Learning to Hash With Optimized Anchor Embedding for Scalable Retrieval

Yuchen Guo, Guiguang Ding, Li Liu, Jungong Han, and Ling Shao, *Senior Member, IEEE*

***Abstract*— Sparse representation and image hashing are powerful tools for data representation and image retrieval respectively. The combinations of these two tools for scalable image retrieval, i.e., sparse hashing (SH) methods, have been proposed in recent years and the preliminary results are promising. The core of those methods is a scheme that can efficiently embed the (high-dimensional) image features into a low-dimensional Hamming space, while preserving the similarity between features. Existing SH methods mostly focus on finding better sparse representations of images in the hash space. We argue that the anchor set utilized in sparse representation is also crucial, which was unfortunately underestimated by the prior art. To this end, we propose a novel SH method that optimizes the integration of the anchors, such that the features can be better embedded and binarized, termed as Sparse Hashing with Optimized Anchor Embedding. The central idea is to push the anchors far from the axis while preserving their relative positions so as to generate similar hashcodes for neighboring features. We formulate this idea as an orthogonality constrained maximization problem and an efficient and novel optimization framework is systematically exploited. Extensive experiments on five benchmark image data sets demonstrate that our method outperforms several state-of-the-art related methods.**

***Index Terms*— Sparse representation, hashing, retrieval, scalability, orthogonality, optimization.**

I. Introduction

Approximate Nearest Neighbor (ANN) search has become a fundamental paradigm in various applications, AQ:1 such as image recognition and image retrieval [1], [2]. Its aim is to find some approximate nearest neighbors for a query from a collection of data. To cope with large-scale data, many techniques for fast ANN search have been proposed in the past. One popular pathway is based on trees, e.g. kd-tree [3], which has logarithmic retrieval complexity for low-dimensional data. However, most tree-based methods may reduce to exhaustive linear scanning for high-dimensional data because of the curse of dimensionality. Another pathway, called hashing [4], represents data by a sequence of binary codes. Benefiting from the binary representation, the storage can be dramatically reduced and the search can be quite efficient, even with a large-scale dataset [5]–[10]. With proper designs, hashing will not necessarily degrade the search accuracy. In view of the above advantages, hashing methods have drawn increasing attention recently from the industry and academia.

The key problem in hashing is how to embed the original features, which are usually *high-dimensional floating-point number* representations, into the *low-dimensional binary* Hamming space while the similarity between the original features can be preserved. Locality Sensitive Hashing (LSH) [11], as the most notable and fundamental hashing method, adopts *random* projections to generate hashcodes. Theoretically, the Hamming distance between those hashcodes can progressively approximate the Euclidean distance between the original features. But in practice, very long hashcodes (say, $1,024$ bits) are required in this approach so as to achieve satisfactory performance. To address this issue, several *learning* based methods have been proposed, such as PCA Hashing [12], Spectral Hashing [13], and Iterative Quantization [14]. Though better performance can be obtained, compared to LSH, these methods still suffer from two shortcomings due to the *linear* projections employed by them: 1) they may fail to preserve the *non-linear* manifold structure of data; and 2) they may achieve high precision but *low recall* as the feature space is segmented so finely that data may be scatted in the Hamming space, which leads to extremely low collision probability [15].

Alternatively, methods exploiting *non-linear* projections [6], [16], [17] have gained increasing popularity due to their superior performance. Specifically, these methods, thanks to the non-linear projections, can better preserve the complicated geometric structure of data, especially the manifold structure. One representative framework is called Sparse Hashing (SH) [6], [16]–[20] since it is based on the Sparse Coding (SC) that was successfully used in image representation [21], [22], classification [23], and denoising [24]. Basically, the algorithm is carried out by two forms of transformation. First, a non-linear transformation converts the original features to the sparse representations. Second, a linear transformation further transfers the sparse representations generated in the previous step to the Hamming space. Generally, non-linear SH methods are capable of overcoming two shortcomings of the linear methods if a proper learning strategy is deployed. However, these two problems, i.e., how to generate effective sparse representations for hashing and how to transform the sparse

AQ:2 Manuscript received July 30, 2016; revised December 2, 2016; accepted December 28, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61571269 and Grant 61271394, in part by the National Basic Research Project of China under Grant 2015CB352300, and in part by the Royal Society Newton Mobility under Grant IE150997. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Lei Zhang. AQ:3 *(Corresponding authors: Guiguang Ding; Jungong Han.)*

AQ:4 Y. Guo and G. Ding are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: yuchen.w.guo@gmail.com; dinggg@tsinghua.edu.cn).

L. Liu and J. Han are with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: li2.liu@northumbria.ac.uk; jungong.han@northumbria.ac.uk).

L. Shao is with the School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, U.K. (e-mail: ling.shao@ieee.org).

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/TIP.2017.2652730

representation into the Hamming space with data similarity preserved, still need to be solved.

In this paper, we propose a novel SH method, aiming at preserving the non-linear manifold structure of the original features in the Hamming space. In particular, motivated by Locally Linear Embedding (LLE) [25] and Anchor Graph [17], we learn a *non-linear locality-preserving* dimension reduction function via the sparse representation of data. This non-linear function secures similar low-dimensional representations for neighboring points. After such an effective dimension reduction, we can easily generate binary hashcodes from the embedded low-dimensional features. When learning this function, previous works [6], [16]–[18], [20] only looked into the sparse representation of data but ignored the importance of the anchors [17] utilized in constructing the sparse representation. We notice that the low-dimensional embedding of the anchors has a significant impact on the hash function. Specifically, it is discovered that pushing anchors far from axis while preserving the geometric structure of them during the anchor embedding usually leads to high-quality hashcodes. We investigate this phenomenon and mathematically formulate the implementation of this idea to an orthogonality constrained maximization problem which optimizes the anchor embedding with the aim to avoid generating two different hashcodes for neighboring low-dimensional points. With such an optimization, the locality of original features can be well preserved and better ANN search performance can be achieved. Moreover, we put forward an efficient learning algorithm to solve the complicated orthogonality constrained optimization problem.

The rest of this paper is organized as follows. In Section II, we briefly describe some preliminaries and review the related hashing works. The proposed SHODE is introduced detailedly in Section III. The experimental results and discussion are given in Section IV, and we draw conclusions in Section V.

II. Preliminaries and Related Work

A. Formulation

Given a set of d-dimensional features $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, we can design a hash function $h(\cdot)$ to generate k-bit *binary* representations, i.e., hashcodes, for them as $\mathbf{b}_i = h(\mathbf{x}_i) \in \{-1, 1\}^k$,[1] such that the similarity between features can be preserved, i.e., similar features have similar hashcodes. This idea can be formulated as the following learning problem,

$$\min_h \sum_{i,j} s_{ij} d_H(h(\mathbf{x}_i), h(\mathbf{x}_j)), \quad \text{s.t. } \mathcal{C}(h), \tag{1}$$

where d_H is the Hamming distance between hashcodes, s_{ij} is the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathcal{C}(h)$ is the constraints applied to h, for example, we always expect the hashcodes to be balanced ($\sum_i \mathbf{b}_i = \mathbf{0}_k$) and uncorrelated ($\mathbf{B}\mathbf{B}^T = n\mathbf{I}_k$).

Since it is difficult, if not impossible, to design an effective hash function by directly converting $\mathbf{X}$ to hashcodes, a two-step strategy is widely adopted [12]–[14], [16]. In the first step, the original features $\mathbf{X}$ are projected into a k-dimensional space as $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_n] \in \mathbb{R}^{k\times n}$ by a projection function $\phi(\cdot)$. Because we usually have $k < d$, this step can be regarded as a dimension reduction step. Then, the low-dimensional embedded representations $\mathbf{Y}$ are quantified into binary codes by, in most cases, the sign function as $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n] = \text{sign}(\mathbf{Y})$, where $\text{sign}(x) = 1$ if $x > 0$ or -1 otherwise. By doing so, the overall hash function becomes $h(\cdot) = \text{sign}(\phi(\cdot))$. In this way, learning h can be achieved by learning ϕ instead. However, the sign function still makes the learning intractable in many cases [13]. A common solution is to remove the sign function and to further *relax* the learning problem as a real-valued problem,

$$\min_\phi \sum_{i,j} s_{ij} d(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)), \quad \text{s.t. } \mathcal{C}(\phi). \tag{2}$$

[1]In implementation, we can use $\{0, 1\}$. In fact, these two representations are equivalent. So we use $\{-1, 1\}$ in this paper for convenience as in [17].

TABLE I
Notations and Descriptions in This Paper

Notation	Description	Notation	Description
$\mathbf{X}$	data matrix	n	#samples
$\mathbf{Y}$	projected matrix	d	#features
$\mathbf{B}$	binary Hashcode	k	#Hashcode
$\mathbf{P}$	projection matrix	m	#anchors
$\mathbf{D}$	anchor set	p	#NN
$\mathbf{A}$	sparse matrix	t	#iterations
$\mathbf{S}$	similarity matrix	h, ϕ, ρ	functions
$\mathbf{R}$	rotation matrix	τ	step size

AQ:5

B. Linear Hashing

Several methods [13], [16], [26]–[29] assume a linear projection for ϕ, i.e., $\phi(\mathbf{x}) = \mathbf{P}\mathbf{x}$, where $\mathbf{P} \in \mathbb{R}^{k\times d}$ is a linear projection matrix. After proper algebra operations and transformations, the learning problem can be rewritten into a simple formulation as follows:

$$\max_{\mathbf{P}} \; tr(\mathbf{P}\mathbf{X}\mathbf{S}\mathbf{X}^T\mathbf{P}^T), \quad \text{s.t. } \mathbf{P}\mathbf{P}^T = \mathbf{I}_k, \tag{3}$$

where $tr(\cdot)$ is the trace function, $\mathbf{S} = [s_{ij}]$ is the similarity matrix among training samples, and the orthogonal constraint requires the selected directions to be uncorrelated. $\mathbf{S}$ determines what kind of information is preserved depending on the intentions of different methods. The statistics reveal that the majority of existing works choose to preserve the local manifold structure of data [13], [30]. After the above assumption and operations, the problem defined in Eq. (3) can be easily solved. However, since only linear projections are used, these methods may still fail to preserve the similarity.

C. Sparse Hashing

To preserve the non-linear manifold structure, Sparse Hashing [6], [16]–[18], [20], which learns a non-linear ϕ, has attracted considerable attention. Given a set of anchors $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_m] \in \mathbb{R}^{d\times m}$, a sparse presentation $\mathbf{A} = [\mathbf{a}_1, ..., \mathbf{a}_n] \in \mathbb{R}^{m\times n}$ is constructed by $\mathbf{A} = \rho(\mathbf{X}, \mathbf{D})$. This can be done by conventional sparse reconstruction [31] as

$$\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \mathcal{R}(\mathbf{A}), \quad \text{s.t. } \mathcal{C}(\mathbf{A}), \tag{4}$$

where $\mathcal{R}(\mathbf{A})$ denotes regularization terms, such as ℓ_1-norm regularization for sparsity, and other terms like Graph regularization [32], and $\mathcal{C}(\mathbf{A})$ is a constraint on $\mathbf{A}$. Obviously, this method is non-linear. in [19] and [33], such schemes are employed, and the sparse codes are then encoded into a set of integers which are composed of the nonzero indices. This index set sacrifices the advantages of efficient storage and speedy binary code matching. Alternatively, in [20], Zhu *et al.* proposed an encoding method in which the binary codes are generated by setting nonzero elements in $\mathbf{A}$ as 1 and the others as 0. The problem of this method lies in its incapability of generating compact and balanced representations because of the sparsity of $\mathbf{A}$, thereby degrading the quality of hashcodes. In addition, Ye and Li [34] proposed the Compact Structure Hashing that combines the linear projection learning in Eq. (3) and sparse reconstruction in Eq. (4) in a unified objective function to simultaneously exploits the non-linear structure of data and finds the optimal projection function. However, this method intrinsically adopts a linear projection to the Hamming space such that it still suffers from the low-recall problem.

A possible way of solving this problem is the usage of the Anchor Graph [17], in which each anchor is either randomly sampled from the data or the cluster centroids after applying a data clustering algorithm, such as Kmeans. The sparse representation can be build in the Anchor Graph as follows:

$$a_{ji} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{d}_j\|^2/\sigma^2)}{\sum_{j' \in \mathcal{N}(\mathbf{x}_i)} \exp(-\|\mathbf{x}_i - \mathbf{d}_{j'}\|^2/\sigma^2)}, & \forall j \in \mathcal{N}(\mathbf{x}_i) \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

where $\mathcal{N}(\mathbf{x}_i)$ is the p-NN of $\mathbf{x}_i$ in $\mathbf{D}$ and σ is the bandwidth parameter. The obtained sparse representation is claimed to preserve the similarity between data. Obviously, $\mathbf{a}_i$ has at most p nonzero elements, implying that $\mathbf{a}$ is sparse. Finally, $\phi(\cdot)$ is constructed by projecting the sparse representation to a low-dimensional space, i.e., $\phi(\mathbf{x}) = \mathbf{P}\rho(\mathbf{x}, \mathbf{D})$. To preserve the similarity, Liu *et al.* [17] proposed the Anchor Graph Hashing that constructs $\mathbf{P}$ by solving an eigenvalue problem on the Anchor Graph. Lin *et al.* [16] proposed the Compressed Hashing in which the sampled p_{ij} from $\mathcal{N}(0, 1/k)$ can construct a projection satisfying Restricted Isometry Property [35] in Compressed Sensing theory [36]. Similarly, Shen *et al.* [6] proposed an inductive method to construct $\mathbf{P}$. Zhu *et al.* [37] proposed a sparse embedding and least variance encoding approach to hashing, which constructs $\mathbf{P}$ by solving a reconstruction problem and adjusts the projected representation to minimize the variance for preserving similarity. Even though promising results have been obtained, how to design effective ρ and $\mathbf{P}$ is still an open issue, which is the focus of this paper.

Moreover, it is noticed that in recent years many works have attempted to combine the deep convolutional neural network [38] with hashing, i.e., deep hashing [39]–[43]. For example, Liong *et al.* [39] proposed a deep hashing method in which the output of the networks is required to preserve the supervised similarity. Lai *et al.* [40] proposed a *piece-wise* function for the network to address the discrete optimization problem in deep hashing. Zhang *et al.* [41] presented a network using similarity regularized triplet loss for person re-identification. However, it should be pointed out that these deep hashing approaches should be categorized into the *supervised hashing* methods in which supervised knowledge (*e.g.*, label information) is required for model training. As is known to all, collecting sufficient supervised knowledge is expensive in many applications [44]. On the contrary, this paper, and many SH methods focus on the *unsupervised hashing* which only exploits the intrinsic unsupervised information of data and thus they are free from the lack of the supervised knowledge.

III. The Proposed Method

Our method follows the framework of SH. Firstly, we construct a sparse representation for the original features in a non-linear manner. Secondly, we linearly project the sparse representation into the low-dimensional space. Thirdly, we obtain hashcodes from low-dimensional embedding using the sign function. The special properties of our projection are 1) the low-dimensional embedding preserves the local manifold structure of original data, and 2) the similarity structure is preserved as well after the sign quantization. The following two subsections will elaborate on them one by one. Since all involved steps take data similarity preservation into account, the obtained hashcodes, without saying, will naturally preserve the similarity relationship of original features, thus resulting in superior ANN search and image retrieval performance.

A. Locality-Preserving Dimension Reduction

In this subsection, we will provide an effective method for non-linear dimension reduction based on Sparse Coding which can well preserve the non-linear local manifold structure.

Locality-preserving dimension reduction aims to find low-dimensional embedding which can preserve the neighborhood structure or manifold structure of the original data. One representative and seminal work is Locally Linear Embedding (LLE) [25] which can find a linear embedding for non-linear manifold. However, LLE does not provide an explicit dimension reduction function for the out-of-sample data (data which is not in the training set). Another celebrated method is called Locality Preserving Projections (LPP) [30] which learns an explicit linear projection function instead. Despite its ability of easily addressing the out-of-sample data, the linear function adopted by LPP may perform worse than the non-linear ones.

Although LLE does not provide the projection function for out-of-sample data, it still reveals an important property of the non-linear manifold: local linearity. That is, the manifold structure is locally linear even though it is non-linear globally. Such a property is also utilized in [45] and [46], which can be further interpreted below. Given some points $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_m]$ and their corresponding low-dimensional embeddings $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_m]$ obtained by non-linear methods like LLE, the low-dimensional embedding $\mathbf{y}$ for a new data point $\mathbf{x}$ is given by

$$\mathbf{y} \leftarrow \sum_{i \in \mathcal{N}(\mathbf{x})} a_i \mathbf{y}_i, \tag{6}$$

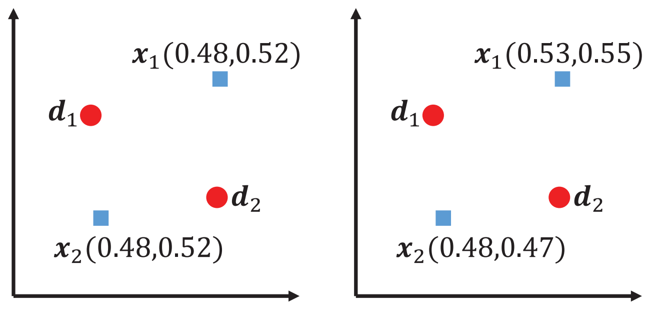


Fig. 1. Sparse representation by different methods. (a) by Eq. (5). (b) by Eq. (7).

where $\mathcal{N}(\mathbf{x})$ is the p-NN of $\mathbf{x}$ in $\mathbf{D}$, and a_i is the corresponding weight. One straightforward way to compute a_i is based on Eq. (5). But it should be noticed that such a formulation only defines the weight and does not reflect the *relative position* between $\mathbf{x}$ and $\mathcal{N}(\mathbf{x})$. Therefore, the embedding $\mathbf{y}$ relying on the weight may lose important information. Therefore, to make use of the local linearity better, in this paper, we propose to generate $\mathbf{a}$ by a sparse reconstruction procedure as follows:

$$\min_{\mathbf{a}} \ \|\mathbf{x} - \mathbf{D}\mathbf{a}\|_F^2, \quad \text{s.t. } a_i \geq 0, \ a_j = 0 \text{ if } j \notin \mathcal{N}(\mathbf{x}). \tag{7}$$

Here, we require $\mathbf{a}$ to be nonnegative so that it can serve as "weight". Moreover, only $\mathcal{N}(\mathbf{x})$ is used to reconstruct $\mathbf{x}$ for preserving the locality. Obviously, the solution $\mathbf{a}$ is sparse in the sense that it has at most p nonzero elements ($p \ll m$).

By combining Eq. (6) and Eq. (7), the overall dimension reduction can be summarized as follows: 1) An anchor set $\mathbf{D}$ is generated from training data by K-means clustering; 2) We find the locality preserving embedding $\mathbf{Y}$ for it by a non-linear method, called Laplacian Eigenmap [47]. As this step is only conducted for the anchor set, there is no need to learn a projection function for the out-of-sample data; 3) For a new data point $\mathbf{x}$, the sparse representation $\mathbf{a}$ is obtained by solving Eq. (7); 4) The low-dimensional embedding $\mathbf{y}$ is obtained by Eq. (6). As a result, the projection function $\mathbf{P}$ in our method can be considered as the low-dimensional embedding $\mathbf{Y}$ of the anchor set. Due to the non-linearity in Eq. (7), the entire procedure is non-linear as in LLE. Meanwhile, it also has an explicit projection function (Eq. (6) and (7)) for out-of-sample data. Hence, it can be concluded that our method combines the advantages of LLE and LPP but gets rid of their shortcomings.

Seen from Eq. (6) and Eq. (7), two points that are close in the original feature space will also have similar low-dimensional representations after the projection, because they will choose similar p-NN anchor sets from $\mathbf{D}$. In other words, these two points will finally lie very close to the embeddings of their corresponding anchor sets, which are also similar.

Here, we discuss the difference between our sparse representation constructed by Eq. (7) and the widely used version expressed in Eq. (5). In principle, representations based on Eq. (5) fail to consider the relative position of $\mathbf{x}$ and $\mathcal{N}(\mathbf{x})$ while using Eq. (7) can achieve this goal. An intuitive illustration is shown in Figure 1, in which $\mathbf{x}_1$ and $\mathbf{x}_2$ have the same p-NN anchors $\mathbf{d}_1$ and $\mathbf{d}_2$. If we adopt Eq. (5), they will end up with the same sparse representation (shown in bracket) because they have the same distances to the anchors, and the same low-dimensional representation because only distance to anchors is considered, even though they might be different. On the contrary, using Eq. (7) will generate the similar representations but with different values, which is more reasonable in reality.

The above analysis clearly states that Eq. (7) and Eq. (6) can lead to non-linear locality-preserving dimension reduction. Then, how to solve Eq. (7) becomes the next problem. Since we are aware of that some elements a_j are definitely zero if $j \notin \mathcal{N}(\mathbf{x})$, it is possible to simplify Eq. (7) by discarding zero elements and only focusing on the possibly non-zero ones:

$$\min_{\tilde{\mathbf{a}}} \ \|\mathbf{x} - \tilde{\mathbf{D}}\tilde{\mathbf{a}}\|_F^2 \quad \text{s.t. } \tilde{a}_i \geq 0, \tag{8}$$

where $\tilde{\mathbf{D}} \in \mathbb{R}^{d\times p}$ is the p-NN of $\mathbf{x}$ in $\mathbf{D}$ and $\tilde{\mathbf{a}} \in \mathbb{R}^p$. Since $\tilde{\mathbf{D}}$ contains mixed signs and $\tilde{\mathbf{a}}$ is constrained to be nonnegative, Eq. (8) is actually a Semi-nonnegative Matrix Factorization (SNMF) problem, which has been extensively studied in [48]. An effective and efficient optimization algorithm for Eq. (8) consists of two steps: 1) $\tilde{\mathbf{a}}$ is randomly initialized by non-negative values, and 2) the following multiplicative updating rule is iteratively applied until $\tilde{\mathbf{a}}$ arrives at a stationary point,

$$\tilde{a}_i \leftarrow \tilde{a}_i \sqrt{\frac{(\tilde{\mathbf{D}}^T\mathbf{x})_i^+ + [(\tilde{\mathbf{D}}^T\tilde{\mathbf{D}})^-\tilde{\mathbf{a}}]_i}{(\tilde{\mathbf{D}}^T\mathbf{x})_i^- + [(\tilde{\mathbf{D}}^T\tilde{\mathbf{D}})^+\tilde{\mathbf{a}}]_i}}, \tag{9}$$

where $\mathbf{M}^+ = \frac{1}{2}(|\mathbf{M}| + \mathbf{M})$ and $\mathbf{M}^- = \frac{1}{2}(|\mathbf{M}| - \mathbf{M})$. The above updating rule guarantees a local convergence of the optimization. Please refer to [48] for more details. In our experiments, we find that 10 to 20 iterations can lead to satisfactory performance because p is usually quite small such that the optimization problem is simple enough in most cases.

B. Optimized Anchor Embedding

Until now, we have introduced the non-linear locality-preserving dimension reduction method, which can exploit the non-linear manifold structure and has an explicit function for out-of-sample data. However, there is a sign function between the low-dimensional representation and the hashcode. In order to preserve manifold structure in the final hashcodes, it is necessary to further consider the influence of the sign function.

From Eq. (6) and Eq. (7) in the previous subsection, it can be observed that a point will fall *close to* the low-dimensional embedding of its p-NN anchors. Hence, the embedding of the anchor set is certainly influential on the quality of hashcodes. We take Figure 2(a) as an example to further explain it. In this figure, red triangles represent embeddings of anchors. The surrounding circles represent points that lie close to the corresponding anchors.[2] In good cases, near points in a circle are in the same quadrant so that they will obtain the same hashcodes after that sign function. In this way, the similarity between data can be preserved. On the contrary, in bad cases,

[2]We use circles for the convenience of illustration. The real-world situation is surely more complicated but intrinsically it has the same problem.

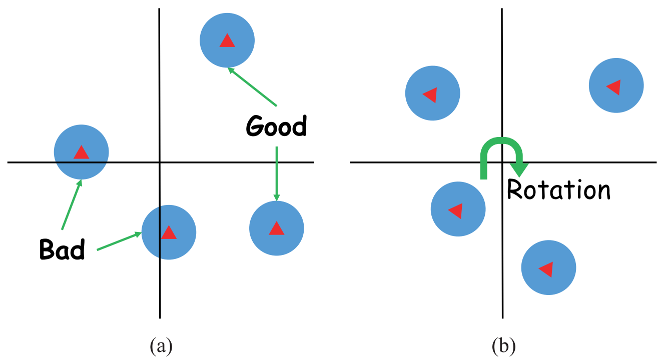


Fig. 2. The influence of anchor embedding. (a) Original embedding. (b) Optimized embedding.

points in a circle may fall into different quadrants resulting in different hashcodes after applying the sign function. In such situations, the similarity is no longer preserved in hashcodes. To avoid the bad cases, we need to adjust the embedding of the anchor set such that it can better preserve the similarity after the sign function while *the initial properties in the embedding are retained*, as illustrated in Figure 2(b). Previous SH methods [6], [16]–[18], [20] mostly ignored the influence of the anchor set but focused on the sparse representation only. From the above discussion, the conclusion is clear: the anchor set embedding plays an important role in SH methods. Next, we continue to introduce how to optimize the anchor embedding.

From Figure 2, we can observe that the bad cases usually happen when the embeddings of anchors lie close to the coordinate axis because such a point by nature is likely to fall into the *other side* of axis and thereby obtain different hashcodes after the sign function. To prevent it, our intuitive idea is to *push the close-to-axis anchors far from axis* while preserving the geometric structure. We carry out a two-step scheme here to implement our idea, in which an anchor-embedding initialization step is followed by an anchor rotation step. In our scheme, the initial embedding of anchors $\mathbf{Y}$ is obtained by means of Laplacian Eigenmap [47] which solves the optimization problem below,

$$\min_{\mathbf{Y}} \ tr(\mathbf{Y}\mathbf{L}\mathbf{Y}^T), \quad \text{s.t. } \mathbf{Y}\mathbf{M}\mathbf{Y}^T = \mathbf{I}_k, \mathbf{Y}\mathbf{M}\mathbf{1}_m = \mathbf{0}, \qquad (10)$$

where $\mathbf{S}_D \in \mathbb{R}^{m\times m}$ is a p_D-NN graph constructed from $\mathbf{D}$, $\mathbf{M}$ is a diagonal matrix with elements $M_{ii} = \sum_j S_{ij}$, and $\mathbf{L} = \mathbf{M} - \mathbf{S}_D$ is the Laplacian of the graph. This problem can be transferred to a generalized eigenvalue problem $\mathbf{L}\mathbf{v} = \lambda\mathbf{M}\mathbf{v}$, and can be solved by selecting the eigenvectors corresponding to the smallest k positive eigenvalues. After the above initialization step, it is very likely that many anchor embeddings are close to axis, which is harmful for hashing as we have explained before. In the second step of our scheme, we apply a rotation to $\mathbf{Y}$ subject to a condition that the optimized anchor embedding $\tilde{\mathbf{Y}}$ after rotation is also the solution to Eq. (10). To do so, one good choice is the exploitation of an orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{k\times k}$ ($\mathbf{R}\mathbf{R}^T = \mathbf{I}_k$ and $\mathbf{R}^T\mathbf{R} = \mathbf{I}_k$), and set $\tilde{\mathbf{Y}} = \mathbf{R}\mathbf{Y}$. Because we have $tr(\mathbf{R}\mathbf{Y}\mathbf{L}\mathbf{Y}^T\mathbf{R}^T) = tr(\mathbf{Y}\mathbf{L}\mathbf{Y}^T)$, $\mathbf{R}\mathbf{Y}\mathbf{M}\mathbf{Y}^T\mathbf{R}^T = \mathbf{R}\mathbf{I}_k\mathbf{R}^T = \mathbf{I}_k$, and $\mathbf{R}\mathbf{Y}\mathbf{M}\mathbf{1}_m = \mathbf{R}\mathbf{0} = \mathbf{0}$, $\tilde{\mathbf{Y}}$ turns out to be also a solution of Eq. (10), meaning that the original geometric structure in $\mathbf{Y}$ is perfectly preserved after a rotation operation.

At this point, our goal becomes finding an orthogonal rotation matrix $\mathbf{R}$ for $\mathbf{Y}$ such that fewer points after the rotation operation (i.e., in $\mathbf{R}\mathbf{Y}$) lie close to the axis, which can be formulated as maximizing the total distance between $\mathbf{R}\mathbf{Y}$ and axis below

$$\max_{\mathbf{R}} \ \mathcal{O} = \sum_{ij} |(\mathbf{R}\mathbf{Y})_{ij}|^r, \quad \text{s.t. } \mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}_k. \qquad (11)$$

In fact, there is still an argument: a rotation can push a close-to-axis anchor far from the axis, and meanwhile, it can also make a far-from-axis anchor closer to the axis. This is true, but the problem is not that vital. Seen from Figure 2, pushing a close-to-axis anchor far is more important, because a subtle change in a close-to-axis anchor can significantly reduce the number of points falling into different quadrants which results in different hashcodes. However, even a huge change in a far-from-axis anchor may not make any difference as long as it is not very close to the axis. In view of this observation, we set the power parameter $r \in (0, 1)$ such that the change in the smaller entries has more effect on $\mathcal{O}$ than the larger entries.

Next, we need to solve this orthogonality constrained optimization problem (11). The basic idea is to construct a *gradient flow in the feasible* set which keeps increasing $\mathcal{O}$ until it reaches a stationary point [49]. Specifically, we adopt an iterative algorithm, in which the rotation $\mathbf{R}$ is randomly initialized. At the t-th iteration, the upgradient of $\mathcal{O}$ at $\mathbf{R}_t$ is:

$$\mathbf{U}_t = -\mathcal{D}\mathcal{O}(\mathbf{R}_t) = -r \cdot \text{sign}(\mathbf{R}_t\mathbf{Y}) \circ |\mathbf{R}_t\mathbf{Y}|^{r-1}\mathbf{Y}^T, \qquad (12)$$

where $\circ$ denotes element-wise multiplication between two matrices, $|\cdot|^{r-1}$ refers to the element-wise power operation for a matrix.[3] A traditional gradient method will move the current point along this direction with a proper step size to obtain the next point. However, the new point will fail to satisfy the constraint, i.e., it is not in the feasible set. Instead, the upgradient is first transformed to a skew-symmetric matrix

$$\mathbf{W}_t = \mathbf{U}_t\mathbf{R}_t^T - \mathbf{R}_t\mathbf{U}_t^T. \qquad (13)$$

We use a Crank-Nicolson-like scheme [50] for the next point:

$$\mathbf{R}_{t+1} = \mathbf{R}_t - \tau\mathbf{W}_t(\frac{\mathbf{R}_t + \mathbf{R}_{t+1}}{2}), \qquad (14)$$

where τ is a step size satisfying Armijo-Wolfe conditions [51]. Solving the above equation offers us the updating rule below:

$$\mathbf{R}_{t+1} = (\mathbf{I}_k + \frac{\tau}{2}\mathbf{W}_t)^{-1}(\mathbf{I}_k - \frac{\tau}{2}\mathbf{W}_t)\mathbf{R}_t. \qquad (15)$$

The above rule is called Cayley transformation. Considering $\mathbf{W}_t$ is a skew-symmetric matrix, i.e., $\mathbf{W}_t^T = -\mathbf{W}_t$, the matrix $\mathbf{I}_k + \frac{\tau}{2}\mathbf{W}_t$ is definitely invertible and $\mathbf{R}_{t+1}$ is orthogonal. Such an updating rule will increase the value of $\mathcal{O}$ until convergence. Please refer to [49, Lemma 3] for the detailed proof. The overall learning algorithm for SHODE is summarized

[3]Because $r \in (0, 1)$, a numeric problem may happen if $(\mathbf{R}\mathbf{Y})_{ij} = 0$. So in the implementation, we add a small number ϵ (say, 10^{-6}) to $|(\mathbf{R}\mathbf{Y})_{ij}|$.

```
Algorithm 1 Learning SHODE
Input: Training features $\mathbf{X}$; Parameters $k$, $m$, $p_D$;
Output: Anchor set $\mathbf{D}$; Projection $\mathbf{P}$;
 1: $\mathbf{D} = \text{Kmeans}(\mathbf{X}, m)$;
 2: Construct $p_D$-NN graph for $\mathbf{D}$;
 3: Compute $\mathbf{S}_D$, $\mathbf{M}$ and $\mathbf{L}$;
 4: Solve Eq. (10) for initial $\mathbf{Y}$;
 5: Initialize $\mathbf{R}_0$ by a random orthogonal matrix, $t = 0$;
 6: repeat
 7:     Compute upgradient $\mathbf{U}_t$ by Eq. (12);
 8:     Compute skew-symmetric matrix $\mathbf{W}_t$ by Eq. (13);
 9:     Update $\mathbf{R}_{t+1}$ by Eq. (15), $t = t + 1$;
10: until Convergence.
11: Return $\mathbf{D}$ and $\mathbf{P} = \mathbf{R}_t\mathbf{Y}$;
```

in Algorithm 1, which at last outputs two key parts for the hashing function ϕ: the anchor set $\mathbf{D}$ and the projection $\mathbf{P}$.

For a new data point $\mathbf{x}$, we first find p-NN from $\mathbf{D}$ and obtain $\tilde{\mathbf{D}}$. Afterwards, we generate sparse representation $\mathbf{a}$ by solving Eq. (8). Next, we obtain a low-dimensional embedding $\mathbf{y} = \mathbf{Pa}$. Finally, the binary hashcode is given by $\mathbf{h} = \text{sign}(\mathbf{y})$.

C. Complexity Analysis

The training time of Algorithm 1 basically consists of 3 parts. The first part is the K-means in line 1. Suppose Kmeans stops at the t_1-th iteration, the time complexity is $\mathcal{O}(nmdt_1)$. The second part is to seek the initial embedding described in lines 2 to 4. Precisely, constructing a p_D-NN graph needs $\mathcal{O}(m^2d + mp_D)$, and solving Eq. (10) requires $\mathcal{O}(mkp_Dt_2)$ if the Lanczos algorithm [52] is adopted, where t_2 means the iteration number which is usually rather small [53]. The third part is learning $\mathbf{R}$, which can be further decomposed into computing $\mathbf{U}_t$ by Eq. (12) ($\mathcal{O}(mk^2)$), computing $\mathbf{W}_t$ by Eq. (13) ($\mathcal{O}(k^3)$), and computing $\mathbf{R}_{t+1}$ by Eq. (15) ($\mathcal{O}(k^3)$). Suppose the iteration depicted from lines 6 to 10 converges at t_3, the total time complexity for learning $\mathbf{R}$ is $\mathcal{O}((mk^2+k^3)t_3)$. Adding them up, the overall complexity will be $\mathcal{O}(nmdt_1 + m^2d + mp_D + mkp_Dt_2 + (mk^2 + k^3)t_3)$.

Given a new data point $\mathbf{x}$, the complexity to generate hashcodes is as follows. Searching p-NN from $\mathbf{D}$ needs $\mathcal{O}(pmd)$. Solving Eq. (8) via Eq. (15) requires $\mathcal{O}((pd + p^2d + p^2)t)$, where t is the number of iterations. And generating the low-dimensional representation by Eq. (6) has the complexity of $\mathcal{O}(pk)$. Therefore, the overall complexity is $\mathcal{O}(pm+(pd+p^2d+p^2)t+pk)$. Because t and p are usually small in practice, this complexity is comparable to the method in [16] and [17].

IV. Experiments

A. Datasets, Metrics, Baselines and Details

To demonstrate the effectiveness of SHODE, we adopt five widely used benchmark datasets for evaluation. The first one is CIFAR-10 [54] consisting of 60,000 images which are manually divided into 10 classes each with 6,000 images. Each image is represented by a 512-dimensional *GIST* [55] feature. The second one is MNIST which has 70,000 images of handwritten digits from '0' to '9'. The 784-dimensional *gray scale* feature is utilized to represent each image. The third dataset is NUS-WIDE [56] with 186,577 images and each image is annotated by at least one of ten classes. Each image is represented by a 500-dimensional *bag-of-visual-words* feature based on SIFT [57]. The forth dataset is SIFT1M [12] which contains more than 1 million *SIFT points*. The fifth dataset is CIFAR-100 which is similar to CIFAR-10. It has 100 classes and each class has 600 images. For CIFAR-100, we adopt the *deep features* for images which are extracted by the ILSVRC2014 challenge winner GoogLeNet [58] pre-trained on ImageNet. Specifically, we adopt the outputs of the last fully-connected layer as the feature for each image which is a 1,024-dimensional vector. For CIFAR-10, MNIST, and CIFAR-100, 10,000 samples are randomly selected as the query set and the remaining samples form the database. For NUS-WIDE, 1% (1,866) images are randomly sampled as the query set, while the remaining images make up the database. We refer to TableII for more detailed statistics of them.

TABLE II
THE STATISTICS OF DATASETS

	#database	#training	#query	#feature
CIFAR-10	50k	10k	10k	512
MNIST	60k	10k	10k	784
NUS-WIDE	$\sim$ 184k	10k	1,866	500
SIFT1M	1m	10k	10k	128
CIFAR-100	50k	10k	10k	1,024

We adopt two retrieval procedures, i.e., Hamming ranking and hash lookup. Hamming ranking first computes the Hamming distance between the query and all points in the database and then sorts points by the distance. Points with smaller distances are first returned. Hamming ranking needs a linear scanning of the database. But since only bit operations are required, it is usually very fast in practice. Hash lookup emphasizes more on retrieval speed because it has constant query time [17] with a single hash table. Following [13], [17], we search within Hamming radius 2 to retrieve neighbors for each query. For a Hamming ranking, we employ Precision-recall curve, Precision curve and Recall curve as evaluation metrics, in which the former shows the precision at different recall levels, the middle reflects the precision level w.r.t. the number of retrieved samples, and the latter reflects the recall level w.r.t. the number of retrieved samples. On top of them, mean Average Precision (mAP) defined as the area under Precision-recall curve is also used. For hash lookup, we use F-measure and Recall within Hamming radius 2 as metrics, in which the former is the harmonic average of precision and recall. For CIFAR-10, MNIST, NUS-WIDE and CIFAR-100, images sharing class labels with the query are considered as true positives. For SIFT1M, following [6], [59], the closest 2 percent of database points to the query measured by the Euclidean distance are defined as the true positives of a query.

We employ the following unsupervised hashing methods as baselines, Anchor Graph Hashing (AGH) [17], Compressed Hashing (CH) [16], Compact Structure Hashing (CSH) [34],

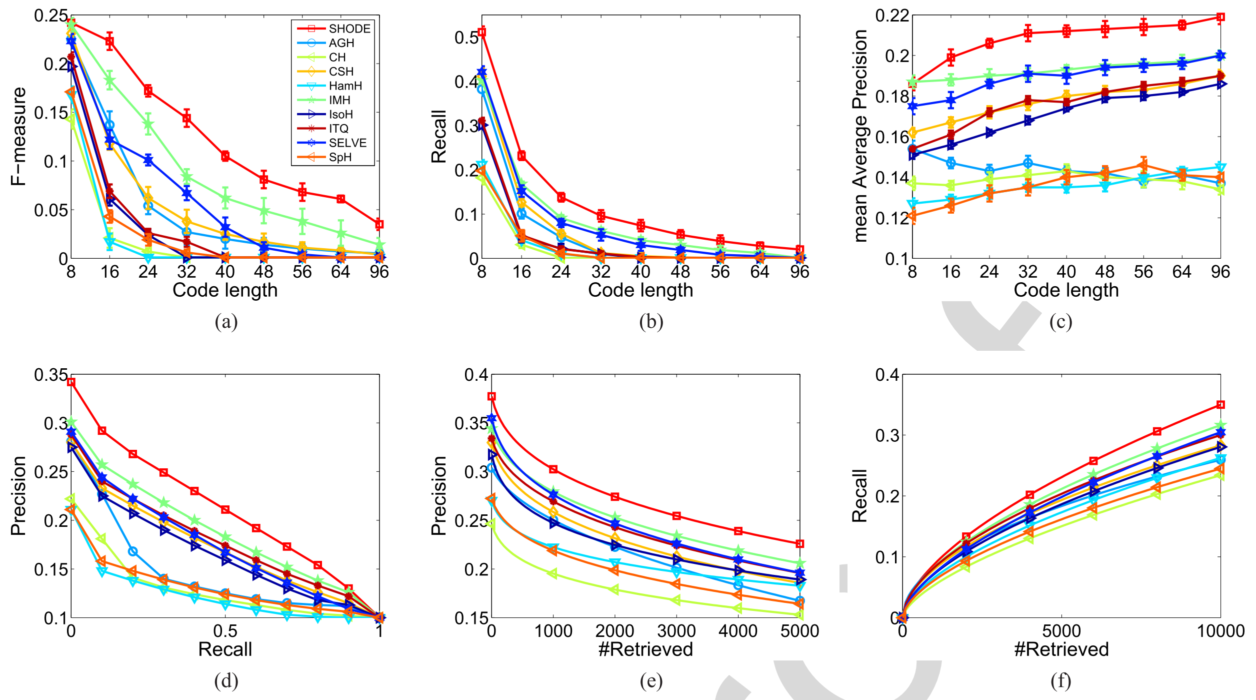


Fig. 3. Results on CIFAR-10 dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

Harmonious Hashing (HamH) [59], Inductive Manifold Hashing (IMH) [6] with LE and ITQ, Isotropic Hashing (IsoH) [60], Iterative Quantization (ITQ) [14], Sparse Embedding and Least Variance Encoding (SELVE) [37], and Spectral Hashing (SpH) [13]. For Ch, CSH, and HamH, we implemented them ourselves. And we used the author-provided codes for the other methods. IMH, AGH, and CH, as well as Sparse Hashing methods like SHODE, rely on two parameters. The first is the size of the anchor set, i.e., m, and the second is p for searching p-NN from anchor set to construct sparse representation $\mathbf{a}$ for a new data point. For a meaningful comparison, we perform grid search ($m \in [100 : 100 : 2000]$ and $p \in [1 : 10]$) and report the best results of them. For the other baselines like ITQ, we use the default settings provided by their authors since most of them do not have important model parameters. Moreover, because this paper focuses on the unsupervised setting where no supervision is provided, thereby not comparing it to the supervised hashing methods, like Kernelized Supervised Hashing [61] and deep hashing methods shown in Section II.

When compared to baselines, we consistently use the following settings. To generate the anchor set, we run K-means and stop at the 100th iteration, and the anchor set size is $m = 1,000$. To generate initial embedding $\mathbf{Y}$ by Laplacian Eigenmap, we set $p_D = 5$ with the Heat kernel. In Algorithm 1, the power parameter r is set to 0.5, p is set to 3 for constructing sparse representation $\mathbf{a}$, and when solving $\tilde{\mathbf{a}}$ iteratively by Eq. (8), we terminate at the 20th iteration. The effect of two key parameters, m and p, will be shown later.

Experiments are conducted on a computer with Intel Core i7-2600 CPU and 16GB RAM. All numeric results reported in this paper are the average values of 25 repeated runs.

B. Results and Discussions

The results on five datasets are shown in Figures 3 to 7 respectively. It can be observed that SHODE significantly outperforms the baselines. Besides, the results also reveal the following important points. 1) SHODE and IMH achieve the best performance, especially when measured by F-measure with long hashcodes. This is because they adopt non-linear projection which can better preserve the manifold structure. In addition, their non-linear function can avoid over-segmentation of space as in linear methods like ITQ, which increases the collision probability in the hashtable. Thus, they can retrieve more points (high recall) with high precision, which highlights the advantage of SH. 2) SHODE takes the influence of anchors on hashcodes into consideration and finds the optimal embedding of anchors, thereby improving the quality of hashcodes. In comparison with other Sparse Hashing methods that completely neglect the effect of anchor embedding, e.g., IMH and AGH, our performance is much better than theirs.

In addition, to evaluate the significance of the improvements by SHODE over the other baseline methods, we perform *paired-sample t-test* on all datasets with different hashcode length. In our experiment, we perform 25 repeated runs for each hashcode length with random data split and all methods follow the same data split. For each method, we take the corresponding mAP values of 25 runs as samples from its mAP distribution, and compare them between algorithms for the significant tests. The significance level is set to 0.01 as a typical value. The results show that the *p-value* in almost all significance tests between SHODE and the other baseline methods is smaller than 10^{-7}, which is far less than the significance level 0.01, indicating that the improve-

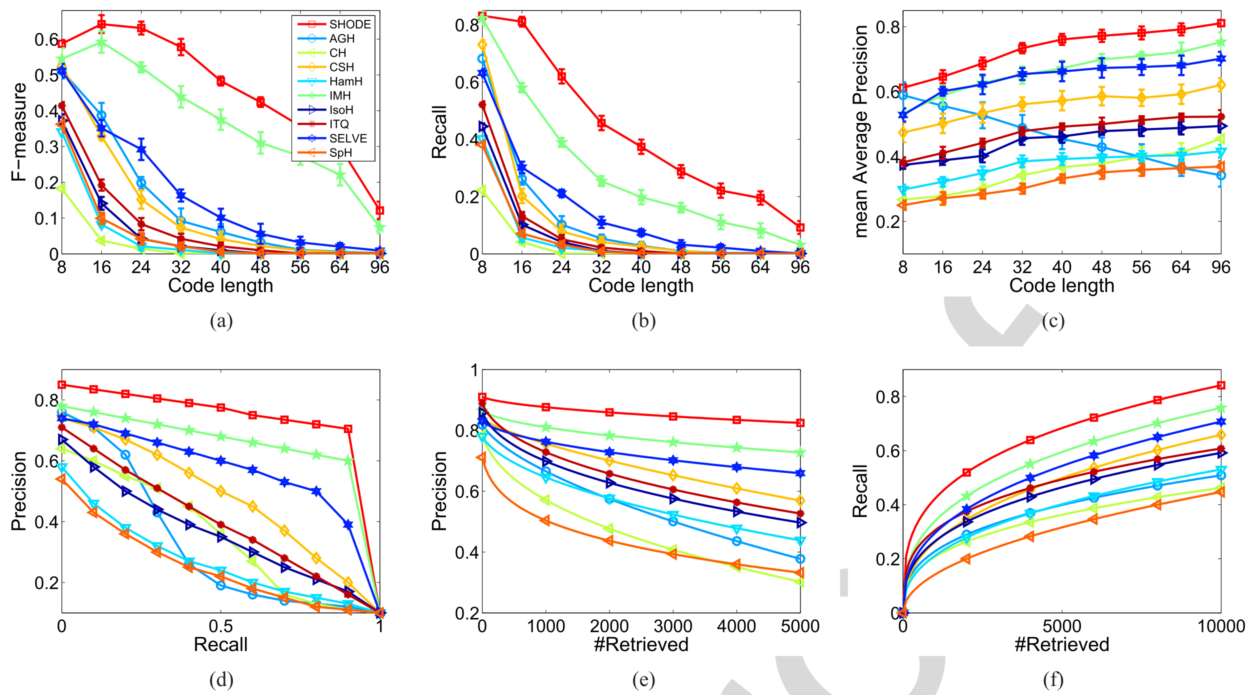


Fig. 4. Results on MNIST dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

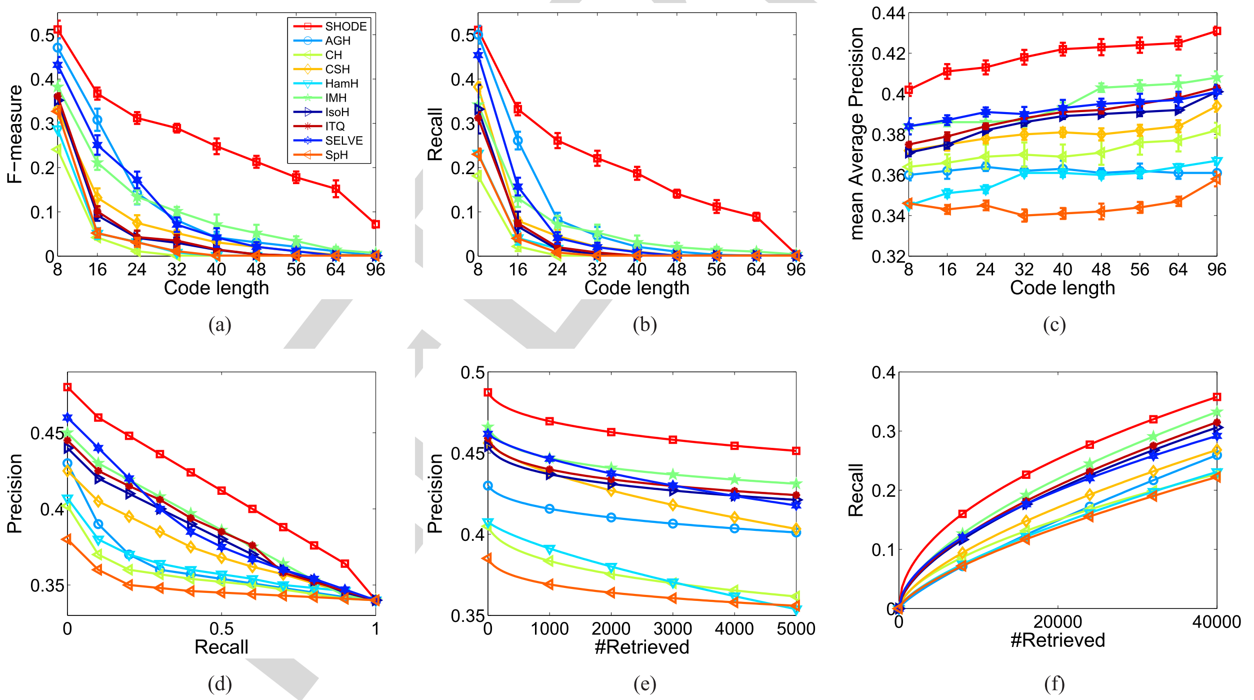


Fig. 5. Results on NUS-WIDE dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

ments gained by SHODE over the baselines are statistically significant.

The effects of m and p on system performance are shown in Figures 8(a) and 8(b) respectively. Seen from the results, on the one hand, if m is too small, the non-linear manifold cannot be well preserved. On the other hand, increasing m can help to improve the performance in the beginning but it will be saturated at a certain point, which means further increase of m after this point does not improve the performance that much. Differently, varying value p within a certain range (e.g., $p < 20$) does not seem to influence the performance dramatically in the sense that the p-mAP curve looks like a

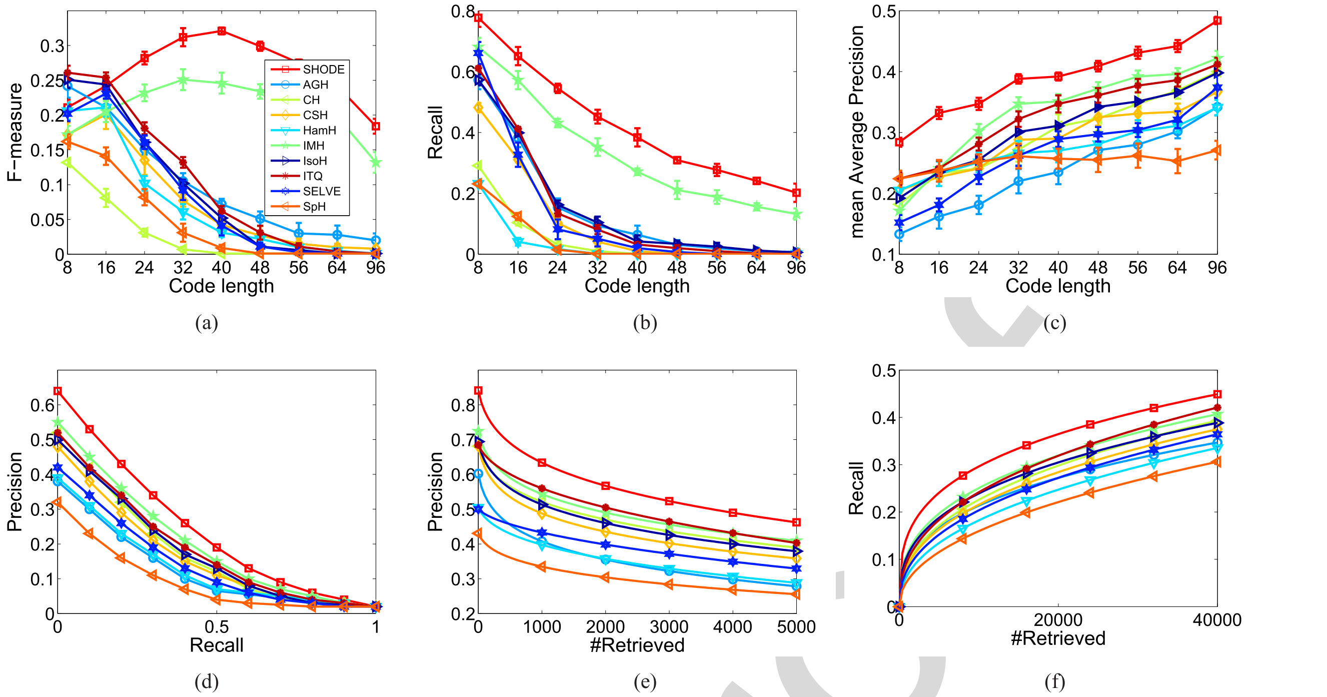


Fig. 6. Results on SIFT1M dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

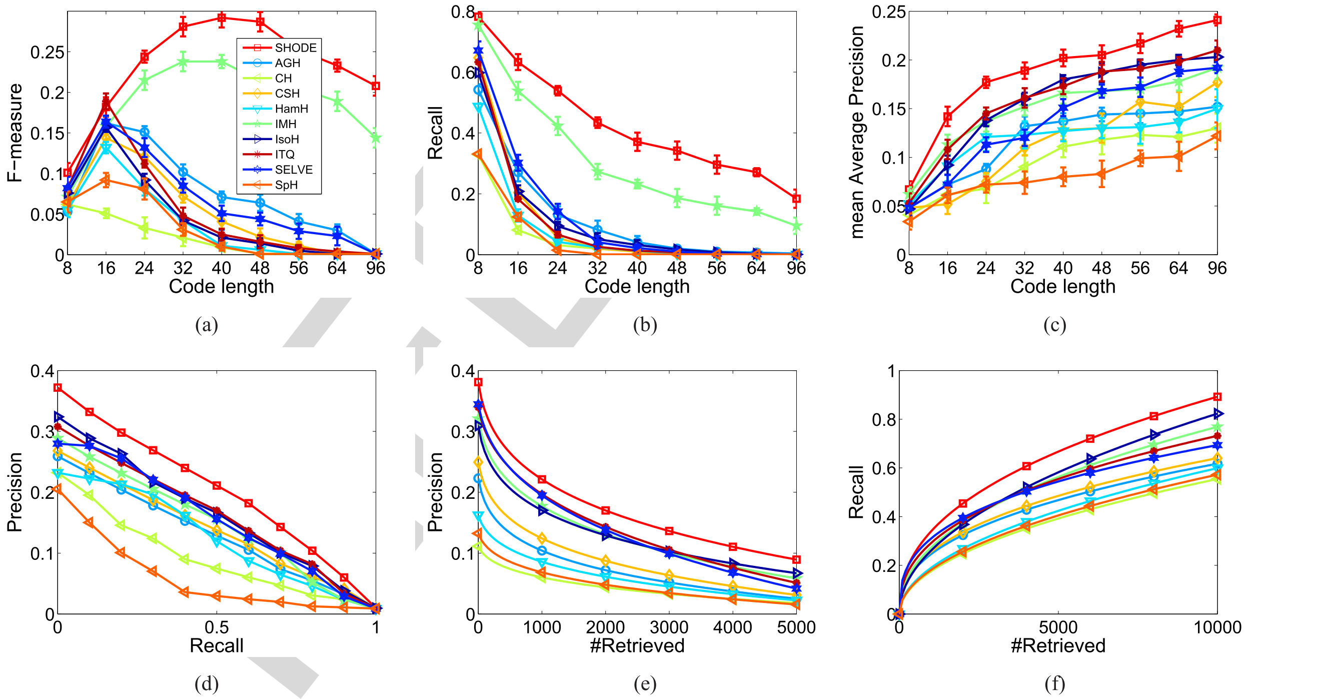


Fig. 7. Results on CIFAR-100 dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

flat line. However, if p is too large (say, 50), anchors not close to data will be selected to compute $\mathbf{a}$, which will break the locality and decrease the performance. Figure 8(c) shows the objective function value in Eq. (11) w.r.t. the number of iterations. We can observe the objective function can increase steadily with more iterations and will converge within 100 iterations, which validates the effectiveness of Algorithm 1.

Figure 8(d) plots the mAP w.r.t. the number of iterations in Algorithm 1. It can be observed that mAP value keeps increasing with more iterations until the algorithm converges. In addition, there is an important result we need to mention that the mAP of SHODE at iteration 0 is much worse than the optimal mAP. In fact, at iteration 0, the anchor embedding is not optimized at all. This phenomenon demonstrates that

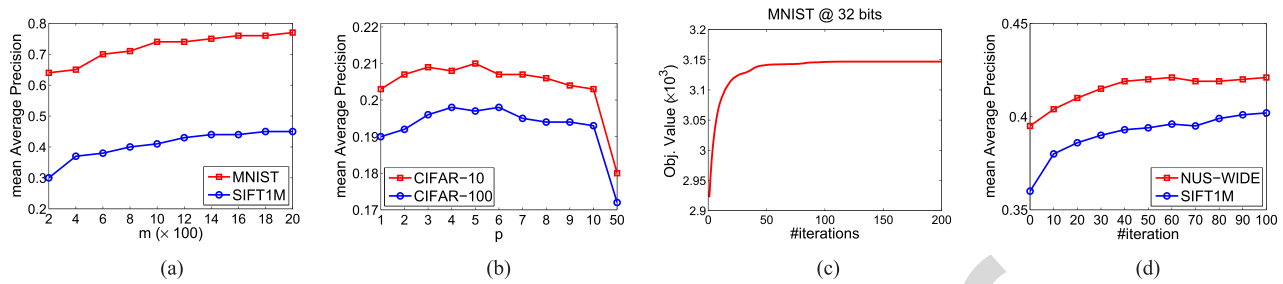


Fig. 8. Effects on parameters (all under 32 bits). (a) Effect of m ($p = 3$). (b) Effect of p ($m = 1,000$). (c) Convergence study. (d) mAP w.r.t. #iteration.

1) the anchor embedding is indeed important for sparse hashing and optimizing the embedding of anchors does lead to higher hashing quality, and 2) with better anchor embedding, SHODE performs better, which is also the motivation of this paper.

V. Conclusion

In this paper, we proposed a novel Sparse Hashing method, namely SHODE, for scalable retrieval. Based on the sparse representation, a non-linear locality-preserving dimension reduction method was presented. Moreover, we discovered the importance of the anchor embedding for Sparse Hashing and proposed a novel method to find the optimized anchor embedding. An efficient learning algorithm was given for optimization. Extensive experiments on five benchmark datasets have verified our motivation and the superiority of SHODE.

References

[1] X. Yang, X. Qian, and T. Mei, "Learning salient visual word for scalable mobile image retrieval," *Pattern Recognit.*, vol. 48, no. 10, pp. 3093–3101, 2015.

[2] X. Yang, X. Qian, and Y. Xue, "Scalable mobile image retrieval by exploring contextual saliency," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1709–1721, Jun. 2015.

[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, 1997.

[4] G. Ding, Y. Guo, J. Zhou, and Y. Gao, "Large-scale cross-modality search via collective matrix factorization hashing," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5427–5440, Sep. 2016.

[5] Z. Lin, G. Ding, J. Han, and J. Wang, "Cross-view retrieval via probability-based semantics-preserving hashing," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2016.2608906.

[6] F. Shen, C. Shen, Q. Shi, A. V. D. Hengel, Z. Tang, and H. T. Shen, "Hashing on nonlinear manifolds," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1839–1851, Jun. 2015.

[7] L. Liu, Z. Lin, L. Shao, F. Shen, G. Ding, and J. Han, "Sequential discrete hashing for scalable cross-modality similarity retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 107–118, Oct. 2017.

[8] X. Lu, X. Zheng, and X. Li, "Latent semantic minimal hashing for image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 355–368, Jan. 2017.

[9] X. Li, Q. Guo, and X. Lu, "Spatiotemporal statistics for video quality assessment," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3329–3342, Jul. 2016.

[10] Y. Guo, G. Ding, X. Jin, and J. Wang, "Learning predictable and discriminative attributes for visual recognition," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 3783–3789.

[11] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2006, pp. 459–468.

[12] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.

[13] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.

[14] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[15] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3419–3427.

[16] Y. Lin, R. Jin, D. Cai, S. Yan, and X. Li, "Compressed hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 446–451.

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

[18] T. Ge, Q. Ke, and J. Sun, "Sparse-coded features for image retrieval," in *Proc. Brit. Mach. Vis. Conf.*, 2013, pp. 1–11.

[19] F. Wu, Z. Yu, Y. Yang, S. Tang, Y. Zhang, and Y. Zhuang, "Sparse multi-modal hashing," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 427–439, Feb. 2014.

[20] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen, "Sparse hashing for fast multimedia search," *ACM Trans. Inf. Syst.*, vol. 31, no. 2, May 2013, Art. no. 9.

[21] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of co-salient objects by looking deep and wide," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 215–232, 2016.

[22] J. Han, D. Zhang, X. Hu, L. Guo, J. Ren, and F. Wu, "Background prior-based salient object detection via deep reconstruction residual," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 8, pp. 1309–1321, Aug. 2015.

[23] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Supervised dictionary learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1033–1040.

[24] Y. Guo, G. Ding, J. Zhou, and Q. Liu, "Robust and discriminative concept factorization for image representation," in *Proc. 5th ACM Int. Conf. Multimedia Retr.*, Shanghai, China, Jun. 2015, pp. 115–122.

[25] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[26] L. Chen, D. Xu, I. W. Tsang, and X. Li, "Spectral embedded hashing for scalable image retrieval," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1180–1190, Jul. 2014.

[27] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3424–3431.

[28] D. Zhang, F. Wang, and L. Si, "Composite hashing with multiple information sources," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2011, pp. 225–234.

[29] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao, "Linear cross-modal hashing for efficient multimedia search," in *Proc. ACM Multimedia Conf.*, 2013, pp. 143–152.

[30] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 153–160.

[31] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 801–808.

[32] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

[33] A. Cherian, "Nearest neighbors using compact sparse codes," in *Proc. 31th Int. Conf. Mach. Learn.*, 2014, pp. 1053–1061.

[34] R. Ye and X. Li, "Compact structure hashing via sparse and similarity preserving embedding," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 718–729, Mar. 2016.

[35] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[36] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[37] X. Zhu, L. Zhang, and Z. Huang, "A sparse embedding and least variance encoding approach to hashing," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[39] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2475–2483.

[40] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3270–3278.

[41] 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.

[42] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2064–2072.

[43] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, Dec. 2016, pp. 1711–1717.

[44] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-UCSD birds-200-2011 dataset," California Inst. Technol., California, CA, USA, Tech. Rep. CNS-TR-2011-001, 2011.

[45] M. Belkin and P. Niyogi, "Using manifold stucture for partially labeled classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 929–936.

[46] B. Shen, B. Liu, Q. Wang, Y. Fang, and J. P. Allebach, "SP-SVM: Large margin classifier for data on multiple manifolds," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2965–2971.

[47] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

[48] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010.

[49] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Math. Program.*, vol. 142, no. 1, pp. 397–434, 2013.

[50] D. Goldfarb, Z. Wen, and W. Yin, "A curvilinear search method for p-harmonic flows on spheres," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 84–109, 2009.

AQ:6 [51] J. Nocedal and S. Wright, *Numerical Optimization*. 1999.

[52] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[53] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2010, pp. 18–25.

[54] A. Krizhevsky, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech Rep., 2009. AQ:7

[55] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[56] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: A real-world Web image database from national University of Singapore," in *Proc. 8th ACM Int. Conf. Image Video Retr.*, 2009, Art. no. 48.

[57] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[58] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[59] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, and D. Cai, "Harmonious hashing," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 1820–1826.

[60] W. Kong and W. Li, "Isotropic hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1655–1663.

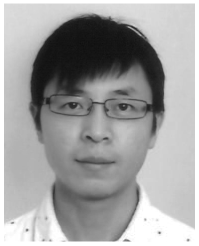
[61] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2074–2081.

Yuchen Guo received the B.Sc. degree from the School of Software, and the B.Ec. degree from the School of Economics and Management, Tsinghua University, Beijing, China, in 2013, where he is currently pursuing the Ph.D. degree with the School of Software. His research interests include multimedia information retrieval, computer vision, and machine learning. AQ:8

Guiguang Ding received the Ph.D. degree in electronic engineering from Xidian University, China, in 2014. In 2006, he has been a Post-Doctoral Research Fellow with the Department of Automation, Tsinghua University. He is currently an Associate Professor with the School of Software, Tsinghua University. He has authored 80 papers AQ:9 in major journals and conferences, including the IEEE TIP, TMM, TKDE, SIG IR, AAAI, ICML, IJCAI, CVPR, and ICCV. His current research centers on the area of multimedia information retrieval, computer vision and machine learning.

Li Liu received the Ph.D. degree from the Department of Electronic and Electrical Engineering, The University of Shefeld, Shefeld, U.K., in 2014. He is currently a Research Fellow with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K.

Jungong Han was a Senior Scientist with Civolution Technology (a combining synergy of Philips CI and Thomson STS) from 2012 to 2015, a Research Staff with the Centre for Mathematics and Computer Science from 2010 to 2012, and a Researcher with the Technical University of Eindhoven, The Netherlands from 2005 to 2010. He is currently a Senior Lecturer with the Department of Computer Science, Northumbria University, U.K.



Ling Shao (M'09–SM'10) was a Professor with Northumbria University from 2014 to 2016, a Senior Lecturer with the University of Sheffield from 2009 to 2014, and a Senior Scientist with Philips Research, The Netherlands, from 2005 to 2009. He is currently a Professor with the School of Computing Sciences, University of East Anglia, Norwich, U.K. His research interests include computer vision, image/video processing, and machine learning. He is a fellow of the British Computer Society and the Institution of Engineering and Technology. He is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS and several other journals.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ:1 = Please note that there were discrepancies between the accepted pdf [SHODE_R1.pdf] and the [bare_jrnl_final.tex] in Reference citations. We have followed [bare_jrnl_final.tex].
AQ:2 = Please confirm/give details of funding source.
AQ:3 = Please confirm whether the corresponding authors information is correct as set.
AQ:4 = Please note that there were discrepancies between the accepted pdf [SHODE_R1.pdf] and the [bare_jrnl_final.tex] in First Footnote. We have followed [bare_jrnl_final.tex].
AQ:5 = Table I is not cited in body text. Please indicate where it should be cited.
AQ:6 = Please provide the publisher name and location for ref. [51].
AQ:7 = Please provide the report no. for ref. [54].
AQ:8 = Please note that there were discrepancies between the accepted pdf [SHODE_R1.pdf] and the [bare_jrnl_final.tex] in biographies. We have followed [bare_jrnl_final.tex].
AQ:9 = Please confirm whether the edits made in the sentence "He has authored …" are OK.

Learning to Hash With Optimized Anchor Embedding for Scalable Retrieval

Yuchen Guo, Guiguang Ding, Li Liu, Jungong Han, and Ling Shao, *Senior Member, IEEE*

***Abstract*— Sparse representation and image hashing are powerful tools for data representation and image retrieval respectively. The combinations of these two tools for scalable image retrieval, i.e., sparse hashing (SH) methods, have been proposed in recent years and the preliminary results are promising. The core of those methods is a scheme that can efficiently embed the (high-dimensional) image features into a low-dimensional Hamming space, while preserving the similarity between features. Existing SH methods mostly focus on finding better sparse representations of images in the hash space. We argue that the anchor set utilized in sparse representation is also crucial, which was unfortunately underestimated by the prior art. To this end, we propose a novel SH method that optimizes the integration of the anchors, such that the features can be better embedded and binarized, termed as Sparse Hashing with Optimized Anchor Embedding. The central idea is to push the anchors far from the axis while preserving their relative positions so as to generate similar hashcodes for neighboring features. We formulate this idea as an orthogonality constrained maximization problem and an efficient and novel optimization framework is systematically exploited. Extensive experiments on five benchmark image data sets demonstrate that our method outperforms several state-of-the-art related methods.**

***Index Terms*— Sparse representation, hashing, retrieval, scalability, orthogonality, optimization.**

I. Introduction

APPROXIMATE Nearest Neighbor (ANN) search has become a fundamental paradigm in various applications, such as image recognition and image retrieval [1], [2]. Its aim is to find some approximate nearest neighbors for a query from a collection of data. To cope with large-scale data, many techniques for fast ANN search have been proposed in the past. One popular pathway is based on trees, e.g. kd-tree [3], which has logarithmic retrieval complexity for low-dimensional data. However, most tree-based methods may reduce to exhaustive linear scanning for high-dimensional data because of the curse of dimensionality. Another pathway, called hashing [4], represents data by a sequence of binary codes. Benefiting from the binary representation, the storage can be dramatically reduced and the search can be quite efficient, even with a large-scale dataset [5]–[10]. With proper designs, hashing will not necessarily degrade the search accuracy. In view of the above advantages, hashing methods have drawn increasing attention recently from the industry and academia.

The key problem in hashing is how to embed the original features, which are usually *high-dimensional floating-point number* representations, into the *low-dimensional binary* Hamming space while the similarity between the original features can be preserved. Locality Sensitive Hashing (LSH) [11], as the most notable and fundamental hashing method, adopts *random* projections to generate hashcodes. Theoretically, the Hamming distance between those hashcodes can progressively approximate the Euclidean distance between the original features. But in practice, very long hashcodes (say, 1, 024 bits) are required in this approach so as to achieve satisfactory performance. To address this issue, several *learning* based methods have been proposed, such as PCA Hashing [12], Spectral Hashing [13], and Iterative Quantization [14]. Though better performance can be obtained, compared to LSH, these methods still suffer from two shortcomings due to the *linear* projections employed by them: 1) they may fail to preserve the *non-linear* manifold structure of data; and 2) they may achieve high precision but *low recall* as the feature space is segmented so finely that data may be scatted in the Hamming space, which leads to extremely low collision probability [15].

Alternatively, methods exploiting *non-linear* projections [6], [16], [17] have gained increasing popularity due to their superior performance. Specifically, these methods, thanks to the non-linear projections, can better preserve the complicated geometric structure of data, especially the manifold structure. One representative framework is called Sparse Hashing (SH) [6], [16]–[20] since it is based on the Sparse Coding (SC) that was successfully used in image representation [21], [22], classification [23], and denoising [24]. Basically, the algorithm is carried out by two forms of transformation. First, a non-linear transformation converts the original features to the sparse representations. Second, a linear transformation further transfers the sparse representations generated in the previous step to the Hamming space. Generally, non-linear SH methods are capable of overcoming two shortcomings of the linear methods if a proper learning strategy is deployed. However, these two problems, i.e., how to generate effective sparse representations for hashing and how to transform the sparse

AQ:1 AQ:2 AQ:3 AQ:4

Manuscript received July 30, 2016; revised December 2, 2016; accepted December 28, 2016. This work was supported in part by the National Natural Science Foundation of China under Grant 61571269 and Grant 61271394, in part by the National Basic Research Project of China under Grant 2015CB352300, and in part by the Royal Society Newton Mobility under Grant IE150997. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Lei Zhang. *(Corresponding authors: Guiguang Ding; Jungong Han.)*

Y. Guo and G. Ding are with the School of Software, Tsinghua University, Beijing 100084, China (e-mail: yuchen.w.guo@gmail.com; dinggg@tsinghua.edu.cn).

L. Liu and J. Han are with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K. (e-mail: li2.liu@northumbria.ac.uk; jungong.han@northumbria.ac.uk).

L. Shao is with the School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, U.K. (e-mail: ling.shao@ieee.org).

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/TIP.2017.2652730

representation into the Hamming space with data similarity preserved, still need to be solved.

In this paper, we propose a novel SH method, aiming at preserving the non-linear manifold structure of the original features in the Hamming space. In particular, motivated by Locally Linear Embedding (LLE) [25] and Anchor Graph [17], we learn a *non-linear locality-preserving* dimension reduction function via the sparse representation of data. This non-linear function secures similar low-dimensional representations for neighboring points. After such an effective dimension reduction, we can easily generate binary hashcodes from the embedded low-dimensional features. When learning this function, previous works [6], [16]–[18], [20] only looked into the sparse representation of data but ignored the importance of the anchors [17] utilized in constructing the sparse representation. We notice that the low-dimensional embedding of the anchors has a significant impact on the hash function. Specifically, it is discovered that pushing anchors far from axis while preserving the geometric structure of them during the anchor embedding usually leads to high-quality hashcodes. We investigate this phenomenon and mathematically formulate the implementation of this idea to an orthogonality constrained maximization problem which optimizes the anchor embedding with the aim to avoid generating two different hashcodes for neighboring low-dimensional points. With such an optimization, the locality of original features can be well preserved and better ANN search performance can be achieved. Moreover, we put forward an efficient learning algorithm to solve the complicated orthogonality constrained optimization problem.

The rest of this paper is organized as follows. In Section II, we briefly describe some preliminaries and review the related hashing works. The proposed SHODE is introduced detailedly in Section III. The experimental results and discussion are given in Section IV, and we draw conclusions in Section V.

II. Preliminaries and Related Work

A. Formulation

Given a set of d-dimensional features $\mathbf{X} = [\mathbf{x}_1, ..., \mathbf{x}_n] \in \mathbb{R}^{d\times n}$, we can design a hash function $h(\cdot)$ to generate k-bit *binary* representations, i.e., hashcodes, for them as $\mathbf{b}_i = h(\mathbf{x}_i) \in \{-1, 1\}^k$,[1] such that the similarity between features can be preserved, i.e., similar features have similar hashcodes. This idea can be formulated as the following learning problem,

$$\min_h \sum_{i,j} s_{ij} d_H(h(\mathbf{x}_i), h(\mathbf{x}_j)), \quad \text{s.t. } \mathcal{C}(h), \tag{1}$$

where d_H is the Hamming distance between hashcodes, s_{ij} is the similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$, and $\mathcal{C}(h)$ is the constraints applied to h, for example, we always expect the hashcodes to be balanced ($\sum_i \mathbf{b}_i = \mathbf{0}_k$) and uncorrelated ($\mathbf{B}\mathbf{B}^T = n\mathbf{I}_k$).

Since it is difficult, if not impossible, to design an effective hash function by directly converting $\mathbf{X}$ to hashcodes, a two-step strategy is widely adopted [12]–[14], [16]. In the first step, the original features $\mathbf{X}$ are projected into a k-dimensional space as $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_n] \in \mathbb{R}^{k\times n}$ by a projection function $\phi(\cdot)$. Because we usually have $k < d$, this step can be regarded as a dimension reduction step. Then, the low-dimensional embedded representations $\mathbf{Y}$ are quantified into binary codes by, in most cases, the sign function as $\mathbf{B} = [\mathbf{b}_1, ..., \mathbf{b}_n] = \text{sign}(\mathbf{Y})$, where $\text{sign}(x) = 1$ if $x > 0$ or -1 otherwise. By doing so, the overall hash function becomes $h(\cdot) = \text{sign}(\phi(\cdot))$. In this way, learning h can be achieved by learning ϕ instead. However, the sign function still makes the learning intractable in many cases [13]. A common solution is to remove the sign function and to further *relax* the learning problem as a real-valued problem,

$$\min_\phi \sum_{i,j} s_{ij} d(\phi(\mathbf{x}_i), \phi(\mathbf{x}_j)), \quad \text{s.t. } \mathcal{C}(\phi). \tag{2}$$

[1]In implementation, we can use $\{0, 1\}$. In fact, these two representations are equivalent. So we use $\{-1, 1\}$ in this paper for convenience as in [17].

TABLE I
Notations and Descriptions in This Paper

Notation	Description	Notation	Description
$\mathbf{X}$	data matrix	n	#samples
$\mathbf{Y}$	projected matrix	d	#features
$\mathbf{B}$	binary Hashcode	k	#Hashcode
$\mathbf{P}$	projection matrix	m	#anchors
$\mathbf{D}$	anchor set	p	#NN
$\mathbf{A}$	sparse matrix	t	#iterations
$\mathbf{S}$	similarity matrix	h, ϕ, ρ	functions
$\mathbf{R}$	rotation matrix	τ	step size

AQ:5

B. Linear Hashing

Several methods [13], [16], [26]–[29] assume a linear projection for ϕ, i.e., $\phi(\mathbf{x}) = \mathbf{P}\mathbf{x}$, where $\mathbf{P} \in \mathbb{R}^{k\times d}$ is a linear projection matrix. After proper algebra operations and transformations, the learning problem can be rewritten into a simple formulation as follows:

$$\max_{\mathbf{P}} \; tr(\mathbf{P}\mathbf{X}\mathbf{S}\mathbf{X}^T\mathbf{P}^T), \quad \text{s.t. } \mathbf{P}\mathbf{P}^T = \mathbf{I}_k, \tag{3}$$

where $tr(\cdot)$ is the trace function, $\mathbf{S} = [s_{ij}]$ is the similarity matrix among training samples, and the orthogonal constraint requires the selected directions to be uncorrelated. $\mathbf{S}$ determines what kind of information is preserved depending on the intentions of different methods. The statistics reveal that the majority of existing works choose to preserve the local manifold structure of data [13], [30]. After the above assumption and operations, the problem defined in Eq. (3) can be easily solved. However, since only linear projections are used, these methods may still fail to preserve the similarity.

C. Sparse Hashing

To preserve the non-linear manifold structure, Sparse Hashing [6], [16]–[18], [20], which learns a non-linear ϕ, has attracted considerable attention. Given a set of anchors $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_m] \in \mathbb{R}^{d\times m}$, a sparse presentation $\mathbf{A} = [\mathbf{a}_1, ..., \mathbf{a}_n] \in \mathbb{R}^{m\times n}$ is constructed by $\mathbf{A} = \rho(\mathbf{X}, \mathbf{D})$. This can be done by conventional sparse reconstruction [31] as

$$\min_{\mathbf{A}} \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 + \mathcal{R}(\mathbf{A}), \quad \text{s.t. } \mathcal{C}(\mathbf{A}), \tag{4}$$

where $\mathcal{R}(\mathbf{A})$ denotes regularization terms, such as ℓ_1-norm regularization for sparsity, and other terms like Graph regularization [32], and $\mathcal{C}(\mathbf{A})$ is a constraint on $\mathbf{A}$. Obviously, this method is non-linear. in [19] and [33], such schemes are employed, and the sparse codes are then encoded into a set of integers which are composed of the nonzero indices. This index set sacrifices the advantages of efficient storage and speedy binary code matching. Alternatively, in [20], Zhu *et al.* proposed an encoding method in which the binary codes are generated by setting nonzero elements in $\mathbf{A}$ as 1 and the others as 0. The problem of this method lies in its incapability of generating compact and balanced representations because of the sparsity of $\mathbf{A}$, thereby degrading the quality of hashcodes. In addition, Ye and Li [34] proposed the Compact Structure Hashing that combines the linear projection learning in Eq. (3) and sparse reconstruction in Eq. (4) in a unified objective function to simultaneously exploits the non-linear structure of data and finds the optimal projection function. However, this method intrinsically adopts a linear projection to the Hamming space such that it still suffers from the low-recall problem.

A possible way of solving this problem is the usage of the Anchor Graph [17], in which each anchor is either randomly sampled from the data or the cluster centroids after applying a data clustering algorithm, such as Kmeans. The sparse representation can be build in the Anchor Graph as follows:

$$a_{ji} = \begin{cases} \dfrac{\exp(-\|\mathbf{x}_i - \mathbf{d}_j\|^2/\sigma^2)}{\sum_{j' \in \mathcal{N}(\mathbf{x}_i)} \exp(-\|\mathbf{x}_i - \mathbf{d}_{j'}\|^2/\sigma^2)}, & \forall j \in \mathcal{N}(\mathbf{x}_i) \\ 0, & \text{otherwise}, \end{cases} \tag{5}$$

where $\mathcal{N}(\mathbf{x}_i)$ is the p-NN of $\mathbf{x}_i$ in $\mathbf{D}$ and σ is the bandwidth parameter. The obtained sparse representation is claimed to preserve the similarity between data. Obviously, $\mathbf{a}_i$ has at most p nonzero elements, implying that $\mathbf{a}$ is sparse. Finally, $\phi(\cdot)$ is constructed by projecting the sparse representation to a low-dimensional space, i.e., $\phi(\mathbf{x}) = \mathbf{P}\rho(\mathbf{x}, \mathbf{D})$. To preserve the similarity, Liu *et al.* [17] proposed the Anchor Graph Hashing that constructs $\mathbf{P}$ by solving an eigenvalue problem on the Anchor Graph. Lin *et al.* [16] proposed the Compressed Hashing in which the sampled p_{ij} from $\mathcal{N}(0, 1/k)$ can construct a projection satisfying Restricted Isometry Property [35] in Compressed Sensing theory [36]. Similarly, Shen *et al.* [6] proposed an inductive method to construct $\mathbf{P}$. Zhu *et al.* [37] proposed a sparse embedding and least variance encoding approach to hashing, which constructs $\mathbf{P}$ by solving a reconstruction problem and adjusts the projected representation to minimize the variance for preserving similarity. Even though promising results have been obtained, how to design effective ρ and $\mathbf{P}$ is still an open issue, which is the focus of this paper.

Moreover, it is noticed that in recent years many works have attempted to combine the deep convolutional neural network [38] with hashing, i.e., deep hashing [39]–[43]. For example, Liong *et al.* [39] proposed a deep hashing method in which the output of the networks is required to preserve the supervised similarity. Lai *et al.* [40] proposed a *piece-wise* function for the network to address the discrete optimization problem in deep hashing. Zhang *et al.* [41] presented a network using similarity regularized triplet loss for person re-identification. However, it should be pointed out that these deep hashing approaches should be categorized into the *supervised hashing* methods in which supervised knowledge (*e.g.*, label information) is required for model training. As is known to all, collecting sufficient supervised knowledge is expensive in many applications [44]. On the contrary, this paper, and many SH methods focus on the *unsupervised hashing* which only exploits the intrinsic unsupervised information of data and thus they are free from the lack of the supervised knowledge.

III. The Proposed Method

Our method follows the framework of SH. Firstly, we construct a sparse representation for the original features in a non-linear manner. Secondly, we linearly project the sparse representation into the low-dimensional space. Thirdly, we obtain hashcodes from low-dimensional embedding using the sign function. The special properties of our projection are 1) the low-dimensional embedding preserves the local manifold structure of original data, and 2) the similarity structure is preserved as well after the sign quantization. The following two subsections will elaborate on them one by one. Since all involved steps take data similarity preservation into account, the obtained hashcodes, without saying, will naturally preserve the similarity relationship of original features, thus resulting in superior ANN search and image retrieval performance.

A. Locality-Preserving Dimension Reduction

In this subsection, we will provide an effective method for non-linear dimension reduction based on Sparse Coding which can well preserve the non-linear local manifold structure.

Locality-preserving dimension reduction aims to find low-dimensional embedding which can preserve the neighborhood structure or manifold structure of the original data. One representative and seminal work is Locally Linear Embedding (LLE) [25] which can find a linear embedding for non-linear manifold. However, LLE does not provide an explicit dimension reduction function for the out-of-sample data (data which is not in the training set). Another celebrated method is called Locality Preserving Projections (LPP) [30] which learns an explicit linear projection function instead. Despite its ability of easily addressing the out-of-sample data, the linear function adopted by LPP may perform worse than the non-linear ones.

Although LLE does not provide the projection function for out-of-sample data, it still reveals an important property of the non-linear manifold: local linearity. That is, the manifold structure is locally linear even though it is non-linear globally. Such a property is also utilized in [45] and [46], which can be further interpreted below. Given some points $\mathbf{D} = [\mathbf{d}_1, ..., \mathbf{d}_m]$ and their corresponding low-dimensional embeddings $\mathbf{Y} = [\mathbf{y}_1, ..., \mathbf{y}_m]$ obtained by non-linear methods like LLE, the low-dimensional embedding $\mathbf{y}$ for a new data point $\mathbf{x}$ is given by

$$\mathbf{y} \leftarrow \sum_{i \in \mathcal{N}(\mathbf{x})} a_i \mathbf{y}_i, \tag{6}$$

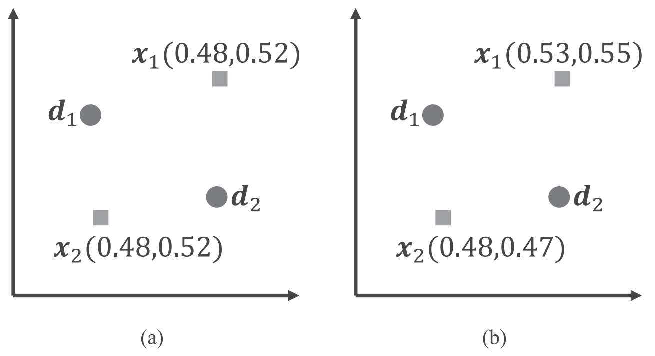


Fig. 1. Sparse representation by different methods. (a) by Eq. (5). (b) by Eq. (7).

where $\mathcal{N}(\mathbf{x})$ is the p-NN of $\mathbf{x}$ in $\mathbf{D}$, and a_i is the corresponding weight. One straightforward way to compute a_i is based on Eq. (5). But it should be noticed that such a formulation only defines the weight and does not reflect the *relative position* between $\mathbf{x}$ and $\mathcal{N}(\mathbf{x})$. Therefore, the embedding $\mathbf{y}$ relying on the weight may lose important information. Therefore, to make use of the local linearity better, in this paper, we propose to generate $\mathbf{a}$ by a sparse reconstruction procedure as follows:

$$\min_{\mathbf{a}} \ \|\mathbf{x}-\mathbf{D}\mathbf{a}\|_F^2, \quad \text{s.t. } a_i \geq 0, \ a_j = 0 \text{ if } j \notin \mathcal{N}(\mathbf{x}). \quad (7)$$

Here, we require $\mathbf{a}$ to be nonnegative so that it can serve as "weight". Moreover, only $\mathcal{N}(\mathbf{x})$ is used to reconstruct $\mathbf{x}$ for preserving the locality. Obviously, the solution $\mathbf{a}$ is sparse in the sense that it has at most p nonzero elements ($p \ll m$).

By combining Eq. (6) and Eq. (7), the overall dimension reduction can be summarized as follows: 1) An anchor set $\mathbf{D}$ is generated from training data by K-means clustering; 2) We find the locality preserving embedding $\mathbf{Y}$ for it by a non-linear method, called Laplacian Eigenmap [47]. As this step is only conducted for the anchor set, there is no need to learn a projection function for the out-of-sample data; 3) For a new data point $\mathbf{x}$, the sparse representation $\mathbf{a}$ is obtained by solving Eq. (7); 4) The low-dimensional embedding $\mathbf{y}$ is obtained by Eq. (6). As a result, the projection function $\mathbf{P}$ in our method can be considered as the low-dimensional embedding $\mathbf{Y}$ of the anchor set. Due to the non-linearity in Eq. (7), the entire procedure is non-linear as in LLE. Meanwhile, it also has an explicit projection function (Eq. (6) and (7)) for out-of-sample data. Hence, it can be concluded that our method combines the advantages of LLE and LPP but gets rid of their shortcomings.

Seen from Eq. (6) and Eq. (7), two points that are close in the original feature space will also have similar low-dimensional representations after the projection, because they will choose similar p-NN anchor sets from $\mathbf{D}$. In other words, these two points will finally lie very close to the embeddings of their corresponding anchor sets, which are also similar.

Here, we discuss the difference between our sparse representation constructed by Eq. (7) and the widely used version expressed in Eq. (5). In principle, representations based on Eq. (5) fail to consider the relative position of $\mathbf{x}$ and $\mathcal{N}(\mathbf{x})$ while using Eq. (7) can achieve this goal. An intuitive illustration is shown in Figure 1, in which $\mathbf{x}_1$ and $\mathbf{x}_2$ have the same p-NN anchors $\mathbf{d}_1$ and $\mathbf{d}_2$. If we adopt Eq. (5), they will end up with the same sparse representation (shown in bracket) because they have the same distances to the anchors, and the same low-dimensional representation because only distance to anchors is considered, even though they might be different. On the contrary, using Eq. (7) will generate the similar representations but with different values, which is more reasonable in reality.

The above analysis clearly states that Eq. (7) and Eq. (6) can lead to non-linear locality-preserving dimension reduction. Then, how to solve Eq. (7) becomes the next problem. Since we are aware of that some elements a_j are definitely zero if $j \notin \mathcal{N}(\mathbf{x})$, it is possible to simplify Eq. (7) by discarding zero elements and only focusing on the possibly non-zero ones:

$$\min_{\tilde{\mathbf{a}}} \ \|\mathbf{x}-\tilde{\mathbf{D}}\tilde{\mathbf{a}}\|_F^2 \quad \text{s.t. } \tilde{a}_i \geq 0, \quad (8)$$

where $\tilde{\mathbf{D}} \in \mathbb{R}^{d\times p}$ is the p-NN of $\mathbf{x}$ in $\mathbf{D}$ and $\tilde{\mathbf{a}} \in \mathbb{R}^p$. Since $\tilde{\mathbf{D}}$ contains mixed signs and $\tilde{\mathbf{a}}$ is constrained to be nonnegative, Eq. (8) is actually a Semi-nonnegative Matrix Factorization (SNMF) problem, which has been extensively studied in [48]. An effective and efficient optimization algorithm for Eq. (8) consists of two steps: 1) $\tilde{\mathbf{a}}$ is randomly initialized by non-negative values, and 2) the following multiplicative updating rule is iteratively applied until $\tilde{\mathbf{a}}$ arrives at a stationary point,

$$\tilde{a}_i \leftarrow \tilde{a}_i \sqrt{\frac{(\tilde{\mathbf{D}}^T\mathbf{x})_i^+ + [(\tilde{\mathbf{D}}^T\tilde{\mathbf{D}})^-\tilde{\mathbf{a}}]_i}{(\tilde{\mathbf{D}}^T\mathbf{x})_i^- + [(\tilde{\mathbf{D}}^T\tilde{\mathbf{D}})^+\tilde{\mathbf{a}}]_i}}, \quad (9)$$

where $\mathbf{M}^+ = \frac{1}{2}(|\mathbf{M}| + \mathbf{M})$ and $\mathbf{M}^- = \frac{1}{2}(|\mathbf{M}| - \mathbf{M})$. The above updating rule guarantees a local convergence of the optimization. Please refer to [48] for more details. In our experiments, we find that 10 to 20 iterations can lead to satisfactory performance because p is usually quite small such that the optimization problem is simple enough in most cases.

B. Optimized Anchor Embedding

Until now, we have introduced the non-linear locality-preserving dimension reduction method, which can exploit the non-linear manifold structure and has an explicit function for out-of-sample data. However, there is a sign function between the low-dimensional representation and the hashcode. In order to preserve manifold structure in the final hashcodes, it is necessary to further consider the influence of the sign function.

From Eq. (6) and Eq. (7) in the previous subsection, it can be observed that a point will fall *close to* the low-dimensional embedding of its p-NN anchors. Hence, the embedding of the anchor set is certainly influential on the quality of hashcodes. We take Figure 2(a) as an example to further explain it. In this figure, red triangles represent embeddings of anchors. The surrounding circles represent points that lie close to the corresponding anchors.[2] In good cases, near points in a circle are in the same quadrant so that they will obtain the same hashcodes after that sign function. In this way, the similarity between data can be preserved. On the contrary, in bad cases,

[2]We use circles for the convenience of illustration. The real-world situation is surely more complicated but intrinsically it has the same problem.

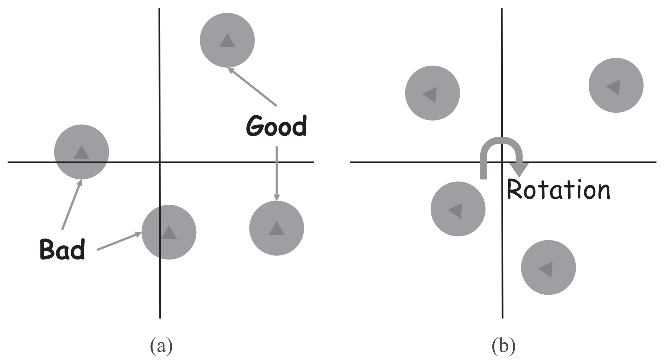


Fig. 2. The influence of anchor embedding. (a) Original embedding. (b) Optimized embedding.

points in a circle may fall into different quadrants resulting in different hashcodes after applying the sign function. In such situations, the similarity is no longer preserved in hashcodes. To avoid the bad cases, we need to adjust the embedding of the anchor set such that it can better preserve the similarity after the sign function while *the initial properties in the embedding are retained*, as illustrated in Figure 2(b). Previous SH methods [6], [16]–[18], [20] mostly ignored the influence of the anchor set but focused on the sparse representation only. From the above discussion, the conclusion is clear: the anchor set embedding plays an important role in SH methods. Next, we continue to introduce how to optimize the anchor embedding.

From Figure 2, we can observe that the bad cases usually happen when the embeddings of anchors lie close to the coordinate axis because such a point by nature is likely to fall into the *other side* of axis and thereby obtain different hashcodes after the sign function. To prevent it, our intuitive idea is to *push the close-to-axis anchors far from axis* while preserving the geometric structure. We carry out a two-step scheme here to implement our idea, in which an anchor-embedding initialization step is followed by an anchor rotation step. In our scheme, the initial embedding of anchors $\mathbf{Y}$ is obtained by means of Laplacian Eigenmap [47] which solves the optimization problem below,

$$\min_{\mathbf{Y}} \; tr(\mathbf{YLY}^T), \quad \text{s.t. } \mathbf{YMY}^T = \mathbf{I}_k, \mathbf{YM1}_m = \mathbf{0}, \qquad (10)$$

where $\mathbf{S}_D \in \mathbb{R}^{m\times m}$ is a p_D-NN graph constructed from $\mathbf{D}$, $\mathbf{M}$ is a diagonal matrix with elements $M_{ii} = \sum_j S_{ij}$, and $\mathbf{L} = \mathbf{M} - \mathbf{S}_D$ is the Laplacian of the graph. This problem can be transferred to a generalized eigenvalue problem $\mathbf{Lv} = \lambda\mathbf{Mv}$, and can be solved by selecting the eigenvectors corresponding to the smallest k positive eigenvalues. After the above initialization step, it is very likely that many anchor embeddings are close to axis, which is harmful for hashing as we have explained before. In the second step of our scheme, we apply a rotation to $\mathbf{Y}$ subject to a condition that the optimized anchor embedding $\tilde{\mathbf{Y}}$ after rotation is also the solution to Eq. (10). To do so, one good choice is the exploitation of an orthogonal rotation matrix $\mathbf{R} \in \mathbb{R}^{k\times k}$ ($\mathbf{RR}^T = \mathbf{I}_k$ and $\mathbf{R}^T\mathbf{R} = \mathbf{I}_k$), and set $\tilde{\mathbf{Y}} = \mathbf{RY}$. Because we have $tr(\mathbf{RYLY}^T\mathbf{R}^T) = tr(\mathbf{YLY}^T)$, $\mathbf{RYMY}^T\mathbf{R}^T = \mathbf{RI}_k\mathbf{R}^T = \mathbf{I}_k$, and $\mathbf{RYM1}_m = \mathbf{R0} = \mathbf{0}$, $\tilde{\mathbf{Y}}$ turns out to be also a solution of Eq. (10), meaning that the original geometric structure in $\mathbf{Y}$ is perfectly preserved after a rotation operation.

At this point, our goal becomes finding an orthogonal rotation matrix $\mathbf{R}$ for $\mathbf{Y}$ such that fewer points after the rotation operation (i.e., in $\mathbf{RY}$) lie close to the axis, which can be formulated as maximizing the total distance between $\mathbf{RY}$ and axis below

$$\max_{\mathbf{R}} \; \mathcal{O} = \sum_{ij} |(\mathbf{RY})_{ij}|^r, \quad \text{s.t. } \mathbf{RR}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I}_k. \qquad (11)$$

In fact, there is still an argument: a rotation can push a close-to-axis anchor far from the axis, and meanwhile, it can also make a far-from-axis anchor closer to the axis. This is true, but the problem is not that vital. Seen from Figure 2, pushing a close-to-axis anchor far is more important, because a subtle change in a close-to-axis anchor can significantly reduce the number of points falling into different quadrants which results in different hashcodes. However, even a huge change in a far-from-axis anchor may not make any difference as long as it is not very close to the axis. In view of this observation, we set the power parameter $r \in (0, 1)$ such that the change in the smaller entries has more effect on $\mathcal{O}$ than the larger entries.

Next, we need to solve this orthogonality constrained optimization problem (11). The basic idea is to construct a *gradient flow in the feasible* set which keeps increasing $\mathcal{O}$ until it reaches a stationary point [49]. Specifically, we adopt an iterative algorithm, in which the rotation $\mathbf{R}$ is randomly initialized. At the t-th iteration, the upgradient of $\mathcal{O}$ at $\mathbf{R}_t$ is:

$$\mathbf{U}_t = -\mathcal{DO}(\mathbf{R}_t) = -r \cdot \text{sign}(\mathbf{R}_t\mathbf{Y}) \circ |\mathbf{R}_t\mathbf{Y}|^{r-1}\mathbf{Y}^T, \qquad (12)$$

where $\circ$ denotes element-wise multiplication between two matrices, $|\cdot|^{r-1}$ refers to the element-wise power operation for a matrix.[3] A traditional gradient method will move the current point along this direction with a proper step size to obtain the next point. However, the new point will fail to satisfy the constraint, i.e., it is not in the feasible set. Instead, the upgradient is first transformed to a skew-symmetric matrix

$$\mathbf{W}_t = \mathbf{U}_t\mathbf{R}_t^T - \mathbf{R}_t\mathbf{U}_t^T. \qquad (13)$$

We use a Crank-Nicolson-like scheme [50] for the next point:

$$\mathbf{R}_{t+1} = \mathbf{R}_t - \tau\mathbf{W}_t(\frac{\mathbf{R}_t + \mathbf{R}_{t+1}}{2}), \qquad (14)$$

where τ is a step size satisfying Armijo-Wolfe conditions [51]. Solving the above equation offers us the updating rule below:

$$\mathbf{R}_{t+1} = (\mathbf{I}_k + \frac{\tau}{2}\mathbf{W}_t)^{-1}(\mathbf{I}_k - \frac{\tau}{2}\mathbf{W}_t)\mathbf{R}_t. \qquad (15)$$

The above rule is called Cayley transformation. Considering $\mathbf{W}_t$ is a skew-symmetric matrix, i.e., $\mathbf{W}_t^T = -\mathbf{W}_t$, the matrix $\mathbf{I}_k + \frac{\tau}{2}\mathbf{W}_t$ is definitely invertible and $\mathbf{R}_{t+1}$ is orthogonal. Such an updating rule will increase the value of $\mathcal{O}$ until convergence. Please refer to [49, Lemma 3] for the detailed proof. The overall learning algorithm for SHODE is summarized

[3]Because $r \in (0, 1)$, a numeric problem may happen if $(\mathbf{RY})_{ij} = 0$. So in the implementation, we add a small number ϵ (say, 10^{-6}) to $|(\mathbf{RY})_{ij}|$.

Algorithm 1 Learning SHODE

```
Input: Training features $\mathbf{X}$; Parameters $k$, $m$, $p_D$;
Output: Anchor set $\mathbf{D}$; Projection $\mathbf{P}$;
 1: $\mathbf{D} = \text{Kmeans}(\mathbf{X}, m)$;
 2: Construct $p_D$-NN graph for $\mathbf{D}$;
 3: Compute $\mathbf{S}_D$, $\mathbf{M}$ and $\mathbf{L}$;
 4: Solve Eq. (10) for initial $\mathbf{Y}$;
 5: Initialize $\mathbf{R}_0$ by a random orthogonal matrix, $t = 0$;
 6: repeat
 7:     Compute upgradient $\mathbf{U}_t$ by Eq. (12);
 8:     Compute skew-symmetric matrix $\mathbf{W}_t$ by Eq. (13);
 9:     Update $\mathbf{R}_{t+1}$ by Eq. (15), $t = t + 1$;
10: until Convergence.
11: Return $\mathbf{D}$ and $\mathbf{P} = \mathbf{R}_t\mathbf{Y}$;
```

in Algorithm 1, which at last outputs two key parts for the hashing function ϕ: the anchor set $\mathbf{D}$ and the projection $\mathbf{P}$.

For a new data point $\mathbf{x}$, we first find p-NN from $\mathbf{D}$ and obtain $\tilde{\mathbf{D}}$. Afterwards, we generate sparse representation $\mathbf{a}$ by solving Eq. (8). Next, we obtain a low-dimensional embedding $\mathbf{y} = \mathbf{Pa}$. Finally, the binary hashcode is given by $\mathbf{h} = \text{sign}(\mathbf{y})$.

C. Complexity Analysis

The training time of Algorithm 1 basically consists of 3 parts. The first part is the K-means in line 1. Suppose Kmeans stops at the t_1-th iteration, the time complexity is $\mathcal{O}(nmdt_1)$. The second part is to seek the initial embedding described in lines 2 to 4. Precisely, constructing a p_D-NN graph needs $\mathcal{O}(m^2d + mp_D)$, and solving Eq. (10) requires $\mathcal{O}(mkp_Dt_2)$ if the Lanczos algorithm [52] is adopted, where t_2 means the iteration number which is usually rather small [53]. The third part is learning $\mathbf{R}$, which can be further decomposed into computing $\mathbf{U}_t$ by Eq. (12) ($\mathcal{O}(mk^2)$), computing $\mathbf{W}_t$ by Eq. (13) ($\mathcal{O}(k^3)$), and computing $\mathbf{R}_{t+1}$ by Eq. (15) ($\mathcal{O}(k^3)$). Suppose the iteration depicted from lines 6 to 10 converges at t_3, the total time complexity for learning $\mathbf{R}$ is $\mathcal{O}((mk^2+k^3)t_3)$. Adding them up, the overall complexity will be $\mathcal{O}(nmdt_1 + m^2d + mp_D + mkp_Dt_2 + (mk^2 + k^3)t_3)$.

Given a new data point $\mathbf{x}$, the complexity to generate hashcodes is as follows. Searching p-NN from $\mathbf{D}$ needs $\mathcal{O}(pmd)$. Solving Eq. (8) via Eq. (15) requires $\mathcal{O}((pd + p^2d + p^2)t)$, where t is the number of iterations. And generating the low-dimensional representation by Eq. (6) has the complexity of $\mathcal{O}(pk)$. Therefore, the overall complexity is $\mathcal{O}(pm+(pd+p^2d+p^2)t+pk)$. Because t and p are usually small in practice, this complexity is comparable to the method in [16] and [17].

IV. Experiments

A. Datasets, Metrics, Baselines and Details

To demonstrate the effectiveness of SHODE, we adopt five widely used benchmark datasets for evaluation. The first one is CIFAR-10 [54] consisting of 60,000 images which are manually divided into 10 classes each with 6,000 images. Each image is represented by a 512-dimensional *GIST* [55] feature. The second one is MNIST which has 70,000 images of handwritten digits from ‘0’ to ‘9’. The 784-dimensional *gray scale* feature is utilized to represent each image. The third dataset is NUS-WIDE [56] with 186,577 images and each image is annotated by at least one of ten classes. Each image is represented by a 500-dimensional *bag-of-visual-words* feature based on SIFT [57]. The forth dataset is SIFT1M [12] which contains more than 1 million *SIFT points*. The fifth dataset is CIFAR-100 which is similar to CIFAR-10. It has 100 classes and each class has 600 images. For CIFAR-100, we adopt the *deep features* for images which are extracted by the ILSVRC2014 challenge winner GoogLeNet [58] pre-trained on ImageNet. Specifically, we adopt the outputs of the last fully-connected layer as the feature for each image which is a 1,024-dimensional vector. For CIFAR-10, MNIST, and CIFAR-100, 10,000 samples are randomly selected as the query set and the remaining samples form the database. For NUS-WIDE, 1% (1,866) images are randomly sampled as the query set, while the remaining images make up the database. We refer to TableII for more detailed statistics of them.

TABLE II
The Statistics of Datasets

	#database	#training	#query	#feature
CIFAR-10	50k	10k	10k	512
MNIST	60k	10k	10k	784
NUS-WIDE	$\sim$ 184k	10k	1,866	500
SIFT1M	1m	10k	10k	128
CIFAR-100	50k	10k	10k	1,024

We adopt two retrieval procedures, i.e., Hamming ranking and hash lookup. Hamming ranking first computes the Hamming distance between the query and all points in the database and then sorts points by the distance. Points with smaller distances are first returned. Hamming ranking needs a linear scanning of the database. But since only bit operations are required, it is usually very fast in practice. Hash lookup emphasizes more on retrieval speed because it has constant query time [17] with a single hash table. Following [13], [17], we search within Hamming radius 2 to retrieve neighbors for each query. For a Hamming ranking, we employ Precision-recall curve, Precision curve and Recall curve as evaluation metrics, in which the former shows the precision at different recall levels, the middle reflects the precision level w.r.t. the number of retrieved samples, and the latter reflects the recall level w.r.t. the number of retrieved samples. On top of them, mean Average Precision (mAP) defined as the area under Precision-recall curve is also used. For hash lookup, we use F-measure and Recall within Hamming radius 2 as metrics, in which the former is the harmonic average of precision and recall. For CIFAR-10, MNIST, NUS-WIDE and CIFAR-100, images sharing class labels with the query are considered as true positives. For SIFT1M, following [6], [59], the closest 2 percent of database points to the query measured by the Euclidean distance are defined as the true positives of a query.

We employ the following unsupervised hashing methods as baselines, Anchor Graph Hashing (AGH) [17], Compressed Hashing (CH) [16], Compact Structure Hashing (CSH) [34],

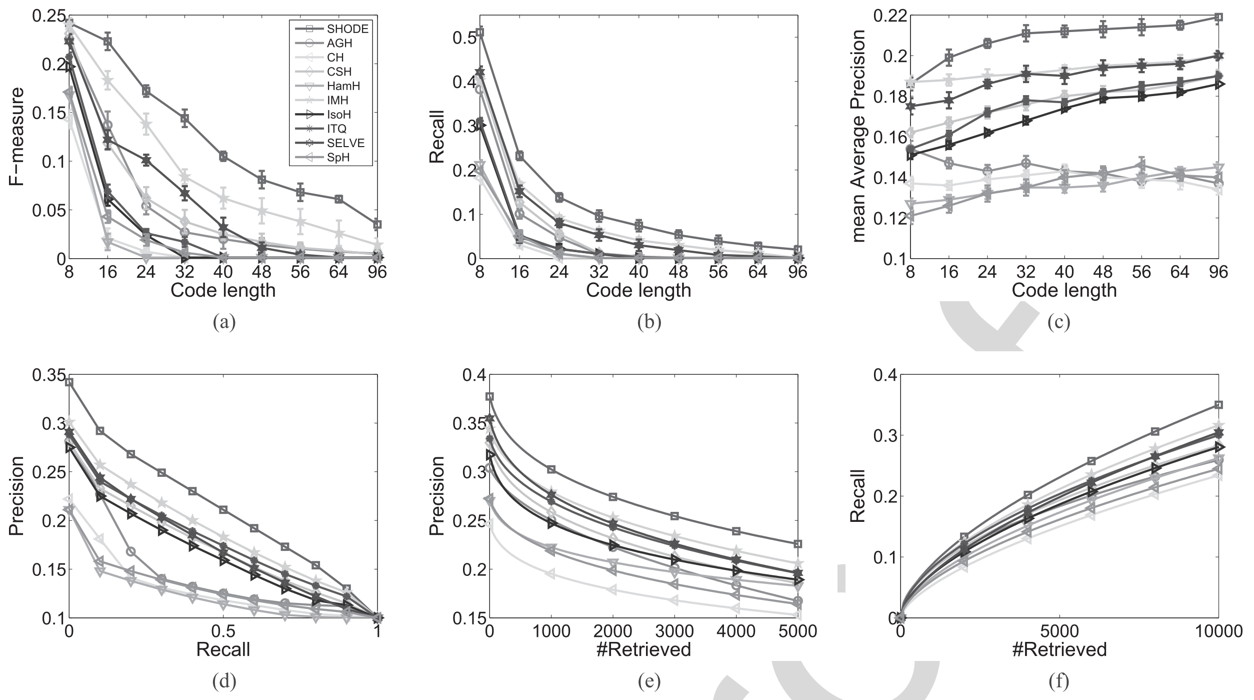


Fig. 3. Results on CIFAR-10 dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

Harmonious Hashing (HamH) [59], Inductive Manifold Hashing (IMH) [6] with LE and ITQ, Isotropic Hashing (IsoH) [60], Iterative Quantization (ITQ) [14], Sparse Embedding and Least Variance Encoding (SELVE) [37], and Spectral Hashing (SpH) [13]. For Ch, CSH, and HamH, we implemented them ourselves. And we used the author-provided codes for the other methods. IMH, AGH, and CH, as well as Sparse Hashing methods like SHODE, rely on two parameters. The first is the size of the anchor set, i.e., m, and the second is p for searching p-NN from anchor set to construct sparse representation $\mathbf{a}$ for a new data point. For a meaningful comparison, we perform grid search ($m \in [100 : 100 : 2000]$ and $p \in [1 : 10]$) and report the best results of them. For the other baselines like ITQ, we use the default settings provided by their authors since most of them do not have important model parameters. Moreover, because this paper focuses on the unsupervised setting where no supervision is provided, thereby not comparing it to the supervised hashing methods, like Kernelized Supervised Hashing [61] and deep hashing methods shown in Section II.

When compared to baselines, we consistently use the following settings. To generate the anchor set, we run K-means and stop at the 100th iteration, and the anchor set size is $m = 1,000$. To generate initial embedding $\mathbf{Y}$ by Laplacian Eigenmap, we set $p_D = 5$ with the Heat kernel. In Algorithm 1, the power parameter r is set to 0.5, p is set to 3 for constructing sparse representation $\mathbf{a}$, and when solving $\tilde{\mathbf{a}}$ iteratively by Eq. (8), we terminate at the 20th iteration. The effect of two key parameters, m and p, will be shown later.

Experiments are conducted on a computer with Intel Core i7-2600 CPU and 16GB RAM. All numeric results reported in this paper are the average values of 25 repeated runs.

B. Results and Discussions

The results on five datasets are shown in Figures 3 to 7 respectively. It can be observed that SHODE significantly outperforms the baselines. Besides, the results also reveal the following important points. 1) SHODE and IMH achieve the best performance, especially when measured by F-measure with long hashcodes. This is because they adopt non-linear projection which can better preserve the manifold structure. In addition, their non-linear function can avoid over-segmentation of space as in linear methods like ITQ, which increases the collision probability in the hashtable. Thus, they can retrieve more points (high recall) with high precision, which highlights the advantage of SH. 2) SHODE takes the influence of anchors on hashcodes into consideration and finds the optimal embedding of anchors, thereby improving the quality of hashcodes. In comparison with other Sparse Hashing methods that completely neglect the effect of anchor embedding, e.g., IMH and AGH, our performance is much better than theirs.

In addition, to evaluate the significance of the improvements by SHODE over the other baseline methods, we perform *paired-sample t-test* on all datasets with different hashcode length. In our experiment, we perform 25 repeated runs for each hashcode length with random data split and all methods follow the same data split. For each method, we take the corresponding mAP values of 25 runs as samples from its mAP distribution, and compare them between algorithms for the significant tests. The significance level is set to 0.01 as a typical value. The results show that the *p-value* in almost all significance tests between SHODE and the other baseline methods is smaller than 10^{-7}, which is far less than the significance level 0.01, indicating that the improve-

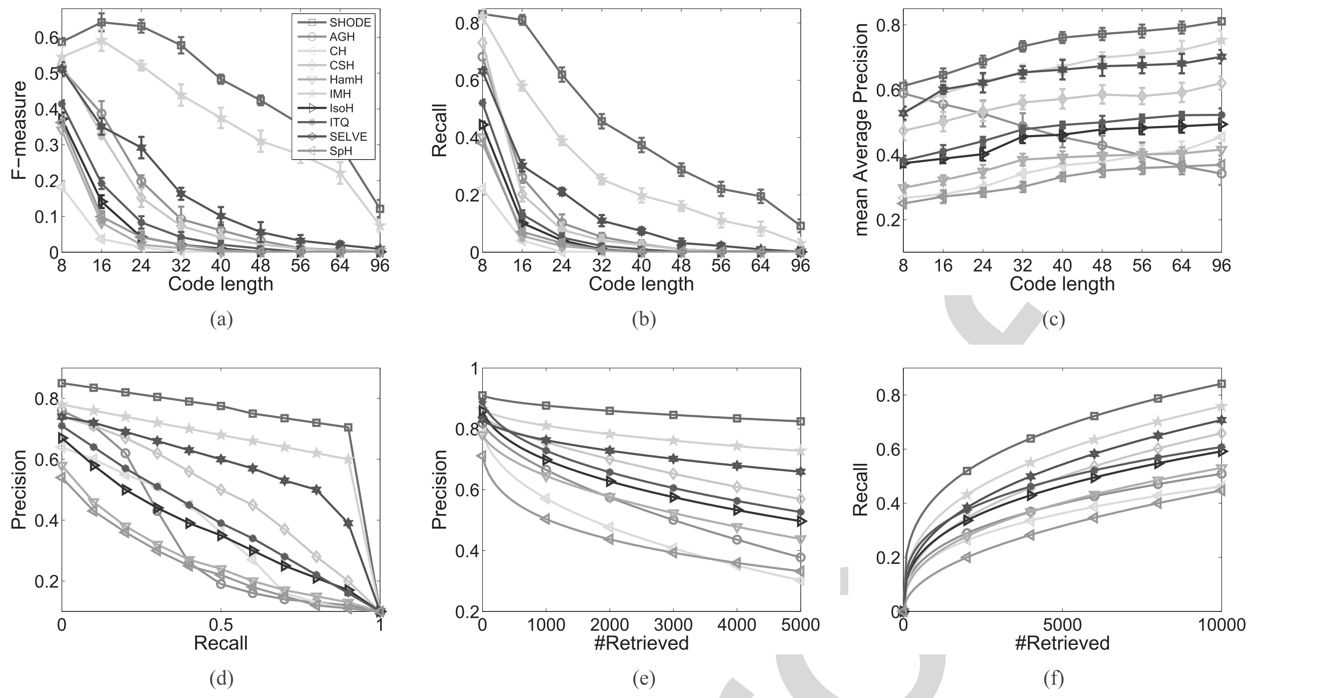


Fig. 4. Results on MNIST dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

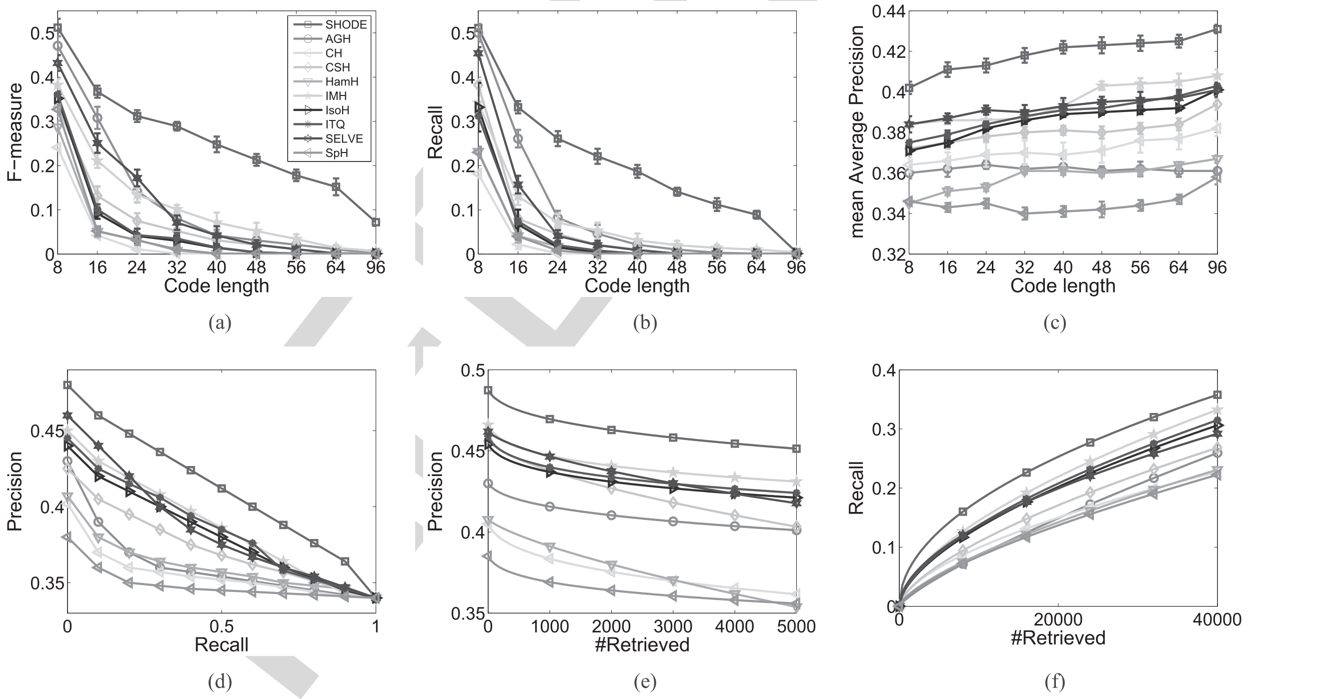


Fig. 5. Results on NUS-WIDE dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

ments gained by SHODE over the baselines are statistically significant.

The effects of m and p on system performance are shown in Figures 8(a) and 8(b) respectively. Seen from the results, on the one hand, if m is too small, the non-linear manifold cannot be well preserved. On the other hand, increasing m can help to improve the performance in the beginning but it will be saturated at a certain point, which means further increase of m after this point does not improve the performance that much. Differently, varying value p within a certain range (e.g., $p < 20$) does not seem to influence the performance dramatically in the sense that the p-mAP curve looks like a

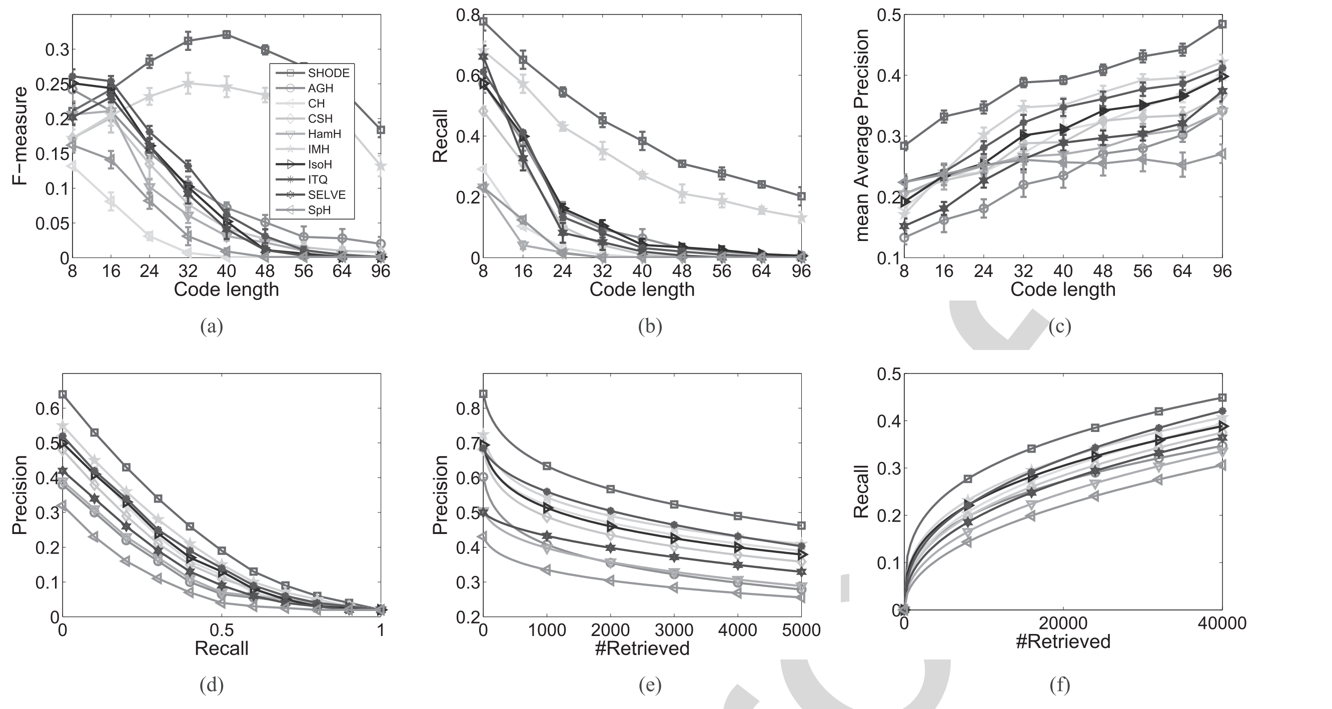


Fig. 6. Results on SIFT1M dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

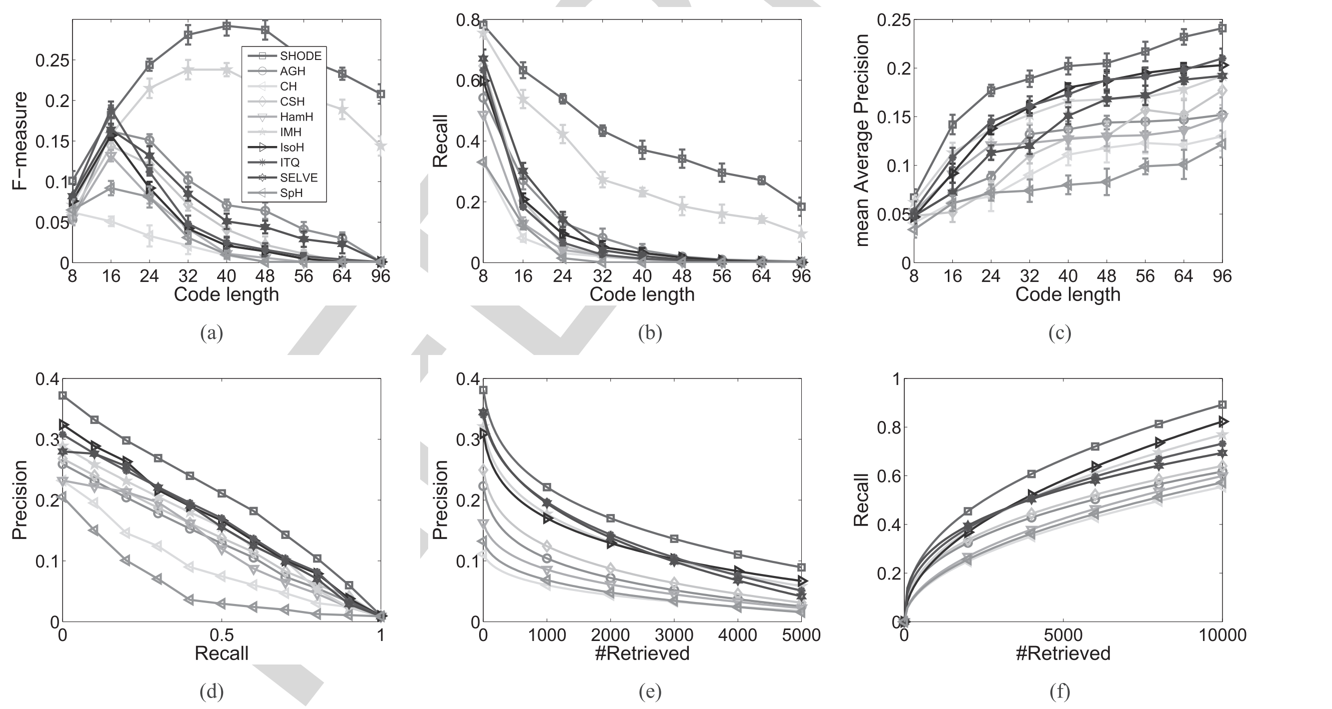


Fig. 7. Results on CIFAR-100 dataset. (a) F-measure in Hamming 2. (b) Recall in Hamming 2. (c) mean Average Precision. (d) Precision-recall curve, 64 bits. (e) Precision curve, 64 bits. (f) Recall curve, 64 bits.

flat line. However, if p is too large (say, 50), anchors not close to data will be selected to compute $\mathbf{a}$, which will break the locality and decrease the performance. Figure 8(c) shows the objective function value in Eq. (11) w.r.t. the number of iterations. We can observe the objective function can increase steadily with more iterations and will converge within 100 iterations, which validates the effectiveness of Algorithm 1.

Figure 8(d) plots the mAP w.r.t. the number of iterations in Algorithm 1. It can be observed that mAP value keeps increasing with more iterations until the algorithm converges. In addition, there is an important result we need to mention that the mAP of SHODE at iteration 0 is much worse than the optimal mAP. In fact, at iteration 0, the anchor embedding is not optimized at all. This phenomenon demonstrates that

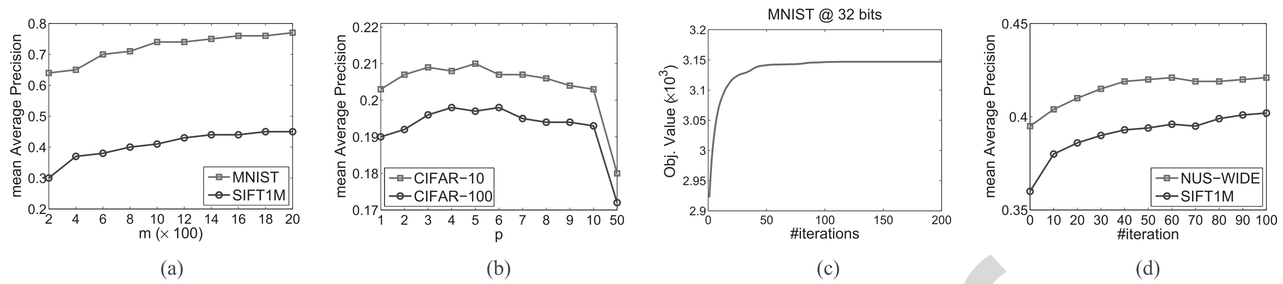


Fig. 8. Effects on parameters (all under 32 bits). (a) Effect of m ($p = 3$). (b) Effect of p ($m = 1,000$). (c) Convergence study. (d) mAP w.r.t. #iteration.

1) the anchor embedding is indeed important for sparse hashing and optimizing the embedding of anchors does lead to higher hashing quality, and 2) with better anchor embedding, SHODE performs better, which is also the motivation of this paper.

V. Conclusion

In this paper, we proposed a novel Sparse Hashing method, namely SHODE, for scalable retrieval. Based on the sparse representation, a non-linear locality-preserving dimension reduction method was presented. Moreover, we discovered the importance of the anchor embedding for Sparse Hashing and proposed a novel method to find the optimized anchor embedding. An efficient learning algorithm was given for optimization. Extensive experiments on five benchmark datasets have verified our motivation and the superiority of SHODE.

References

[1] X. Yang, X. Qian, and T. Mei, "Learning salient visual word for scalable mobile image retrieval," *Pattern Recognit.*, vol. 48, no. 10, pp. 3093–3101, 2015.

[2] X. Yang, X. Qian, and Y. Xue, "Scalable mobile image retrieval by exploring contextual saliency," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1709–1721, Jun. 2015.

[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, 1997.

[4] G. Ding, Y. Guo, J. Zhou, and Y. Gao, "Large-scale cross-modality search via collective matrix factorization hashing," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5427–5440, Sep. 2016.

[5] Z. Lin, G. Ding, J. Han, and J. Wang, "Cross-view retrieval via probability-based semantics-preserving hashing," *IEEE Trans. Cybern.*, to be published. doi: 10.1109/TCYB.2016.2608906.

[6] F. Shen, C. Shen, Q. Shi, A. V. D. Hengel, Z. Tang, and H. T. Shen, "Hashing on nonlinear manifolds," *IEEE Trans. Image Process.*, vol. 24, no. 6, pp. 1839–1851, Jun. 2015.

[7] L. Liu, Z. Lin, L. Shao, F. Shen, G. Ding, and J. Han, "Sequential discrete hashing for scalable cross-modality similarity retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 107–118, Oct. 2017.

[8] X. Lu, X. Zheng, and X. Li, "Latent semantic minimal hashing for image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 355–368, Jan. 2017.

[9] X. Li, Q. Guo, and X. Lu, "Spatiotemporal statistics for video quality assessment," *IEEE Trans. Image Process.*, vol. 25, no. 7, pp. 3329–3342, Jul. 2016.

[10] Y. Guo, G. Ding, X. Jin, and J. Wang, "Learning predictable and discriminative attributes for visual recognition," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 3783–3789.

[11] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. Annu. IEEE Symp. Found. Comput. Sci.*, Oct. 2006, pp. 459–468.

[12] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid, "Aggregating local image descriptors into compact codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 9, pp. 1704–1716, Sep. 2012.

[13] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.

[14] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[15] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3419–3427.

[16] Y. Lin, R. Jin, D. Cai, S. Yan, and X. Li, "Compressed hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 446–451.

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

[18] T. Ge, Q. Ke, and J. Sun, "Sparse-coded features for image retrieval," in *Proc. Brit. Mach. Vis. Conf.*, 2013, pp. 1–11.

[19] F. Wu, Z. Yu, Y. Yang, S. Tang, Y. Zhang, and Y. Zhuang, "Sparse multi-modal hashing," *IEEE Trans. Multimedia*, vol. 16, no. 2, pp. 427–439, Feb. 2014.

[20] X. Zhu, Z. Huang, H. Cheng, J. Cui, and H. T. Shen, "Sparse hashing for fast multimedia search," *ACM Trans. Inf. Syst.*, vol. 31, no. 2, May 2013, Art. no. 9.

[21] D. Zhang, J. Han, C. Li, J. Wang, and X. Li, "Detection of co-salient objects by looking deep and wide," *Int. J. Comput. Vis.*, vol. 120, no. 2, pp. 215–232, 2016.

[22] J. Han, D. Zhang, X. Hu, L. Guo, J. Ren, and F. Wu, "Background prior-based salient object detection via deep reconstruction residual," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 25, no. 8, pp. 1309–1321, Aug. 2015.

[23] J. Mairal, F. R. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Supervised dictionary learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1033–1040.

[24] Y. Guo, G. Ding, J. Zhou, and Q. Liu, "Robust and discriminative concept factorization for image representation," in *Proc. 5th ACM Int. Conf. Multimedia Retr.*, Shanghai, China, Jun. 2015, pp. 115–122.

[25] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[26] L. Chen, D. Xu, I. W. Tsang, and X. Li, "Spectral embedded hashing for scalable image retrieval," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1180–1190, Jul. 2014.

[27] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3424–3431.

[28] D. Zhang, F. Wang, and L. Si, "Composite hashing with multiple information sources," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2011, pp. 225–234.

[29] X. Zhu, Z. Huang, H. T. Shen, and X. Zhao, "Linear cross-modal hashing for efficient multimedia search," in *Proc. ACM Multimedia Conf.*, 2013, pp. 143–152.

[30] X. He and P. Niyogi, "Locality preserving projections," in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 153–160.

[31] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proc. Adv. Neural Inf. Process. Syst.*, 2006, pp. 801–808.

[32] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1548–1560, Aug. 2011.

[33] A. Cherian, "Nearest neighbors using compact sparse codes," in *Proc. 31th Int. Conf. Mach. Learn.*, 2014, pp. 1053–1061.

[34] R. Ye and X. Li, "Compact structure hashing via sparse and similarity preserving embedding," *IEEE Trans. Cybern.*, vol. 46, no. 3, pp. 718–729, Mar. 2016.

[35] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Trans. Inf. Theory*, vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[36] D. L. Donoho, "Compressed sensing," *IEEE Trans. Inf. Theory*, vol. 52, no. 4, pp. 1289–1306, Apr. 2006.

[37] X. Zhu, L. Zhang, and Z. Huang, "A sparse embedding and least variance encoding approach to hashing," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014.

[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1106–1114.

[39] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2475–2483.

[40] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3270–3278.

[41] 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.

[42] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 2064–2072.

[43] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell.*, Dec. 2016, pp. 1711–1717.

[44] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-UCSD birds-200-2011 dataset," California Inst. Technol., California, CA, USA, Tech. Rep. CNS-TR-2011-001, 2011.

[45] M. Belkin and P. Niyogi, "Using manifold stucture for partially labeled classification," in *Proc. Adv. Neural Inf. Process. Syst.*, 2002, pp. 929–936.

[46] B. Shen, B. Liu, Q. Wang, Y. Fang, and J. P. Allebach, "SP-SVM: Large margin classifier for data on multiple manifolds," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2965–2971.

[47] M. Belkin and P. Niyogi, "Laplacian eigenmaps for dimensionality reduction and data representation," *Neural Comput.*, vol. 15, no. 6, pp. 1373–1396, 2003.

[48] C. Ding, T. Li, and M. I. Jordan, "Convex and semi-nonnegative matrix factorizations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 1, pp. 45–55, Jan. 2010.

[49] Z. Wen and W. Yin, "A feasible method for optimization with orthogonality constraints," *Math. Program.*, vol. 142, no. 1, pp. 397–434, 2013.

[50] D. Goldfarb, Z. Wen, and W. Yin, "A curvilinear search method for p-harmonic flows on spheres," *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 84–109, 2009.

AQ:6 [51] J. Nocedal and S. Wright, *Numerical Optimization*. 1999.

[52] G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[53] D. Zhang, J. Wang, D. Cai, and J. Lu, "Self-taught hashing for fast similarity search," in *Proc. 33rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2010, pp. 18–25.

[54] A. Krizhevsky, "Learning multiple layers of features from tiny images," AQ:7 Univ. Toronto, Toronto, ON, Canada, Tech Rep., 2009.

[55] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[56] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: A real-world Web image database from national University of Singapore," in *Proc. 8th ACM Int. Conf. Image Video Retr.*, 2009, Art. no. 48.

[57] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[58] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[59] B. Xu, J. Bu, Y. Lin, C. Chen, X. He, and D. Cai, "Harmonious hashing," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 1820–1826.

[60] W. Kong and W. Li, "Isotropic hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1655–1663.

[61] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2074–2081.

Yuchen Guo received the B.Sc. degree from the School of Software, and the B.Ec. degree from the School of Economics and Management, Tsinghua University, Beijing, China, in 2013, where he is currently pursuing the Ph.D. degree with the School of Software. His research interests include multimedia information retrieval, computer vision, and machine learning. AQ:8

Guiguang Ding received the Ph.D. degree in electronic engineering from Xidian University, China, in 2014. In 2006, he has been a Post-Doctoral Research Fellow with the Department of Automation, Tsinghua University. He is currently an Associate Professor with the School of Software, Tsinghua University. He has authored 80 papers AQ:9 in major journals and conferences, including the IEEE TIP, TMM, TKDE, SIG IR, AAAI, ICML, IJCAI, CVPR, and ICCV. His current research centers on the area of multimedia information retrieval, computer vision and machine learning.

Li Liu received the Ph.D. degree from the Department of Electronic and Electrical Engineering, The University of Shefeld, Shefeld, U.K., in 2014. He is currently a Research Fellow with the Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, U.K.

Jungong Han was a Senior Scientist with Civolution Technology (a combining synergy of Philips CI and Thomson STS) from 2012 to 2015, a Research Staff with the Centre for Mathematics and Computer Science from 2010 to 2012, and a Researcher with the Technical University of Eindhoven, The Netherlands from 2005 to 2010. He is currently a Senior Lecturer with the Department of Computer Science, Northumbria University, U.K.

Ling Shao (M'09–SM'10) was a Professor with Northumbria University from 2014 to 2016, a Senior Lecturer with the University of Sheffield from 2009 to 2014, and a Senior Scientist with Philips Research, The Netherlands, from 2005 to 2009. He is currently a Professor with the School of Computing Sciences, University of East Anglia, Norwich, U.K. His research interests include computer vision, image/video processing, and machine learning. He is a fellow of the British Computer Society and the Institution of Engineering and Technology. He is an Associate Editor of the IEEE Transactions on Image Processing, the IEEE Transactions on Neural Networks and Learning Systems and several other journals.

AUTHOR QUERIES

AUTHOR PLEASE ANSWER ALL QUERIES

PLEASE NOTE: We cannot accept new source files as corrections for your paper. If possible, please annotate the PDF proof we have sent you with your corrections and upload it via the Author Gateway. Alternatively, you may send us your corrections in list format. You may also upload revised graphics via the Author Gateway.

AQ:1 = Please note that there were discrepancies between the accepted pdf [SHODE_R1.pdf] and the [bare_jrnl_final.tex] in Reference citations. We have followed [bare_jrnl_final.tex].
AQ:2 = Please confirm/give details of funding source.
AQ:3 = Please confirm whether the corresponding authors information is correct as set.
AQ:4 = Please note that there were discrepancies between the accepted pdf [SHODE_R1.pdf] and the [bare_jrnl_final.tex] in First Footnote. We have followed [bare_jrnl_final.tex].
AQ:5 = Table I is not cited in body text. Please indicate where it should be cited.
AQ:6 = Please provide the publisher name and location for ref. [51].
AQ:7 = Please provide the report no. for ref. [54].
AQ:8 = Please note that there were discrepancies between the accepted pdf [SHODE_R1.pdf] and the [bare_jrnl_final.tex] in biographies. We have followed [bare_jrnl_final.tex].
AQ:9 = Please confirm whether the edits made in the sentence “He has authored …” are OK.