hashing" (DUGH), which aims at learning both *accurate* and *discriminative* binary codes for large-scale hashing tasks. Specifically, in order to keep similar data points close and dissimilar points far away from each other in hash space, the low-dimensional encoding of the original data points is learned through a probabilistic method. In particular, our method models both the positive and negative edges (here negative edges mean the vertices[1] that are not linked by an edge) in the KNN graph of the original data, and then maximizes the likelihood of observing these edges in hash space. Furthermore, in order to efficiently and accurately measure the similarity and dissimilarity relationship inherent in the original data, we propose an efficient graph construction algorithm to build the sparse KNN graph based on the random projection (RP) tree [36] and neighbor exploring [37] techniques. The experimental results on one synthetic dataset and four typical image datasets demonstrate that the proposed method significantly outperforms the state-of-the-art unsupervised hashing methods.

The rest of this paper is organized as follows. We briefly introduce some related works in Section II. The details of our method are shown in Section III. The out-of-sample extension of our method is presented in Section IV. Our approach is empirically evaluated in Section V. Finally, we conclude the entire paper in Section VI.

II. Related Works

In this section, we briefly review some related works in the recent hashing literatures. Existing hashing algorithms can be roughly divided into two groups: data-independent and data-dependent.

A. Data-Independent Hashing

The most well-known data-independent hashing technique is Locality-Sensitive Hashing (LSH) [1]. LSH simply utilizes random linear projection to map the original data close in the Euclidean space to similar codes, and theoretical analyses illustrate that as the code length increases, the Hamming distance between two codes will approach to the Euclidean distance between their corresponding data points. With the success of LSH, random projection based hash functions have been extended to various similarity measures, including p-norm distance [28], Mahalanobis metric [38], and kernel similarity [39], [40], and these methods have been successfully utilized for large-scale image retrieval and classification. However, in realistic applications, LSH-related methods require long hash codes to achieve high precision, which results in low recall because the collision probability that two points fall into the same bucket decreases as the code length increases [41].

[1] In this paper, the terms "vertex", "point" and "example" have the same meaning.

B. Data-Dependent Hashing

To solve the problem of data-independent hashing methods, data-dependent hashing methods are proposed to learn effective but compact hash codes through a set of training data. Desirable hashing functions can be effectively learned by mining the structure of the original data points, representing the structure on the objective function, and solving optimization problems associated with the objective function. Therefore, data-dependent methods are also called "Learning to Hash" (L2H) [4]. L2H methods can be further divided into two groups: unsupervised [6], [7], [16], [23], [24], [31], [42] and supervised [8], [12]–[14], [17], [43]. Supervised hashing algorithms employ various supervision information (labels or tags) to improve the performance of learned binary codes, and have demonstrated to be promising to some extent. However, the lack of the semantic labels in most real applications limits the utilization of these methods. Therefore, we mainly focus on unsupervised hashing in this paper.

Unsupervised hashing algorithms mainly learn hash functions through the data distribution of training set. For example, Principal Component Analysis based Hashing (PCAH) [7] generates linear hash functions through simple PCA projection, which performs better than random projection. Gong *et al.* [7] developed the Iterative Quantization (ITQ) method, which employs a simple and efficient alternating minimization scheme for finding an orthogonal rotation of zero-centered data to minimize the quantization error. Except these two PCA-based hashing methods, Heo *et al.* [42] proposed a novel hypersphere-based hashing method, Spherical Hashing (SpH), which is able to map more spatially coherent data points into binary codes compared to hyperplane-based methods. Graph techniques [33], [44]–[48] have develop rapidly and have been widely utilized in hashing methods. For instance, Spectral Hashing (SH) [6] is a representative unsupervised hashing method, which learns similarity preserved hash codes by spectral decomposition on Laplacian matrix with the balanced and uncorrelated constraints. Liu *et al.* [30] proposed the Anchor Graph-based Hashing (AGH) method, which automatically discovers the neighboring structure inherent in the original data to learn binary codes. In AGH, the anchor graph technique [33] is employed to make such approach computationally feasible. Shen *et al.* [31] proposed the Inductive Manifold Hashing (IMH) method, which attempts to learn compact embedding on the intrinsic manifolds of original data. More recently, hashing with Binary Autoencoders (BA) algorithm is proposed to combine the data dimension reduction and binary quantization into a single step by using autoencoder, where the algorithm encourages similar inputs map to similar binary codes. Song *et al.* [21] proposed a unsupervised deep video hashing method which learns the video hash codes by simultaneous reconstructing the video contents and neighborhood structure. Shen *et al.* [23] proposed an unsupervised hashing framework, which can directly handle the binary constraints with a general hashing loss.

III. Methodology

Suppose we have an image collection $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i \in \mathbb{R}^d$ is the d-dimensional feature vector of the i-th image example. Without loss of generality, the data are assumed to be zero centered which means $\sum_{i=1}^N \mathbf{x}_i = \mathbf{0}$. In general, the goal of hashing is to map each point $\mathbf{x}_i$ into a binary code $\mathbf{b}_i \in \{-1, 1\}^c$, where c denotes the code length. Ideally, we hope that the Hamming distance between $\mathbf{b}_i$ and $\mathbf{b}_j$ should be small when the two data points are similar in the original space, while the Hamming distance should be large when they are dissimilar. To solve this problem, an approximate KNN graph should be constructed, then we encode the obtained KNN graph into low-dimensional hash space where the neighboring structure is preserved. The proposed algorithm DUGH is detailed as follows.

A. Graph Construction

As we know, constructing the exact KNN graph takes $O(N^2 d)$ time, which is too time-consuming in real cases. Therefore, various indexing techniques have been proposed to approximate the KNN graph [1], [28], [36], [37], [49]–[51]. Among these techniques, RP tree [36] technique has been proved to be efficient for ANN search in high-dimensional space. However, constructing a KNN graph with high accuracy requires many trees, which significantly hurts the model efficiency. Therefore, in this paper, we propose to construct an approximate KNN graph $G = \{V, E\}$ with limited number of RP trees and then improve the accuracy of the graph via using the idea of neighbor exploring [37], [52].

Firstly, we build up N_T RP trees to construct the initial neighbor graph. Specifically, for every non-leaf node of each tree, the algorithm selects a random hyperplane to split the subspace corresponding to the non-leaf node into two, which become the children of that node. This process continues until the depth of the tree reaches a threshold. Once the RP trees are constructed, the corresponding leaf node of each point can be found through traversing in the trees. The points in the subspaces of the same leaf nodes will be treated as the candidates of their neighbors, and the K-nearest neighbors in all the candidates of N_T RP trees are utilized for initializing the KNN graph.

Secondly, starting from the initial KNN graph, the neighbor exploring technique is employed to iteratively update the graph. The key idea of neighbor exploring is that "a neighbor of my neighbor is also likely to be my neighbor" [37]. During each iteration, we search the neighbors of the neighbors for each data point, and the union of its neighbors and its neighbors' neighbors forms the candidate set of its "new" neighbors. We then update the KNN graph by searching the K-nearest neighbors in this candidate set. This neighbor exploring is repeated for T times to improve the accuracy of the approximate KNN graph.

After that, we calculate the weights of the edges using the same approach as [53]. For a pair of points $(\mathbf{x}_i, \mathbf{x}_j)$ which are connected in the obtained graph, the weight w_{ij} is defined as

$$w_{ij} = \frac{p_{j|i} + p_{i|j}}{2N}, \tag{1}$$

where $p_{j|i}$ denotes the conditional probability and it is defined as

$$p_{j|i} = \frac{e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma_i^2}}}{\sum_{(i,k)\in E} e^{\frac{-\|\mathbf{x}_i - \mathbf{x}_k\|^2}{2\sigma_i^2}}}, \tag{2}$$

Algorithm 1: Algorithm for Graph Construction

Input: Training set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, the number of RP trees N_T, the number of the nearest neighbors in KNN graph K, the maximum iterations for neighbor exploring T.

Output: Approximate KNN graph $G = \{V, E\}$.

```
Build N_T RP trees on X;
// Initialize the KNN graph
for each point x_i ∈ X do
    Search the K-nearest neighbors based on RP trees, and then store the results in the set K_i;
end for
// Neighbor exploring
t = 1;
while t ≤ T do
    for each point x_i ∈ X
        Search the candidate set of its "new" neighbors C_i = K_i ∪ (⋃_{x_j ∈ K_i} K_j);
        Search the K-nearest neighbors of x_i in the candidate set C_i, and then update K_i;
    end for
    t = t + 1;
end while
for each point x_i ∈ X, x_j ∈ K_i
    Add edge e_ij into KNN graph G;
    Calculate the weight w_ij of the edge e_ij according to Eqn. (1);
end for
```

where E denotes the edge set and the parameter σ_i is set in such a way that the perplexity of the conditional distribution $p_{\cdot|i}$ equals to a predefined perplexity.

The entire procedure of graph construction is briefly summarized in Algorithm 1.

B. Encoding the Graph

Once the KNN graph is constructed, we would like to encode the graph into a low-dimensional space to learn the relaxed vectors of data, and then quantize the relaxed vectors into binary codes. In this paper, we take the discrimination of data points into consideration and solve the encoding problem through two aspects: 1) we utilize a Student t-distribution function to calculate the probability of observing the edges in the KNN graph; and 2) a probabilistic method is adopted to learn the encoding of the original data, which models both the positive and negative edges in the KNN graph. We detail the entire procedure as follow.

Given a pair of points $(\mathbf{x}_i, \mathbf{x}_j)$, we first define the probability of observing a binary edge $e_{ij} = 1$ between $\mathbf{x}_i$ and $\mathbf{x}_j$ as

$$P(e_{ij} = 1) = f(\|\mathbf{u}_i - \mathbf{u}_j\|), \tag{3}$$

where $\mathbf{u}_i \in \mathbb{R}^c$ is the relaxed code of point $\mathbf{x}_i$ in the low-dimensional space, and $f(\cdot)$ is a probabilistic function w.r.t. the distance between $\mathbf{u}_i$ and $\mathbf{u}_j$. The function $f(x) = 1/(1 + x^2)$ is employed in this paper, as it specifies a Student t-distribution and is able to deal with the "crowd problem" according to [53]. This is because this probabilistic function is insensitive to the scale change of the distance between the encoding results. As shown in Fig. 3, we perform the comparison between the t-distribution and the widely utilized Gaussian distribution in hash tasks. As we can see, if we enforce the probabilities of observing an edge in the graph under two different distribution functions to be equal, the distance between the encoding results under the t-distribution is supposed to be smaller than that under the Gaussian distribution when the original points are similar, otherwise the distance under the t-distribution is supposed to be larger than that under the Gaussian distribution when the original points are dissimilar. This property is extremely important for achieving the discriminative hashing results.

Eqn. (3) only defines the probability of observing a binary edge between a pair of data points. To further extend it to general weighted edges, we define the likelihood of observing a weighted edge $e_{ij} = w_{ij}$ as

$$P(e_{ij} = w_{ij}) = P(e_{ij} = 1)^{w_{ij}}. \tag{4}$$

With the above definition and the weighted graph $G = \{V, E\}$, the likelihood of observing all the edges in the graph can be calculated as

$$\begin{aligned}\mathcal{L} &= \prod_{(i,j)\in E} p(e_{ij} = 1)^{w_{ij}} \prod_{(i,j)\in \bar{E}} (1 - p(e_{ij} = 1))^{\gamma} \\ &\propto \sum_{(i,j)\in E} w_{ij} \log\, p(e_{ij} = 1) \\ &\quad + \sum_{(i,j)\in \bar{E}} \gamma \log\,(1 - p(e_{ij} = 1)),\end{aligned} \tag{5}$$

in which $\bar{E}$ is the set of vertex pairs without edges and γ is an unified weight assigned to the negative edges. Eqn. (5) models the likelihood of observing all the edges in the graph through two parts. The first part $\sum_{(i,j)\in E} w_{ij} \log\, p(e_{ij} = 1)$ models the likelihood of observing all the positive edges. By maximizing this part, similar data points will be put close together in the low-dimensional space. The second part $\sum_{(i,j)\in \bar{E}} \gamma \log\,(1 - p(e_{ij} = 1))$ models the likelihood of observing all the vertex pairs without edges (i.e., negative edges). By maximizing this part, dissimilar data points will be pushed away from each other.

By maximizing Eqn. (5), the relaxed code of each point can be learned. However, directly solving Eqn. (5) with stochastic gradient descent (SGD) algorithm is intractable, as the number of negative edges is quadratic to the number of nodes. To address this bottleneck, negative edge sampling technique [54] is adopted for optimization. For each vertex $i \in \mathbf{V}$, we randomly sample several vertices $j \in \mathbf{V}$ according to a noisy distribution $P_n(j) \propto d_j^{0.75}$ and treat $\{i, j\}$ as the negative edges, where d_j is the degree of vector j. The objective function (5) can be reformulated as

$$\begin{aligned}\mathcal{L}' &= \sum_{(i,j)\in E} w_{ij} (\log\, p(e_{ij} = 1) \\ &\quad + \sum_{k=1}^{M} E_{j_k \sim P_n(j)} \gamma \log\,(1 - p(e_{ij} = 1))),\end{aligned} \tag{6}$$

Algorithm 2: The training procedure of DUGH

Input: Training set $\mathcal{X}$, code length c.
Output: The binary codes $\mathbf{B}$.

1: Construct the approximate KNN graph according to Algorithm 1;
2: Encode the KNN graph into the low-dimentional space by maximizing Eqn. (5);
3: Initialize the $\mathbf{R}$ with a random orthogonal matrix;
4: Minimize Eqn. (7) by alternatively updating $\mathbf{B}$ and $\mathbf{R}$ while fixing the other one;
5: Calculate the binary codes as $\mathbf{B} = sgn(\mathbf{UR})$.

where M is the number of negative edges for each positive edge. The objective function (6) is still problematic to be optimized by SGD. This is because that the weight of edges w_{ij} is multiplied into the gradient, and as the value of w_{ij} varies in a wide range, it is very difficult to choose a suitable learning rate. To solve this problem, the approach of edge sampling [52], [55] is used in our paper, in which the edges are randomly sampling with the probability proportional to their weights and then the sampling edges are treated as binary edges. As a result, the learning process is robust to the variations of the weights. With all the above mentioned techniques, the entire optimization problem can be solved efficiently with asynchronous stochastic gradient descent [56].

After obtaining the relaxed codes $\mathbf{U} = [\mathbf{u}_1, \ldots, \mathbf{u}_N]^T$, our next goal is to find the binary codes $\mathbf{B} = [\mathbf{b}_1, \ldots, \mathbf{b}_N]^T$. Directly thresholding the continuous vectors into binary codes would lead to large quantization errors, and the performance might deteriorate as the code length c increases. Therefore, the orthogonal transformation is employed to minimize the quantization loss following [7]. The quantization loss is defined as

$$\min_{\mathbf{B},\mathbf{R}} \|\mathbf{B} - \mathbf{UR}\|_F^2,$$
$$\text{s.t.}\, \mathbf{B} \in \{-1, 1\}^{N\times c}, \mathbf{R}^T\mathbf{R} = \mathbf{I}, \tag{7}$$

where $\mathbf{R} \in \mathbb{R}^{c\times c}$ is the orthgonal matrix. To solve Eqn. (7), we firstly initialize $\mathbf{R}$ with a random orthogonal matrix, and then iteratively minimize the quantization loss in an alternating procedure. The details are explained below.

Fix $\mathbf{R}$ and update $\mathbf{B}$. It is obvious that the optimal solution is $\mathbf{B} = sgn(\mathbf{UR})$.

Fix $\mathbf{B}$ and update $\mathbf{R}$. While fixing $\mathbf{B}$, Eqn. (7) corresponds to the classic Orthogonal Procrustes Problem (OPP) [57]. It can be minimized by firstly computing the SVD of the matrix $\mathbf{B}^T\mathbf{U}$ as $\mathbf{B}^T\mathbf{U} = \mathbf{S}\Omega\hat{\mathbf{S}}^T$, and then setting $\mathbf{R} = \hat{\mathbf{S}}\mathbf{S}^T$.

We briefly summarize the entire training procedure of our method in Algorithm 2.

IV. Out-of-Sample Extension

For a new coming data point $\mathbf{x}_q$, the binary code $\mathbf{b}_q$ should be derived efficiently. By following [31], we solve this problem by utilizing a two-stage strategy: firstly we generate $\mathbf{u}_q$ through its s-nearest neighbors $\mathcal{N}_s(\mathbf{x}_q)$ in the training set $\mathcal{X}$, and then quantize it into the binary code $\mathbf{b}_q$ with the learned orthogonal transformation. This two-stage procedure can simultaneously preserve the neighboring structure of the query points and minimize the quantization loss.

The first problem for generating the embedding $\mathbf{u}_q$ is how to identify $\mathcal{N}_s(\mathbf{x}_q)$. In this paper, we search the approximate s-nearest neighbors of $\mathbf{u}_q$ through clustering algorithm such as K-means. We firstly partition the training set $\mathcal{X}$ into M clusters offline. For a query $\mathbf{x}_q$, we can find its nearest cluster (i.e., the distance from the query to the centroid of the selected cluster is the smallest) and then retrieve the s-nearest neighbors in this cluster to form $\mathcal{N}_s(\mathbf{x}_q)$.

After that, we minimize the following objective to learn $\mathbf{u}_q$, namely

$$\min_{\mathbf{u}_q} \sum_{\mathbf{x}_i\in\mathcal{N}_s(\mathbf{x}_q)} w(\mathbf{x}_q, \mathbf{x}_i)\|\mathbf{u}_q - \mathbf{u}_i\|^2. \tag{8}$$

Here we define $w(\mathbf{x}_q, \mathbf{x}_i)$ as $w(\mathbf{x}_q, \mathbf{x}_i) = \exp(-\|\mathbf{x}_q - \mathbf{x}_i\|^2/\sigma^2)$, where σ is the bandwidth parameter. Simply, the optimal solution is calculated as

$$\mathbf{u}_q^* = \frac{\sum_{i=1}^s w(\mathbf{x}_q, \mathbf{x}_i)\mathbf{u}_i}{\sum_{i=1}^s w(\mathbf{x}_q, \mathbf{x}_i)},\ \mathbf{x}_i \in \mathcal{N}_s(\mathbf{x}_q). \tag{9}$$

The hash code for the query $\mathbf{x}_q$ is then calculated by

$$\mathbf{b_q} = sgn(\mathbf{R}^T\mathbf{u}_q^*). \tag{10}$$

V. Experimental Results

This section presents the experimental results and demonstrates the effectiveness of the proposed approach. We start by introducing the details of our experimental setting, and then provide the comparison results of our method with typical existing approaches.

A. Experimental Setting

In the following experiments, we compare the proposed DUGH method with several state-of-the-art hashing methods, which include: Locality-Sensitive Hashing (LSH) [1], Spectral Hashing (SH) [6], Anchor Graph Hashing (AGH) [30], Inductive Manifold Hashing (IMH) [31], Principal Component Analysis based Hashing (PCAH) [7], Iterative Quantization (ITQ) [7], Spherical Hashing (SpH) [42], Asymmetric Inner-product Binary Coding (AIBC) [16], and Similarity-Adaptive Deep Hashing (SADH) [23]. As we all know, convolutional neural network (CNN) has emerged as the state-of-the-art method for global descriptors in image retrieval tasks [13], [58], but it is not the focus of our paper to compare the description ability of the CNN features and hand-crafted features. Therefore, in this paper, we do not compare our method with unsupervised deep hashing methods based on CNN architectures such as [59], [60]. Furthermore, to illustrate the advantage of the quantization strategy, we show the performance of the proposed DUGH method without the stage of quantization (represented as DUGH_0 in the experiments).

We run the previous methods using publicly available codes with suggested parameters in their papers. Specifically, for our

TABLE I
THE COMPARISON BETWEEN DUGH AND SEVERAL TYPICAL UNSUPERVISED HASHING ALGORITHMS WITH THE HASH CODES OF DIFFERENT LENGTHS ON TWO-SWISSROLLS DATASET. THE BEST RESULT UNDER EACH SETTING IS SHOWN IN BOLDFACE

Method	mAP					F1 score					Precision@500				
	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits
DUGH	**0.5859**	**0.5915**	**0.5859**	0.5770	0.5774	0.2219	**0.1987**	**0.1696**	**0.1241**	**0.1097**	**0.5584**	**0.5588**	**0.5579**	0.5501	0.5457
$DUGH_0$	0.5554	0.5742	0.5703	**0.5860**	**0.5854**	**0.3836**	0.1905	0.0577	0.0384	0.0139	0.5452	0.5567	0.5518	**0.5607**	**0.5630**
LSH	0.5043	0.5071	0.5053	0.5055	0.5049	0.1873	0.0516	0.0324	0.0220	0.0058	0.5032	0.5102	0.5058	0.5066	0.5039
SH	0.5105	0.5097	0.5095	0.5083	0.5062	0.0387	0.0035	0.0008	0.0003	0.0000	0.5130	0.5123	0.5128	0.5120	0.5096
AGH	0.5044	0.5034	0.5043	0.5041	0.5033	0.0623	0.0125	0.0067	0.0045	0.0018	0.5036	0.5017	0.5040	0.5030	0.5008
IMH	0.5031	0.5038	0.5035	0.5035	0.5040	0.2110	0.0671	0.0331	0.0274	0.0072	0.5005	0.5025	0.5012	0.5021	0.5025
PCAH	0.5010	0.5014	0.5013	0.5012	0.5010	0.0041	0.0000	0.0000	0.0000	0.0000	0.4997	0.5005	0.5009	0.5001	0.4996
ITQ	0.5016	0.5025	0.5015	0.5019	0.5023	0.2553	0.1335	0.1187	0.1198	0.0625	0.5005	0.5005	0.5004	0.5003	0.4998
SpH	0.5047	0.5048	0.5063	0.5049	0.5057	0.1588	0.0624	0.0339	0.0225	0.0055	0.5059	0.5046	0.5095	0.5054	0.5069
AIBC	0.5014	0.5014	0.5014	0.5014	0.5014	0.2410	0.1451	0.1064	0.0812	0.0428	0.4995	0.4998	0.4998	0.4993	0.4995
SADH	0.5018	0.5022	0.5015	0.5025	0.5023	0.0065	0.0000	0.0000	0.0000	0.0000	0.5003	0.5028	0.5002	0.5034	0.5029

DUGH, the parameters are set as follows unless otherwise specified. The number of the RP trees is set to be $N_T = 10$. The number of neighbors for each data point in the KNN graph is decided as $K = 150$. We repeat the neighbor exploring for three iterations to improve the accuracy of the KNN graph, as it has shown satisfactory performance as discussed in Section V-G. In the out-of-sample stage, the number of K-means clusters is tuned to $M = 300$, and the number of the nearest neighbors is determined as $s = 5$.

By following the evaluation protocols used in previous hashing methods [4], we split each dataset into training set and test set. We evaluate all mentioned methods by *Hamming ranking* and *hash lookup* using 16 to 128 hash bits. For *Hamming ranking*, we perform mean Average Precision (mAP), which is a widely used metric for evaluating the performance of hashing methods. We also use the average precision of first N ranked images (Precision@N) for each query to measure the *Hamming ranking* performance. For *hash lookup*, we measure the results by F1 score within Hamming radius 2 [61], which is calculated as $\mathrm{F1} = 2(precision \cdot recall)/(precision + recall)$. The precision-recall curve is also employed to measure the performance in the experiments.

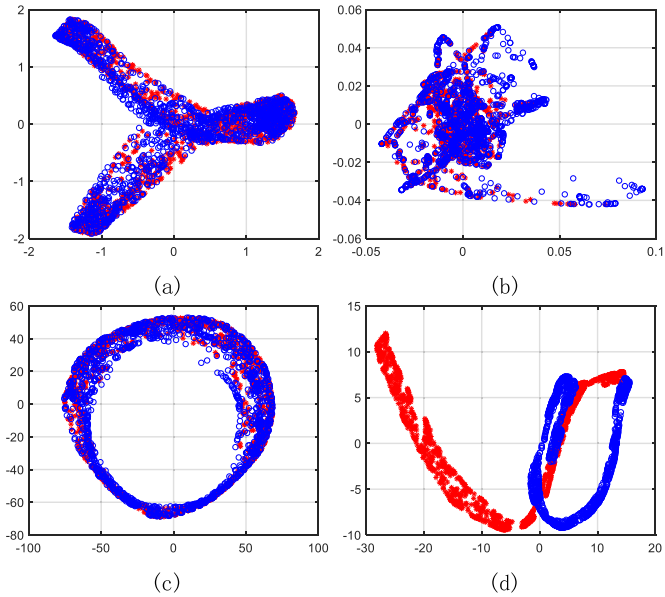


Fig. 2. Visualization of the 16-bit relaxed codes learned by four graph-based unsupervised hashing algorithms. The codes are projected into 2-dimensional space with PCA technique. (a) SH. (b) AGH. (c) IMH. (d) Ours.

B. Results on Synthetic Dataset

To illustrate the basic performance of the proposed DUGH, we firstly report the results on a synthetic dataset dubbed Two-SwissRolls. The visualization of this dataset in 3-dimensional space is shown in Fig. 1, where we can see that the two manifolds are partially overlapped. Each of the manifolds consists of 2,000 points. Following [62], we first generate the 3-dimensional points, and then embed them into $\mathbb{R}^{200}$ by adding 197-dimensional uniform noise. We randomly choose 1,000 examples as the test set, and the rest of examples form the training set. We perform the feature normalization on all examples to make each dimension have zero mean.

In order to illustrate the superiority of our method for learning discriminative codes, we perform the visualization of the 16-bit relaxed codes learned by four graph-based unsupervised hashing algorithms (i.e., SH, AGH, IMH, and DUGH) in Fig. 2, where the codes are projected into 2-dimensional space with PCA technique. As we can see, the learned binary codes of the proposed DUGH is able to better preserve the original neighboring structure when compared with the other three graph-based methods. This is mainly because DUGH pulls the similar points together and meanwhile pushes dissimilar points far away from each other, and these results demonstrate the effectiveness and discriminability of our methods.

Table I and Fig. 4 show other experimental results. As we can see, the proposed method outperforms other methods in most cases. Compared with other graph-based method including SH, AGH, and IMH, the proposed DUGH outperforms these methods with a large margin, which validates the advantage of DUGH to learn both accurate and discriminative hash codes.

TABLE II
THE COMPARISON BETWEEN DUGH AND SEVERAL TYPICAL UNSUPERVISED HASHING ALGORITHMS WITH THE HASH CODES OF DIFFERENT LENGTHS ON MNIST DATASET. THE BEST RESULT UNDER EACH SETTING IS SHOWN IN BOLDFACE

Method	mAP					F1 score					Precison@500				
	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits
DUGH	0.8179	**0.8661**	0.8383	**0.8714**	**0.8791**	**0.8318**	**0.8459**	**0.7975**	**0.7976**	**0.7678**	0.8436	**0.9136**	0.8522	**0.9075**	0.9141
$DUGH_0$	**0.8228**	0.8218	**0.8728**	0.8342	0.8705	0.7584	0.6971	0.5947	0.4185	0.1610	**0.8781**	0.8816	**0.9212**	0.9069	**0.9256**
LSH	0.2214	0.2415	0.2926	0.3431	0.3702	0.0457	0.0016	0.0003	0.0001	0.0000	0.3812	0.5221	0.5864	0.6074	0.7022
SH	0.2662	0.2603	0.2515	0.2399	0.2359	0.0765	0.0024	0.0002	0.0001	0.0000	0.5698	0.6178	0.6286	0.6223	0.6246
AGH	0.5436	0.4472	0.4033	0.3686	0.3059	0.4210	0.0823	0.0288	0.0173	0.0086	0.8331	0.8553	0.8605	0.8599	0.8383
IMH	0.6191	0.7068	0.6922	0.7069	0.7252	0.6105	0.5481	0.3051	0.2986	0.1301	0.6736	0.7919	0.8143	0.7988	0.8141
PCAH	0.2667	0.2413	0.2214	0.2095	0.1850	0.0681	0.0015	0.0001	0.0000	0.0000	0.5738	0.6045	0.5888	0.5780	0.5400
ITQ	0.3991	0.4393	0.4494	0.4540	0.4584	0.1740	0.0259	0.0077	0.0027	0.0001	0.6848	0.7429	0.7611	0.7700	0.7889
SpH	0.2637	0.3222	0.3388	0.3552	0.3823	0.0850	0.0060	0.0004	0.0001	0.0000	0.5101	0.6325	0.6793	0.7072	0.7578
AIBC	0.3502	0.3441	0.3468	0.3446	0.3534	0.2389	0.1266	0.0778	0.0591	0.0244	0.5738	0.5979	0.6078	0.6069	0.6097
SADH	0.4905	0.4101	0.3577	0.3283	0.2814	0.2558	0.0099	0.0004	0.0000	0.0000	0.7153	0.7454	0.7513	0.7351	0.7136

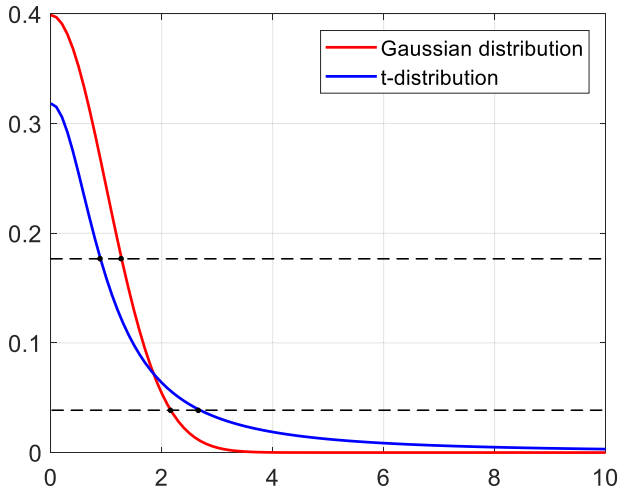


Fig. 3. Comparison of probability density functions between the Gaussian distribution and t-distribution. The red curve is the standard Gaussian distribution, and the blue one follows the Student t-distribution with one degree of freedom.

Furthermore, we can find that the F1 score of all these methods decreases dramatically when the code length increase, but DUGH still achieves relatively high results with long code lengths. All these results significantly demonstrate the effectiveness of our method in unsupervised hashing. Besides, by comparing the results of DUGH and $DUGH_0$, we can find that DUGH obtains much higher F1 score than $DUGH_0$, and performs a little bit better in terms of mAP and Precision@500. As for precision-recall curves shown in Fig. 4, the performance of these two methods is similar, and DUGH performs better than $DUGH_0$ with 16-bit binary codes. Therefore, the experimental results confirm the effectiveness of the quantization strategy.

C. Results on MNIST

We next test our method on a popular handwritten digit dataset, i.e., MNIST [63]. MNIST consists of 70,000 images regarding the digits from "0" to "9", and each of which is represented by a 784-dimensional vector. In our experiments, 1,000 queries are randomly selected as the test set, while the remaining 69,000 examples are regarded as the training set. The feature normalization is also performed on this dataset.

We report the comparative results with code length ranging from 16 to 128 bits in Table II and Fig. 5. We can see that DUGH outperforms other state-of-the-art methods under all the code lengths. The gain of DUGH is huge over IMH which is the most competitive one, and this result illustrates the remarkable ability of DUGH in unsupervised graph hashing as it takes the discrimination of data points into consideration. As shown in Fig. 5, DUGH consistently performs better than other methods in terms of the precision-recall curve. Furthermore, while F1 scores of all the other methods decrease sharply with the increase of code length, the results of DUGH are constant. In addition, it can be found that DUGH achieves better performance when compared with $DUGH_0$ in terms of F1 scores and precision-recall curves. This is mainly because the quantization strategy can significantly reduce the quantization error and improve the quanlity of the binary codes.

D. Results on CIFAR-10

In this section, we evaluate our method on CIFAR-10 [64], which is a subset of the well-known 80M tiny image collection. It consists of 60,000 images that are manually labeled with 10 classes (6000 examples per class). Each image in this dataset is represented by the GIST feature vector [65] of which the dimension is 384. We randomly sample 1,000 queries as the test set and use the rest as the training set. The feature normalization is also performed on this dataset.

The comparison experimental results on CIFAR-10 are reported in Table III and Fig. 6. Compared with the other methods, the proposed DUGH achieves better performances in most cases. This demonstrates the effectiveness of our methods in image hashing tasks. By comparing the mAP results and F1 scores among the four graph-based methods including SH, AGH, IMH, and DUGH, we can see that the proposed DUGH outperforms the other three methods. In addition, it can be seen that the F1 scores of all the other methods except DUGH is extremely low

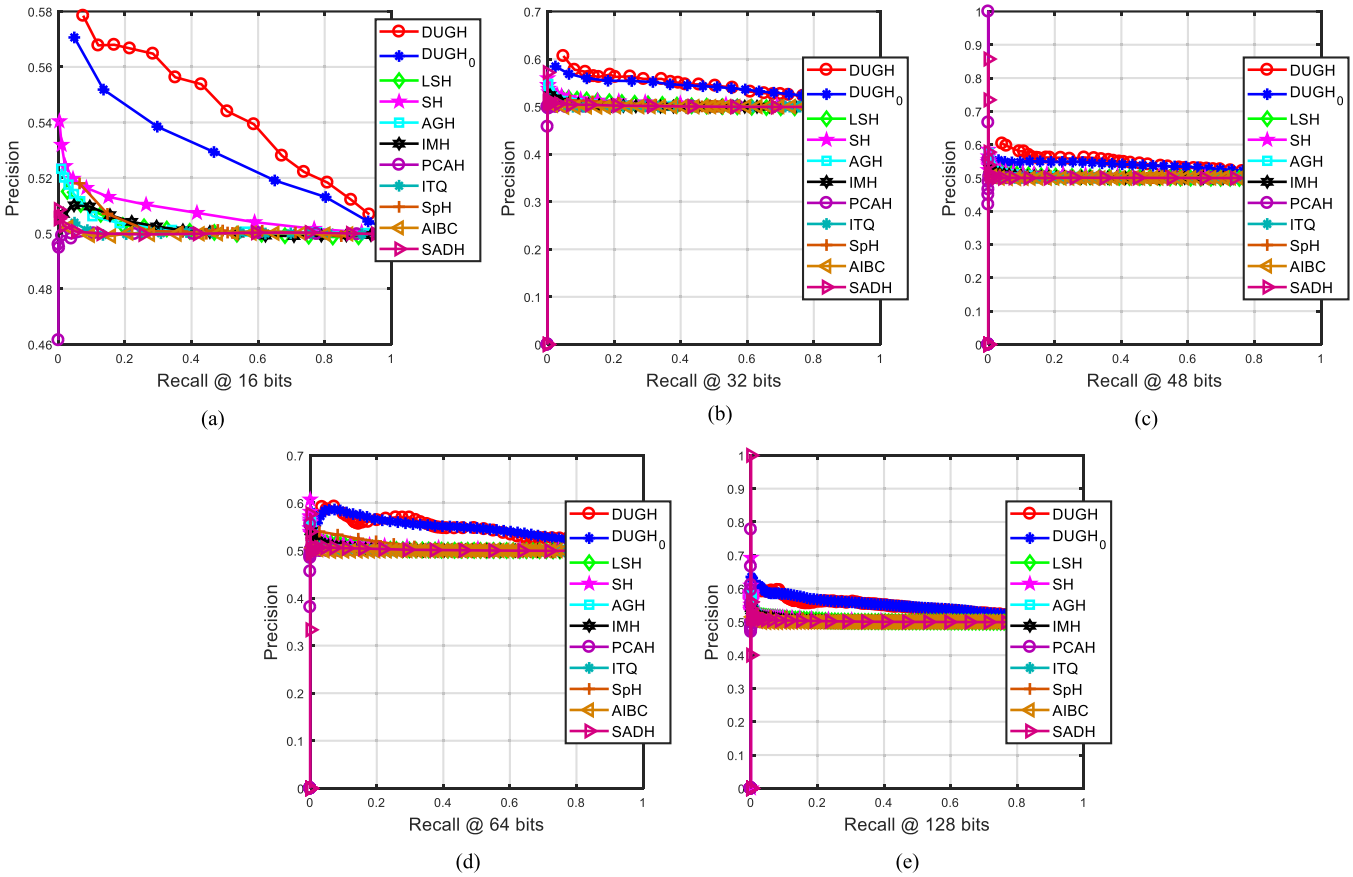


Fig. 4. Comparison of precision-recall curves with the hash codes of different lengths on Two-SwissRolls dataset.

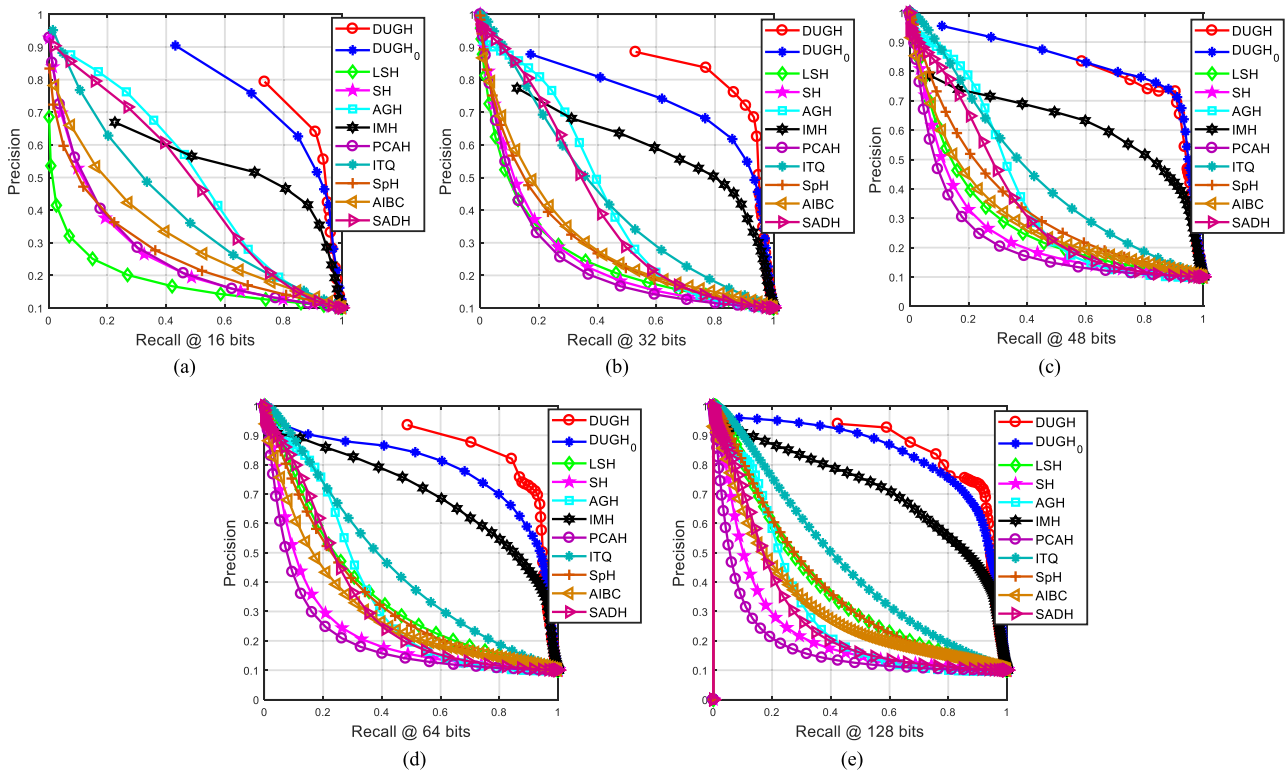


Fig. 5. Comparison of precision-recall curves with the hash codes of different lengths on MNIST dataset.

TABLE III
THE COMPARISON BETWEEN DUGH AND SEVERAL TYPICAL UNSUPERVISED HASHING ALGORITHMS WITH THE HASH CODES OF DIFFERENT LENGTHS ON CIFAR-10 DATASET. THE BEST RESULT UNDER EACH SETTING IS SHOWN IN BOLDFACE

Method	mAP					F1 score					Precison@500				
	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits
DUGH	**0.2169**	**0.2275**	0.2241	**0.2274**	0.2261	0.2530	**0.2013**	**0.1630**	**0.1472**	**0.1197**	0.2751	0.2863	0.2911	0.2957	0.2964
$DUGH_0$	0.2036	0.2235	**0.2307**	0.2248	**0.2361**	**0.2682**	0.2004	0.1137	0.0589	0.0099	0.2566	0.2792	**0.3038**	**0.2980**	**0.3205**
LSH	0.1333	0.1452	0.1532	0.1596	0.1615	0.0231	0.0002	0.0001	0.0000	0.0000	0.1773	0.2021	0.2144	0.2252	0.2544
SH	0.1375	0.1326	0.1301	0.1281	0.1267	0.0153	0.0001	0.0000	0.0000	0.0000	0.2095	0.2118	0.2091	0.2039	0.2003
AGH	0.1616	0.1544	0.1457	0.1413	0.1312	0.0733	0.0100	0.0047	0.0044	0.0033	**0.2855**	**0.2949**	0.2913	0.2888	0.2931
IMH	0.1791	0.1910	0.1933	0.1906	0.1927	0.2014	0.0894	0.0390	0.0344	0.0040	0.2314	0.2575	0.2572	0.2711	0.2729
PCAH	0.1364	0.1316	0.1269	0.1247	0.1182	0.0118	0.0001	0.0000	0.0000	0.0000	0.2062	0.2067	0.1972	0.1932	0.1763
ITQ	0.1656	0.1742	0.1776	0.1789	0.1856	0.0676	0.0055	0.0014	0.0006	0.0001	0.2344	0.2587	0.2733	0.2770	0.2884
SpH	0.1412	0.1495	0.1559	0.1586	0.1670	0.0223	0.0001	0.0000	0.0000	0.0000	0.2000	0.2192	0.2407	0.2536	0.2814
AIBC	0.1579	0.1641	0.1623	0.1617	0.1639	0.2096	0.1449	0.1327	0.1262	0.0721	0.1933	0.2067	0.2065	0.2066	0.2087
SADH	0.1619	0.1539	0.1527	0.1518	0.1496	0.0319	0.0003	0.0000	0.0000	0.0000	0.2407	0.2554	0.2672	0.2683	0.2782

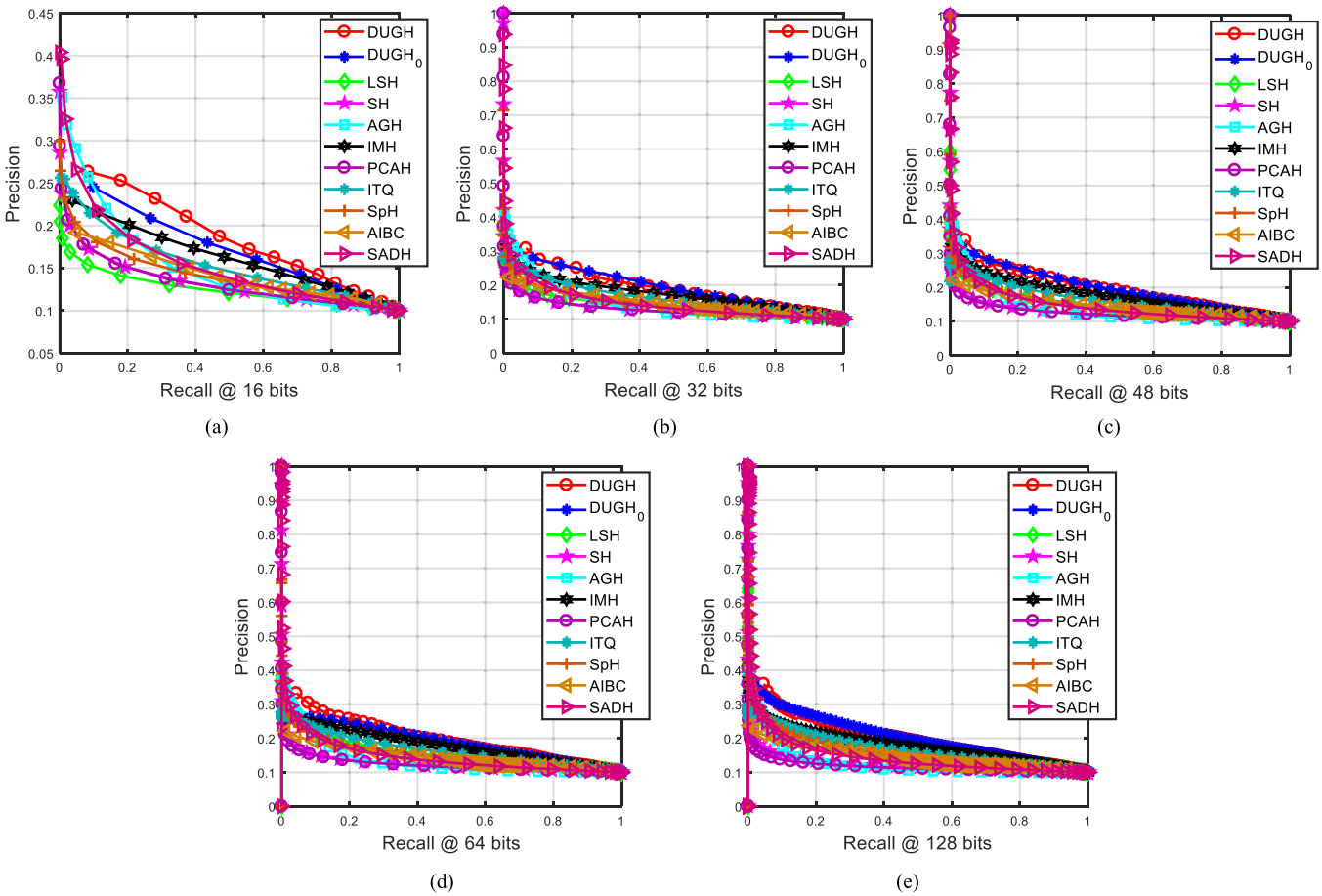


Fig. 6. Comparison of precision-recall curves with the hash codes of different lengths on CIFAR-10 dataset.

on this dataset when the code length is long, and this is because the precision and recall within Hamming radius 2 in these cases is almost zero in this case. Furthermore, the precision-recall curves of all these methods are shown in Fig. 6, and we can find that the results of DUGH and $DUGH_0$ are similar, and both of them perform better than other compared methods.

E. Results on Fashion-MNIST

In this section, we conduct the extensive experiments on Fashion-MNIST dataset [66], which consists of totally 70,000 images. On this dataset, each example is a 28×28 grayscale image belonging to one of 10 classes. In the experiments, we directly flatten each image into a 784-dimensional feature

TABLE IV
THE COMPARISON BETWEEN DUGH AND SEVERAL TYPICAL UNSUPERVISED HASHING ALGORITHMS WITH THE HASH CODES OF DIFFERENT LENGTHS ON FASHION-MNIST DATASET. THE BEST RESULT UNDER EACH SETTING IS SHOWN IN BOLDFACE

Method	mAP					F1 score					Precison@500				
	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits
DUGH	**0.5602**	0.5460	**0.5697**	**0.5932**	**0.5964**	**0.5890**	**0.6074**	**0.5930**	**0.6065**	**0.5431**	0.5886	0.5703	0.6115	0.6479	0.6549
$DUGH_0$	0.5551	**0.5568**	0.5642	0.5857	0.5857	0.5778	0.4896	0.4757	0.3528	0.1528	0.6172	0.6282	0.6574	0.6666	0.6979
LSH	0.2981	0.3369	0.3725	0.3929	0.4293	0.1614	0.0128	0.0012	0.0002	0.0000	0.4440	0.5289	0.5817	0.6068	0.6570
SH	0.3456	0.3091	0.2976	0.2861	0.2721	0.1917	0.0075	0.0007	0.0001	0.0000	0.5944	0.6216	0.6325	0.6342	0.6358
AGH	0.4203	0.3393	0.3105	0.2919	0.2525	0.3579	0.1058	0.0416	0.0243	0.0135	**0.6591**	**0.6887**	**0.6935**	**0.6910**	0.6902
IMH	0.4676	0.5359	0.5545	0.5566	0.5517	0.5048	0.4928	0.3750	0.3388	0.1523	0.5599	0.6147	0.6368	0.6488	0.6552
PCAH	0.3047	0.2642	0.2424	0.2291	0.2024	0.1158	0.0028	0.0001	0.0000	0.0000	0.5662	0.5990	0.6005	0.5981	0.5786
ITQ	0.4580	0.4777	0.4906	0.4901	0.4923	0.4377	0.1963	0.0868	0.0341	0.0019	0.6144	0.6608	0.6809	0.6851	**0.7003**
SpH	0.3513	0.3757	0.4170	0.4124	0.4345	0.1689	0.0068	0.0012	0.0001	0.0000	0.5804	0.6427	0.6654	0.6718	0.6996
AIBC	0.2875	0.3121	0.3095	0.3139	0.3149	0.3568	0.3667	0.3413	0.3107	0.2276	0.3751	0.4027	0.4078	0.4194	0.4222
SADH	0.3970	0.3602	0.3142	0.3012	0.2730	0.2418	0.0236	0.0016	0.0002	0.0000	0.6126	0.6499	0.6569	0.6697	0.6881

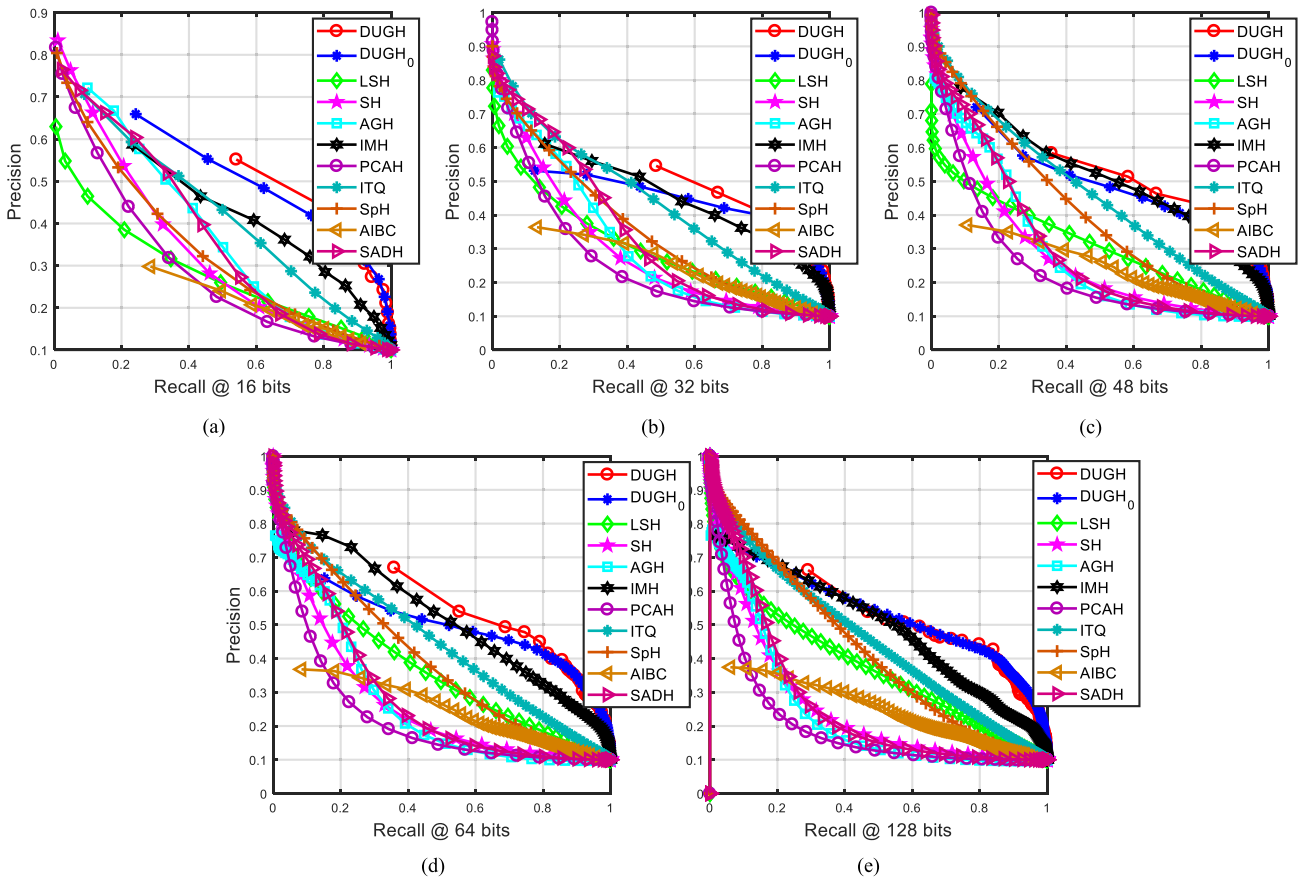


Fig. 7. Comparison of precision-recall curves with the hash codes of different lengths on Fashion-MNIST dataset.

vector. We select 1,000 queries as the test set, while the remaining 69,000 examples are regarded as the training set. The feature normalization is performed on this dataset.

Table IV shows the performance comparison of different hashing methods in terms of mAP, F1 score, and Precision@500. As we can see, the proposed DUGH obtains better mAP results and F1 scores in most cases. According to the quantitative results of Precision@500, AGH performs slightly better than our methods with 16-bit, 32-bit, 48-bit, and 64-bit codes, while ITQ performs best with 128-bit codes among all the other methods. However, our method still shows comparable results with these two methods. Moreover, as shown in Fig. 7, the proposed DUGH

TABLE V
THE COMPARISON BETWEEN DUGH AND SEVERAL TYPICAL UNSUPERVISED HASHING ALGORITHMS WITH THE HASH CODES OF DIFFERENT LENGTHS ON NUS-WIDE DATASET. THE BEST RESULT UNDER EACH SETTING IS SHOWN IN BOLDFACE

Method	mAP					F1 score					Precision@5000				
	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits	16bits	32bits	48bits	64bits	128bits
DUGH	0.3566	0.3583	0.3579	0.3582	0.3574	**0.2675**	**0.1912**	**0.1685**	**0.1572**	**0.1423**	0.3763	0.3804	0.3801	0.3811	0.3816
$DUGH_0$	**0.3634**	**0.3641**	0.3653	**0.3659**	**0.3663**	0.2528	0.1285	0.0930	0.0554	0.0178	**0.3941**	0.3912	0.3907	0.3964	0.3982
LSH	0.3363	0.3348	0.3452	0.3467	0.3471	0.0130	0.0001	0.0001	0.0000	0.0000	0.3645	0.3653	0.3866	0.3857	0.3901
SH	0.3446	0.3459	0.3441	0.3410	0.3383	0.0321	0.0002	0.0000	0.0000	0.0000	0.3932	0.3984	0.3951	0.3889	0.3831
AGH	0.3430	0.3420	0.3387	0.3380	0.3327	0.0423	0.0050	0.0028	0.0026	0.0022	0.3938	0.3983	0.4038	0.4074	0.4065
IMH	0.3591	0.3624	**0.3656**	0.3651	0.3659	0.1844	0.0950	0.0696	0.0434	0.0104	0.3830	0.3874	0.3968	0.4000	0.4001
PCAH	0.3489	0.3467	0.3435	0.3407	0.3345	0.0127	0.0000	0.0000	0.0000	0.0000	0.3900	0.3995	0.3927	0.3859	0.3687
ITQ	0.3591	0.3634	0.3641	0.3647	0.3655	0.0549	0.0088	0.0050	0.0035	0.0017	0.3935	0.4052	**0.4194**	**0.4186**	0.4181
SpH	0.3522	0.3544	0.3589	0.3603	0.3619	0.0227	0.0045	0.0006	0.0000	0.0000	0.3923	**0.4055**	0.4072	0.4129	**0.4196**
SADH	0.3454	0.3513	0.3476	0.3454	0.3458	0.0254	0.0004	0.0000	0.0000	0.0000	0.3937	0.4012	0.4157	0.4130	0.4119

outperforms all the other methods significantly. These results demonstrate the superiority of our method.

F. Results on NUS-WIDE Dataset

Lastly, we compare the proposed DUGH and $DUGH_0$ with the state-of-the-art methods on NUS-WIDE dataset [67]. The NUS-WIDE dataset contains 269,648 images collected from Flickr. Each image is represented by a 1134-dimensionnal low-level feature vector. This dataset contains 81 ground-truth concepts, and each image is tagged with multiple semantic labels. Following [23], we select the images associated with the 21 most frequent concepts and utilize a total of 195,834 images for evaluation. We randomly select 1,000 images to form the test set, and all the remaining images are used as the training set. The feature normalization is also performed on this dataset. Note that we do not compare our method with AIBC on this dataset, since it requires to compute the inner product of original features for all points and this is infeasible on the large-scale dataset.

The comparison results are shown in Table V and Fig. 8. As we can see, the proposed DUGH and $DUGH_0$ outperform other methods in terms of mAP, F1 score, and precision-recall curve in most cases. Specifically, the F1 score of DUGH is significantly higher than all the other methods, and the precision-recall curves with the hash codes of different code lengths also show that DUGH obtains better results than all the other algorithms. These results illustrate the effectiveness of our method. As for the precision@5000, the ITQ method obtains the best results with 48-bit and 64-bit hash codes among all mentioned methods, and the SpH method performs better than ours with 32-bit and 128-bit binary codes. However, our method is still comparable to ITQ and SpH in these cases, while it performs much better than these methods in terms of mAP, F1 score, and precision-recall curve.

G. Analyses and Discussions

1) Efficacy of Graph Construction Methods: As mentioned in Section III-A, we adopt an approximate way to efficiently construct the KNN graph. To illustrate the efficacy of our method, we study the quality of the established graph in this section. Firstly, we perform the ablation study to compare the performances of the exact KNN graph (denoted "Exact KNN") and the graph built by our method (denoted "Ours"). The experimental results are shown in Table VI and Table VII. We can find that the proposed method based on RP tree and neighbor exploring techniques can significantly reduce the computation time for graph construction on both CIFAR-10 and NUS-WIDE, which demonstrates the superiority of our method on large-scale hashing task over the methods based on exact KNN graph. Moreover, the mAP results of both two methods are extremely similar, which validates that our method is able to maintain the mAP obtained by the accurate KNN graph.

TABLE VI
THE COMPARISON BETWEEN TWO DIFFERENT GRAPH CONSTRUCTION METHODS ON CIFAR-10 DATASET IN TERMS OF CPU TIME AND MAP

Method	Time (s)	CIFAR-10				
		16bits	32bits	48bits	64bits	128bits
Exact KNN	146.724	0.2141	0.2155	0.2258	0.2285	0.2298
Ours	37.5753	0.2200	0.2235	0.2255	0.2270	0.2293

TABLE VII
THE COMPARISON BETWEEN TWO DIFFERENT GRAPH CONSTRUCTION METHODS ON NUS-WIDE DATASET IN TERMS OF CPU TIME AND MAP

Method	Time (s)	NUS-WIDE				
		16bits	32bits	48bits	64bits	128bits
Exact KNN	4528.12	0.3575	0.3593	0.3595	0.3592	0.3588
Ours	311.361	0.3566	0.3583	0.3579	0.3582	0.3574

Secondly, we perform the comparison results under different numbers of iterations for neighbor exploring on the five adopted datasets. As shown in Fig. 9, the accuracy of the approximate KNN graph improves from 98.91% to 100% with only one iteration for neighbor exploring on Two-SwissRolls, improves from 62.77% to 99.51% on MNIST, improves from 36.06% to 98.89% on CIFAR-10, improves from 77.66% to 99.81%

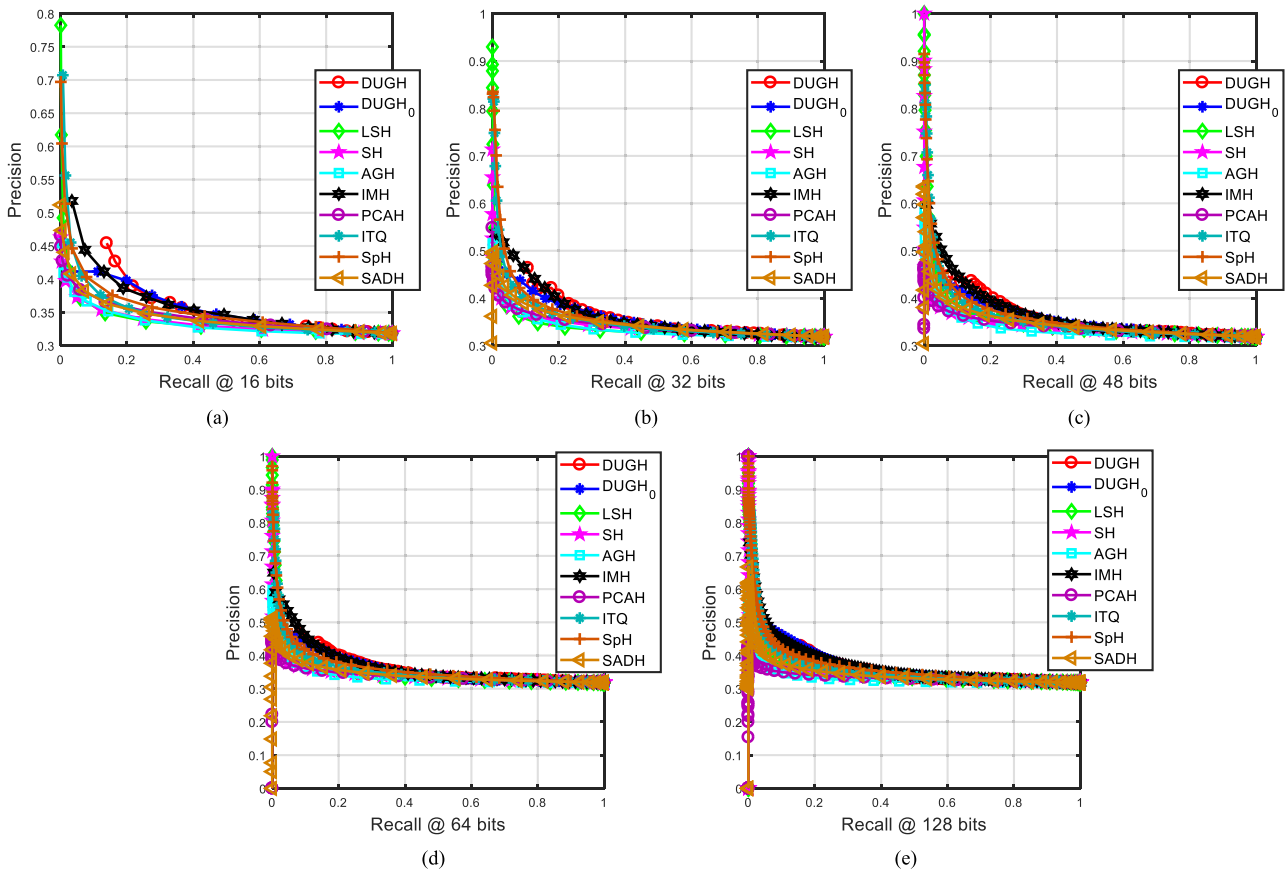


Fig. 8. Comparison of precision-recall curves with the hash codes of different lengths on NUS-WIDE dataset.

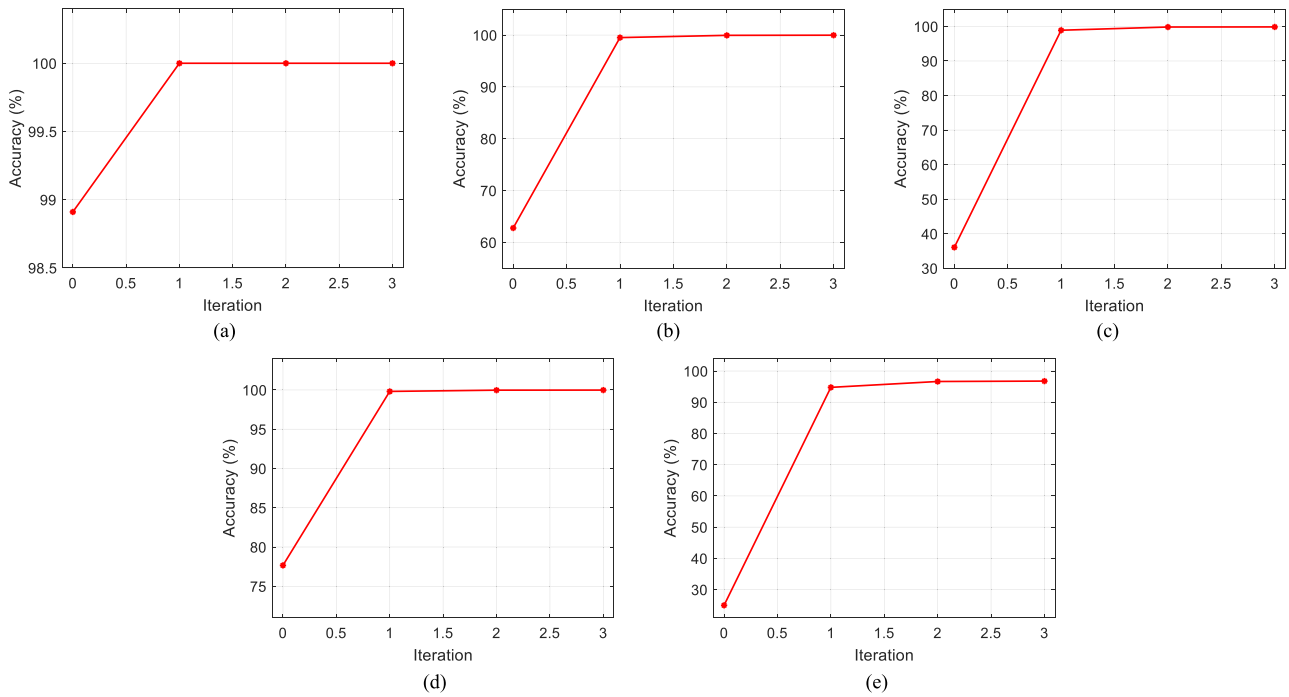


Fig. 9. Accuracy of the KNN graph under different numbers of iterations for neighbor exploring on all five datasets. (a) Two-SwissRolls. (b) MNIST. (c) CIFAR-10. (d) Fashion-MNIST. (e) NUS-WIDE.

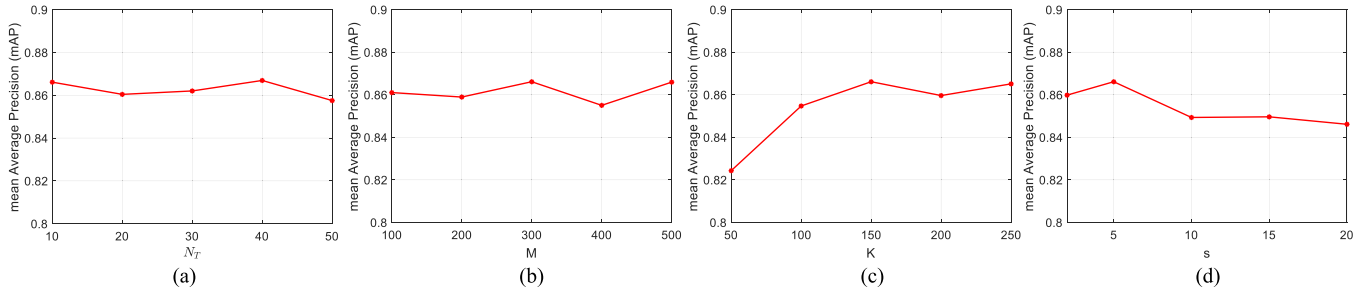


Fig. 10. Parameter sensitivity on MNIST. (a) N_T. (b) M. (c) K. (d) s.

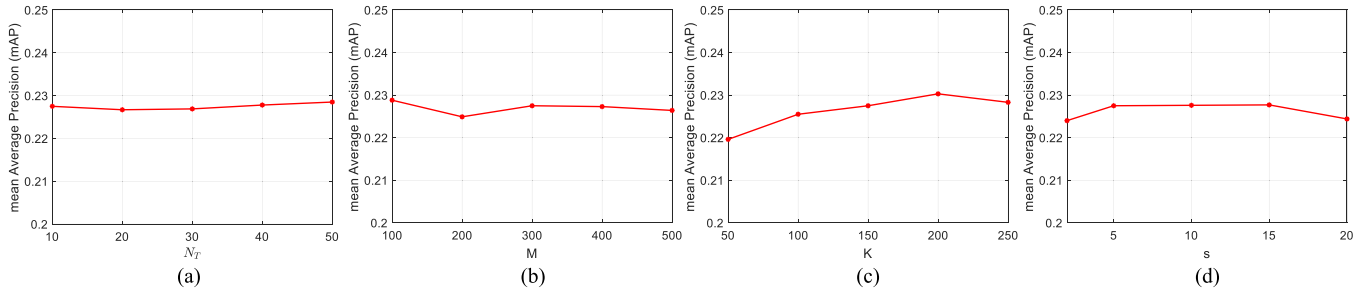


Fig. 11. Parameter sensitivity on CIFAR-10. (a) N_T. (b) M. (c) K. (d) is s.

on Fashion-MNIST, and improves from 24.96% to 94.76% on NUS-WIDE. Besides, it can be found that our method needs at most three iterations to achieve the almost accurate KNN graph on all five datasets. This result demonstrates the effectiveness of the neighbor exploring strategy for improving the accuracy of the approximate KNN graph.

2) Parameter Sensitivity: In this section, we study the influences of parameters on the performance of DUGH on MNIST and CIFAR-10 with 32-bit hash codes. The parameters include the number of RP trees N_T, the neighbor number of the KNN graph K, the cluster number of the training set M, and the number of selected points for out-of-sample coding s. The defaults of these parameters are set as $N_T = 10$, $K = 150$, $M = 300$, and $s = 5$. We then vary one of these parameters while fixing the others to the defaults. The experimental results are shown in Fig. 10 and Fig. 11. It reveals that the performance of DUGH is not sensitive to the variations of these parameters, so they can be easily tuned for practical use.

VI. Conclusion

In this paper, we propose a novel unsupervised hashing method named "Discriminative Unsupervised Graph Hashing" (DUGH), which aims at learning both accurate and discriminative binary codes for unsupervised hashing task. Different from existing methods that mainly focus on recovering the pairwise similarity of the original data in hash space, DUGH also takes the discrimination of data points into consideration and employs a probabilistic method to model both data similarity and dissimilarity in the original space. To efficiently and accurately calculate the neighbor graph, an effective graph construction algorithm is adopted in this paper, and it has been shown that our method can achieve the almost accurate KNN graph with limited number of iterations for neighbor exploring. The experimental results on five datasets reveal that the proposed method achieves very encouraging performance in terms of both hash lookup and hamming ranking when compared with the state-of-the-art unsupervised hashing methods.

In our future work, we will extend our method with deep learning technique, which is able to simultaneously learn binary codes and hash functions in an end-to-end way. Furthermore, we will also try to solve the optimization problem with discrete constraint of hashing problems.

References

[1] A. Gionis *et al.*, "Similarity search in high dimensions via hashing," *VLDB*, vol. 99, no. 6, 1999, pp. 518–529.
[2] 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.
[3] L. Gao *et al.*, "Learning in high-dimensional multimedia data: The state of the art," *Multimedia Syst.*, vol. 23, no. 3, pp. 303–313, 2017.
[4] J. Wang *et al.*, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.
[5] T. Zhou, C. Zhang, C. Gong, H. Bhaskar, and J. Yang, "Multiview latent space learning with feature redundancy minimization," *IEEE Trans. Cybern.*, 2018.
[6] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds., Red Hook, NY, USA: Curran Associates, Inc., 2009, pp. 1753–1760.
[7] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, IEEE, 2011, pp. 817–824.
[8] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, IEEE, 2012, pp. 2074–2081.
[9] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. on Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.

[10] J. Song, Y. Yang, X. Li, Z. Huang, and Y. Yang, "Robust hashing with local models for approximate similarity search," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1225–1236, Jul. 2014.

[11] J. Song, L. Gao, Y. Yan, D. Zhang, and N. Sebe, "Supervised hashing with pseudo labels for scalable multimedia retrieval," in *Proc. 23rd ACM Int. Conf. Multimedia*, ACM, 2015, pp. 827–830.

[12] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2015, pp. 37–45.

[13] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," *Preprints, arXiv:1511.03855*, 2015.

[14] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2018.

[15] X.-J. Mao, Y.-B. Yang, and N. Li, "Hashing with pairwise correlation learning and reconstruction," *IEEE Trans. Multimedia*, vol. 19, no. 2, pp. 382–392, Feb. 2017.

[16] F. Shen *et al.*, "Asymmetric binary coding for image search," *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2022–2032, Sep. 2017.

[17] F. Shen, X. Gao, L. Liu, Y. Yang, and H. T. Shen, "Deep asymmetric pairwise hashing," in *Proc. ACM Multimedia Conf.* ACM, 2017, pp. 1522–1530.

[18] Z. Chen, J. Lu, J. Feng, and J. Zhou, "Nonlinear discrete hashing," *IEEE Trans. Multimedia*, vol. 19, no. 1, pp. 123–135, Jan. 2017.

[19] X. Zhu *et al.*, "Graph PCA hashing for similarity search," *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2033–2044, Sep. 2017.

[20] J. Zhang and Y. Peng, "Query-adaptive image retrieval by deep weighted hashing," *IEEE Trans. Multimedia*, vol. 20, no. 9, pp. 2400–2414, Sep. 2018.

[21] J. Song *et al.*, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3210–3221, Jul. 2018.

[22] J. Song *et al.*, "A distance-computation-free search scheme for binary code databases," *IEEE Trans. Multimedia*, vol. 18, no. 3, pp. 484–495, Mar. 2016.

[23] F. Shen *et al.*, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, Dec. 2018.

[24] M. Hu, Y. Yang, F. Shen, N. Xie, and H. T. Shen, "Hashing with angular reconstructive embeddings," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 545–555, Feb. 2018.

[25] J. Song, L. Gao, L. Liu, X. Zhu, and N. Sebe, "Quantization-based hashing: A general framework for scalable image and video retrieval," *Pattern Recognit.*, vol. 75, pp. 175–187, 2018.

[26] Z. Chen, X. Yuan, J. Lu, Q. Tian, and J. Zhou, "Deep hashing via discrepancy minimization," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2018.

[27] K. Ghasedi Dizaji *et al.*, "Unsupervised deep generative adversarial hashing network," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, Jun. 2018.

[28] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geometry*, ACM, 2004, pp. 253–262.

[29] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2012.

[30] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. 28th Int. Conf. Mach. Learn. (ICML-11)*, 2011, pp. 1–8.

[31] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, and Z. Tang, "Inductive hashing on manifolds," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2013, pp. 1562–1569.

[32] F. Shen *et al.*, "A fast optimization method for general binary code learning." *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5610–5621, Dec. 2016.

[33] W. Liu, J. He, and S.-F. Chang, "Large graph construction for scalable semi-supervised learning," in *Proc. 27th Int. Conf. Mach. Learn. (ICML-10)*, 2010, pp. 679–686.

[34] M. Wang, W. Fu, S. Hao, D. Tao, and X. Wu, "Scalable semi-supervised learning by efficient anchor graph regularization," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 7, pp. 1864–1877, Jul. 2016.

[35] X. Li, C. Ma, J. Yang, and X. Huang, "Discrete locally-linear preserving hashing," in *Proc. 25th IEEE Int. Conf. Image Process. (ICIP)*, IEEE, 2018, pp. 490–494.

[36] S. Dasgupta and Y. Freund, "Random projection trees and low dimensional manifolds," in *Proc. 40th Annu. ACM Symp. Theory Comput.*, ACM, 2008, pp. 537–546.

[37] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proc. 20th Int. Conf. World Wide Web*, ACM, 2011, pp. 577–586.

[38] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 12, pp. 2143–2157, Dec. 2009.

[39] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. IEEE 12th Int. Conf. Comput. Vision*, IEEE, 2009, pp. 2130–2137.

[40] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1509–1517.

[41] L. Liu, M. Yu, and L. Shao, "Latent structure preserving hashing," *Int. J. Comput. Vision*, vol. 122, no. 3, pp. 439–457, 2017.

[42] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit., CVPR*, IEEE, 2012, pp. 2957–2964.

[43] P. Zhang, W. Zhang, W. J. Li, and M. Guo, "Supervised hashing with latent factor models," in *Proc. 37th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2014, pp. 173–182.

[44] J. Song *et al.*, "Optimized graph learning using partial tags and multiple features for image and video annotation," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 4999–5011, Nov. 2016.

[45] X. Wang, L. Gao, P. Wang, X. Sun, and X. Liu, "Two-stream 3-D convNet fusion for action recognition in videos with arbitrary size and length," *IEEE Trans. Multimedia*, vol. 20, no. 3, pp. 634–644, Mar. 2018.

[46] C. Gong, H. Shi, J. Yang, and J. Yanga, "Multi-manifold positive and unlabeled learning for visual analysis," *IEEE Trans. Circuits Syst. Video Technol.*, 2019.

[47] C. Gong, D. Tao, W. Liu, L. Liu, and J. Yang, "Label propagation via teaching-to-learn and learning-to-teach," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 6, pp. 1452–1465, Jun. 2016.

[48] C. Gong, D. Tao, X. Chang, and J. Yang, "Ensemble teaching for hybrid label propagation," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 388–402, Feb. 2017.

[49] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[50] 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.

[51] C. Silpa-Anan and R. Hartley, "Optimised kd-trees for fast image descriptor matching," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.,* IEEE, 2008, pp. 1–8.

[52] J. Tang, J. Liu, M. Zhang, and Q. Mei, "Visualizing large-scale and high-dimensional data," in *Proc. 25th Int. Conf. World Wide Web*, International World Wide Web Conferences Steering Committee, 2016, pp. 287–297.

[53] L. V. D. Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[54] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[55] J. Tang *et al.*, "Line: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web*, International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[56] B. Recht, C. Re, S. Wright, and F. Niu, "Hogwild: A lock-free approach to parallelizing stochastic gradient descent," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 693–701.

[57] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.

[58] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI*, vol. 1, 2014, p. 2.

[59] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 1183–1192.

[60] C. Huang, C. C. Loy, and X. Tang, "Unsupervised learning of discriminative attributes and visual representations," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 5175–5184.

[61] H. Schütze, "Introduction to information retrieval," in *Proc. Int. Commun. Assoc. Comput. Machinery Conf.*, 2008.

[62] G. Irie, Z. Li, X. M. Wu, and S. F. Chang, "Locally linear hashing for extracting non-linear manifolds," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2014, pp. 2123–2130.

[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.

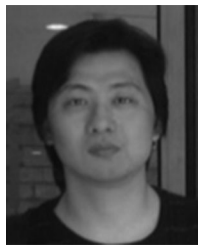
[64] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., Univ. Toronto, 2009.
[65] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vision*, vol. 42, no. 3, pp. 145–175, 2001.
[66] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *Preprints, arXiv:1708.07747*.
[67] T.-S. Chua *et al.*, "NUS-WIDE: A real-world web image database from National University of Singapore," in *Proc. ACM Int. Conf. Image Video Retrieval*, ACM, 2009, p. 48.

Chao Ma received the B.E. degree in automation science and technology from Xi'an Jiaotong University, Xi'an, China, in 2015. He is currently working toward the Ph.D. degree with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, under the supervision of Prof. Jie Yang. His research areas mainly include multimedia hashing and machine learning.

Chen Gong received the B.E. degree from East China University of Science and Technology (E- CUST), Shanghai, China, in 2010, and dual doctoral degree from Shanghai Jiao Tong University (SJTU), Shanghai, China and University of Technology Sydney (UTS), NSW, Australia, in 2016 and 2017, respectively. Currently, he is a Full Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests mainly include machine learning, data mining, and learning-based vision problems. He has authored or coauthored more than 50 technical papers at prominent journals and conferences such as IEEE T-NNLS, IEEE T-IP, IEEE T-CYB, IEEE T-CSVT, IEEE T-MM, IEEE T-ITS, CVPR, AAAI, IJCAI, ICDM, etc. He also serves as the reviewer for more than 20 international journals such as AIJ, IEEE T-NNLS, IEEE T-IP, and also the PC member of several top-tier conferences such as ICML, AAAI, IJCAI, ICDM, AISTATS, etc. He received the Excellent Doctorial Dissertation awarded by Shanghai Jiao Tong University (SJTU) and Chinese Association for Artificial Intelligence (CAAI). He was also enrolled by the "Summit of the Six Top Talents" Program of Jiangsu Province, China, and the "Young Elite Scientists Sponsorship Program" of China Association for Science and Technology.

Xiang Li received the B.E. degree in automation science and technology from the Xi'an Jiaotong University, Xi'an, China, in 2017. She is currently working toward the M.E. degree with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China, under the supervision of Prof. Jie Yang. Her research interests mainly include machine learning and computer vision with respect to deep learning and image hashing.

Xiaolin Huang (S'10–M'12–SM'18) received the B.S. degree in control science and engineering, and the B.S. degree in applied mathematics from Xi'an Jiaotong University, Xi'an, China, in 2006. In 2012, he received the Ph.D. degree in control science and engineering from Tsinghua University, Beijing, China. From 2012 to 2015, he worked as a Postdoctoral researcher with ESAT-STADIUS, KU Leuven, Leuven, Belgium. After that he was selected as an Alexander von Humboldt Fellow and is working in Pattern Recognition Lab, the Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany. From 2016, he has been an Associate Professor with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China. In 2017, he was awarded by "1000-Talent Plan" (Young Program). His current research interests include machine learning and optimization.



Wei Liu received the M.Phil. and Ph.D. degrees in electrical engineering from Columbia University, New York, NY, USA, in 2012. He is currently a Research Scientist with Tencent AI Lab, and holds an adjunct faculty position with Rensselaer Polytechnic Institute, Troy, NY, USA. In 2012, he was the Josef Raviv Memorial Postdoctoral Fellow with the IBM T. J. Watson Research Center, for one year. His current research interests include machine learning, data mining, computer vision, pattern recognition, and information retrieval.

Jie Yang received the Ph.D. degree from the Department of Computer Science, Hamburg University, Germany, in 1994. Currently, he is a Professor with the Institute of Image Processing and Pattern Recognition, Shanghai Jiao Tong University, Shanghai, China. He has led many research projects (e.g., National Science Foundation, 863 National High Tech. Plan), had one book published in Germany, and authored more than 200 journal papers. His major research interests include object detection and recognition, data fusion and data mining, and medical image processing.