Nonlinear Discrete Hashing

Zhixiang Chen, Jiwen Lu, *Senior Member, IEEE*, Jianjiang Feng, *Member, IEEE*, and Jie Zhou, *Senior Member, IEEE*

***Abstract*—In this paper, we propose a nonlinear discrete hashing approach to learn compact binary codes for scalable image search. Instead of seeking a single linear projection in most existing hashing methods, we pursue a multilayer network with nonlinear transformations to capture the local structure of data samples. Unlike most existing hashing methods that adopt an error-prone relaxation to learn the transformations, we directly solve the discrete optimization problem to eliminate the quantization error accumulation. Specifically, to leverage the similarity relationships between data samples and exploit the semantic affinities of manual labels, the binary codes are learned with the objective to: 1) minimize the quantization error between the original data samples and the learned binary codes; 2) preserve the similarity relationships in the learned binary codes; 3) maximize the information content with independent bits; and 4) maximize the accuracy of the predicted labels based on the binary codes. With an alternating optimization, the nonlinear transformation and the discrete quantization are jointly optimized in the hashing learning framework. Experimental results on four datasets including CIFAR10, MNIST, SUN397, and ILSVRC2012 demonstrate that the proposed approach is superior to several state-of-the-art hashing methods.**

***Index Terms*—Binary code, discrete optimization, hashing, multilayer neural network, nonlinear transformation.**

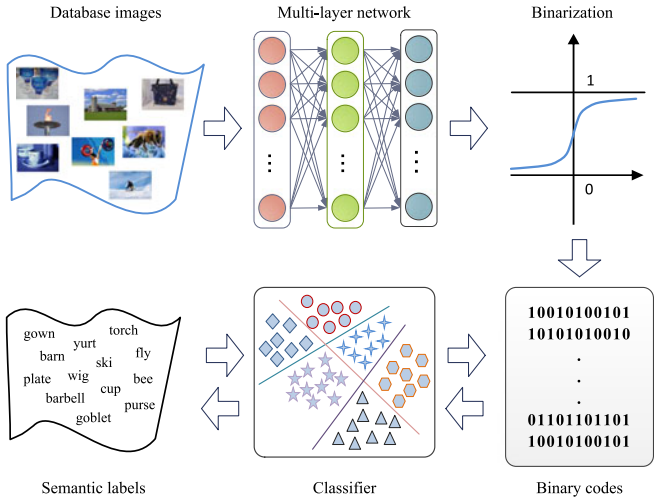

Fig. 1. Illustration of our proposed approach. To achieve short yet discriminative binary codes, a joint learning framework is proposed to: 1) preserve the nonlinear structure of database images with a multi-layer neural network; 2) minimize the quantization loss for better similarity preservation; 3) maximize the information content of binary codes with the independence constraint between different bits; and 4) exploit the semantic affinities of manual labels.

I. Introduction

HASHING approaches map high dimensional data samples into short binary descriptors to leverage the storage and the computing speed efficiencies of Hamming codes. Existing hashing based approaches can be broadly categorized as data-independent and data-dependent schemes. In data-independent techniques, projections are first randomly chosen to map data samples into a low dimensional space and then the results are binarized to obtain binary codes [1]–[8]. To take advantage of the distribution of data points, recent research attentions have been shifted to learning data-dependent binary codes by incorporating various machine learning techniques, which can be summarized as unsupervised [9]–[17], semi-supervised [18], and supervised [19]–[25] methods.

Manuscript received May 8, 2016; revised August 3, 2016; accepted October 17, 2016. Date of publication October 24, 2016; date of current version December 14, 2016. This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1001001, in part by the National Natural Science Foundation of China under Grant 61225008, Grant 61672306, Grant 61572271, Grant 61527808, Grant 61373074, and Grant 61373090, in part by the National 1000 Young Talents Plan Program, in part by the National Basic Research Program of China under Grant 2014CB349304, in part by the Ministry of Education of China under Grant 20120002110033, and in part by the Tsinghua University Initiative Scientific Research Program. The guest editor coordinating the review of this manuscript and approving it for publication was Dr. Nan Cao. *(Corresponding author: Jiwen Lu.)*

The authors are with the Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: chen-zx10@mails.tsinghua.edu.cn; lujiwen@tsinghua.edu.cn; jfeng@tsinghua.edu.cn; jzhou@tsinghua.edu.cn).

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

Digital Object Identifier 10.1109/TMM.2016.2620604

Learning based hashing methods demonstrate promising results by successfully addressing the speed and storage challenges. Most existing hashing methods assume a linear transformation to project data points from the high dimensional space into a low dimensional space. But in many applications, the data is linearly inseparable and thus the nonlinear data structure cannot be captured by such single linear projection. On the other side, by reducing the dimensionality of the data through learned transformations, the variances of the data in different dimensions show differences and thus carry different amounts of information. Directly quantizing the transformed data to produce binary codes is bound to degrade the performance. To this end, it is important to design a hashing method that can capture the latent local structure of data samples and efficiently optimize the quantization to preserve the similarity relationships between data points.

This paper proposes a nonlinear discrete hashing (NDH) learning framework to learn compact binary codes efficiently that addresses the abovementioned problems. Fig. 1 illustrates the framework of the proposed approach, which learns binary codes to simultaneously preserve the similarity relationships between images and exploit the semantic affinities of labels. Specifically, a network with multiple layers of nonlinear transformations is employed to capture the local structure of data samples. To produce compact yet discriminative binary codes, we directly incorporate the quantization optimization into the objective and efficiently solve the optimization problem in the Hamming space to optimize bits of binary codes iteratively

through coordinate descending. In particular, to incorporate the knowledge of both the local data structure and the semantic affinities between manual labels, the network is trained under four constraints over its output: 1) the quantization error between the input features and the learned binary codes is minimized; 2) the similarity relationships between data points are preserved; 3) the bits of learned binary codes are maximally independent to each other; 4) the classification loss between the predicted labels and the ground truth semantic labels is minimized. Experimental results on four datasets, including two typical datasets and two large scale datasets, validate the efficacy of the proposed approach in comparison with several state-of-the-art learning based hashing methods.

II. Related Work

A. Feature Space Transformation

In most existing learning based hashing methods, a single linear transformation is learned to encode input data samples with different objectives. Mapping similar items to close binary codes is achieved through a single projection matrix [11], [14], [15], [17]. In contrast to the searching metric distance neighbor in unsupervised hashing methods, semi-supervised and supervised hashing methods aim to retrieve semantically similar neighbor by incorporating pairwise supervised information for performance enhancement. In addition, compact bilinear projections and sparse projections are introduced in [9], [12], [26] to speed up the code generation and save memory cost. However, similar as most abovementioned approaches, the pursued linear projection cannot well capture the local neighbor structure of data samples.

To exploit the local structure of data samples, some recently proposed hashing approaches go beyond the linear transformation. For example, a kernelized supervised hashing [21], [27] employs a kernel formulation for the target hash functions. Following the work in [28], a graph-based hashing method is proposed to capture semantic neighborhoods [29]. Manifold inductive method in [30] proposes to learn binary embeddings on their intrinsic manifolds. While these approaches have shown performance enhancement over the linear ones, they suffer from the problems of scalability and out-of-sample extension due to the non-parametric formulation.

B. Quantization Optimization

Since the discrete constraint imposed by binary codes makes the optimization a challenging problem to be solved, learning based hashing usually deals with a relaxed continuous problem and quantizing the resulting continuous solution to generate binary codes [14], [18], [28], [29]. However, such methods may lead to low quality solution as suggested in [24], [31] due to the accumulation of quantization error.

To deal with the accumulation of quantization error, a rotation matrix is introduced in [10] to balance the variances of different PCA directions. Binary reconstructive embedding [22] realizes the poor optima under continuous relaxation and considers the alternating method to iteratively optimize one entry or one row of the projection matrix. Similarly, fast supervised hashing based on boosted decision trees [23] also iteratively optimizes a subset of the binary codes in an alternating optimizing scheme. Discrete graph hashing [31] preserves the neighbor structure of massive data in a discrete optimization framework with a graph-based hashing model, which suffers from the scalability problem. Supervised discrete hashing [24] introduces an auxiliary variable to reformulate the discrete optimization objective and adopts an existing kernel based hashing function. The abovementioned hashing approaches are either suboptimal due to the optimization on the fixed codes or subjected to the scalability problem.

III. Approach

In this section, we describe our proposed hashing method, starting with the nonlinear hashing functions to exploit the local structure of data samples. We then present the objective function to be minimized. Also, we develop an alternative optimizing method to solve the formulated problem directly in the Hamming space.

A. Nonlinear Hashing

While various hashing methods have been proposed in recent years, most of them aim to seek a single projection matrix [10], [18], [22], which is essentially linear and cannot preserve the nonlinear structure of data samples. Our goal is to learn a binary code for each data point by projecting the input data onto a binary code. To obtain the desirable binary codes, a good form of hash functions is important. To this end, in this paper, we consider the form of a multi-layer neural network to obtain the compact binary codes through multiple nonlinear transformations.

As shown in Fig. 1, a network with multiple layers of nonlinear transformations is constructed to compute the binary code $\boldsymbol{z}_i$ for each sample $\boldsymbol{x}_i$. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ denote the training set with n samples, where each sample $\boldsymbol{x}_i \in \mathbb{R}^d (1 \leq i \leq n)$ is a data point of d dimension. Given a data point $\boldsymbol{x}_i$, a r-bit binary code $\boldsymbol{z}_i \in \{+1, -1\}^r$ is generated by a set of hash functions $\boldsymbol{H}(\boldsymbol{x}) = [h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \ldots, h_r(\boldsymbol{x})]$. In a network with $M+1$ layers, assume the mth layer consists of $p^{(m)}$ units, where $1 \leq m \leq M$. Given a sample $\boldsymbol{x}_i \in \mathbb{R}^d$, the output of the first layer is computed as $\boldsymbol{h}^{(1)} = s(\boldsymbol{W}^{(1)}\boldsymbol{x}_i + \boldsymbol{c}^{(1)}) \in \mathbb{R}^{p^{(1)}}$, where $s(\cdot)$ is a nonlinear activation function, *e.g.*, the $\tanh$ or sigmoid function, and the projection matrix $\boldsymbol{W}^{(1)} \in \mathbb{R}^{p^{(1)}\times d}$ and the bias vector $\boldsymbol{c}^{(1)} \in \mathbb{R}^{p^{(1)}}$ are the parameters to be learned for the first layer of the network. The second layer has the output $\boldsymbol{h}^{(2)} = s(\boldsymbol{W}^{(2)}\boldsymbol{h}^{(1)} + \boldsymbol{c}^{(2)}) \in \mathbb{R}^{p^{(2)}}$ with $\boldsymbol{W}^{(2)} \in \mathbb{R}^{p^{(2)}\times p^{(1)}}$ and $\boldsymbol{c}^{(2)} \in \mathbb{R}^{p^{(2)}}$ as the projection matrix and the bias vector. Similarly, we can obtain the output of the mth layer with $\boldsymbol{h}^{(m)} = s(\boldsymbol{W}^{(m)}\boldsymbol{h}^{(m-1)} + \boldsymbol{c}^{(m)}) \in \mathbb{R}^{p^{(m)}}$. And, at the top layer, we can have the output of the network in the form of

$$\boldsymbol{F}(\boldsymbol{x}) = (\boldsymbol{h}^{(M)(\boldsymbol{x})})^T \in \mathbb{R}^{p^{(M)}} \tag{1}$$

where the mapping $\boldsymbol{F} : \mathbb{R}^d \mapsto \mathbb{R}^{p^{(M)}}$ is a parametric nonlinear function determined by $\{\boldsymbol{W}^{(m)}, \boldsymbol{c}^{(m)}\}_{m=1}^{M}$. Then, the sign of

the output of the top layer of the network is considered as the binary codes

$$\boldsymbol{z}_i = (\text{sgn}(\boldsymbol{h}^{(M)}(\boldsymbol{x}_i)))^T.$$

We put the binary vectors of all samples together as

$$\boldsymbol{B} = (\text{sgn}(\boldsymbol{h}^{(M)}))^T \tag{2}$$

where $\boldsymbol{B} \in \{-1, +1\}^{n\times r}$. On the base of (2), both the parameterized network and the set of binary codes of the training samples can be learned to exploit the nonlinear relationships between data samples. And the binary code of a new data point can be obtained by passing it through the learned network.

B. Objective Function

The objective is to find an $n \times r$ dimensional binary code matrix $\boldsymbol{B}$

$$\arg\min_{\boldsymbol{B}} \mathcal{Q} = \mathcal{Q}(\boldsymbol{L}, \boldsymbol{B}) + \mathcal{Q}(\boldsymbol{B}, \boldsymbol{X}) \tag{3}$$

where $\mathcal{Q}(\boldsymbol{L}, \boldsymbol{B})$ measures the difference between the ground truth labels and the predicted labels based on $\boldsymbol{B}$, whilst $\mathcal{Q}(\boldsymbol{B}, \boldsymbol{X})$ indicates the information loss caused by projecting the samples onto binary codes $\boldsymbol{B}$. This objective enables to exploit the semantic affinities of manual labels and preserve the similarity relationships between data samples simultaneously.

For the first term in (3), by denoting the classifier weight matrix as $\boldsymbol{P}$ and considering the quadratic loss function, the classification error has the following formula:

$$\mathcal{Q}_{\boldsymbol{P}}(\boldsymbol{L}, \boldsymbol{B}) = \left\|\boldsymbol{L} - \boldsymbol{P}\boldsymbol{B}^T\right\|_F^2 \tag{4}$$

where $\boldsymbol{L}$ is the labels of the samples and $\|\cdot\|_F^2$ denotes the Frobenius norm.

The second term in (3) measures the discrepancy between the data structures of the binary codes and the data samples. It consists of the quantization loss term

$$\mathcal{Q}_{\text{quan}} = \sum_{i=1}^{n} \left\|\boldsymbol{B}(i,:) - \boldsymbol{F}^{(M)}(i,:)\right\|_F^2 \tag{5}$$

and the similarity preserving term

$$\mathcal{Q}_{\text{sim}} = \sum_{i=1}^{n}\sum_{j=1}^{n} \boldsymbol{S}_{\text{ij}} \left\|\boldsymbol{F}^{(M)}(i,:) - \boldsymbol{F}^{(M)}(j,:)\right\|_F^2 \tag{6}$$

with $\boldsymbol{B}(i,:)$ being the binary code of sample $\boldsymbol{x}_i$ and $\boldsymbol{S}$ being the similarity matrix. Combining (5) and (6) together and writing them in the matrix form, we can reach

$$\begin{aligned} &\mathcal{Q}_{\boldsymbol{F}}(\boldsymbol{B}, \boldsymbol{X}) = \left\|\boldsymbol{B} - \boldsymbol{F}^{(M)}\right\|_F^2 + \alpha tr\big((\boldsymbol{F}^{(M)})^T \boldsymbol{D}\boldsymbol{F}^{(M)}\big) \\ &s.t. \quad \boldsymbol{B} \in \{-1, 1\}^{n\times r}, \boldsymbol{B}^T\boldsymbol{B} = n\boldsymbol{I}_r \end{aligned} \tag{7}$$

where $\boldsymbol{D} = Diag - \boldsymbol{S}$, $Diag$ is a diagonal matrix with $Diag_{i,i} = \sum_j \boldsymbol{S}_{i,j}$. The constraint $\boldsymbol{B}^T\boldsymbol{B} = n\boldsymbol{I}_r$ imposes column orthogonality on $\boldsymbol{B}$, which leads to the different bits of the binary vectors to be independent with each other and thus least information redundancy.

1) Bit-independent Constraint: Since concurrently imposing $\boldsymbol{B} \in \{-1, +1\}^{n\times r}$ and $\boldsymbol{B}^T\boldsymbol{B} = n\boldsymbol{I}_r$ will make hashing computationally intractable, most previous work adopts a relaxed form of the independent constraint. However, the relaxation [18], [28] may lead to poor codes due to the amplification of the error as the code length r increases. In contrast, we take into consideration the independent constraint without resorting to such error-prone relaxations. By defining a real-valued matrix set $\Omega = \{\boldsymbol{Y} \in \mathbb{R}^{n\times r} | \boldsymbol{Y}^T\boldsymbol{Y} = n\boldsymbol{I}_r\}$, the independent constraint is softened to minimizing the distance from matrix $\boldsymbol{B}$ to the set Ω, which is the minimal distance between matrix $\boldsymbol{B}$ and any matrix $\boldsymbol{Y} \in \Omega$ and is with the following formula:

$$\mathcal{Q}_I(\boldsymbol{B}) = \|\boldsymbol{B} - \boldsymbol{Y}\|_F^2 . \tag{8}$$

To substitute the independent constraint in objective function (7) with this softened one, the distance is scaled with a nonnegative factor. By imposing a very large factor, the distance is enforced to be zero and thus the distance minimization is turned back into the independent constraint. In practice, a certain discrepancy between $\boldsymbol{B}$ and Ω is allowed to make the optimization problem more flexible.

2) Companion Loss: The quantization loss function defined in (7) only considers the outputs of the top layer of the network, where the outputs of hidden layers are not exploited to train the network. In contrast to the conventional approach which only measures the output layer loss of the network and propagates the loss back to the hidden layers, we introduce the companion loss functions to provide an integrated direct measurement on each hidden layer. Specifically, the quantization loss function in (7) is reformulated as

$$\mathcal{Q}_{\boldsymbol{F}}(\boldsymbol{B}, \boldsymbol{X}) = \mathcal{Q}_{\boldsymbol{F}}^{(M)} + \sum_{m=1}^{M-1} \alpha^{(m)} h\big(\mathcal{Q}_{\boldsymbol{F}}^{(m)} - \tau^{(m)}\big) \tag{9}$$

where

$$\mathcal{Q}_{\boldsymbol{F}}^{(M)} = \left\|\boldsymbol{B} - \boldsymbol{F}^{(M)}\right\|_F^2 + \alpha^{(M)} tr\big((\boldsymbol{F}^{(M)})^T \boldsymbol{D}\boldsymbol{F}^{(M)}\big)$$

is referred to as the primary loss and

$$\mathcal{Q}_{\boldsymbol{F}}^{(m)} = tr\big((\boldsymbol{F}^{(m)})^T \boldsymbol{D}\boldsymbol{F}^{(m)}\big), m = 1, 2, \ldots, M-1$$

are referred to as the companion losses. These companion loss functions can be seen as additional constraints within the learning process, which evaluate the qualities of those hidden layer mappings. Moreover, by introducing the companion loss, an additional feedback is brought in for each hidden layer, which can be viewed as a new regularization [32], [33]. Concerning that the performance of the overall network might be interfered by the direct pursuit of locality structure preserving at all hidden layers, the impact of the companion loss is weaken during training to ensure that the overall goal of learning good structure preserving binary codes at the output layer is not fundamentally altered. The method by which we pursue this companion loss weakening is to employ the hinge loss function, which is defined as: $h(x) = max(x, 0)$. The positive threshold $\tau^{(m)}$ controls the loss $Q^{(m)}$ to appear or not in the learning procedure. The second term in (9) will disappear during the learning procedure if the overall loss of the mth hidden layer is below the threshold $\tau^{(m)}$.

The loss of the mth hidden layer will not impact the course of learning if its value is lower than the threshold $\tau^{(m)}$ in (9). The mth balance parameter $\alpha^{(m)}$ balances the effects of the primary loss and the corresponding companion loss.

On the basis of the semantic exploition in (4) and the similarity preservation in (8) and (9), the overall objective function in (3) has the specific form of

$$\begin{aligned}\underset{\boldsymbol{B},\boldsymbol{P},\{\boldsymbol{F}^{(m)}\}_{m=1}^{M},\boldsymbol{Y}}{\arg\min}\quad & \mathcal{Q}=\mathcal{Q}_{\boldsymbol{P}}+\lambda_1\mathcal{Q}_I+\lambda_2\mathcal{Q}_{\boldsymbol{F}}+\lambda_3\mathcal{Q}_R\\ s.t.\quad & \boldsymbol{B}\in\{-1,1\}^{n\times r}\end{aligned}\tag{10}$$

with $\mathcal{Q}_{\boldsymbol{P}}$, $\mathcal{Q}_{\boldsymbol{I}}$, $\mathcal{Q}_{\boldsymbol{F}}$ defined in (4), (8), (9), respectively, and

$$\mathcal{Q}_R=\|\boldsymbol{P}\|_F^2+\sum_{m=1}^{M}\left\|\boldsymbol{W}^{(m)}\right\|_F^2+\sum_{m=1}^{M}\left\|\boldsymbol{c}^{(m)}\right\|_F^2$$

contains the regularizers to control the scales of the parameters. In (10), the first three terms represent the classification error, column independent constraint on learned binary matrix, and the measurement on the trained network with quantization loss and similarity preservation, respectively. λ_1, λ_2, and λ_3 are the parameters representing the trade-off among the effects of different terms.

C. Optimization

To solve the optimization problem described in (10), we decompose the problem into four simpler sub-problems that are optimized in an alternative procedure. This results in the following four steps.

$\boldsymbol{P}$-step: By fixing all variables but $\boldsymbol{P}$, the objective in (10) is equivalently transformed to

$$\arg\min_{\boldsymbol{P}}\mathcal{Q}=\left\|\boldsymbol{L}-\boldsymbol{P}\boldsymbol{B}^T\right\|_F^2+\lambda_3\|\boldsymbol{P}\|_F^2.\tag{11}$$

For this problem, with the given $\boldsymbol{B}$, it is easy to solve $\boldsymbol{P}$ by the regularized least squares problem, which has a closed-form solution

$$\boldsymbol{P}=\boldsymbol{L}\boldsymbol{B}(\boldsymbol{B}^T\boldsymbol{B}+\lambda_3\boldsymbol{I})^{-1}.\tag{12}$$

$\boldsymbol{F}$-step: This sub-problem has the following form:

$$\begin{aligned}\underset{\{\boldsymbol{W}^{(m)},\boldsymbol{c}^{(m)}\}_{m=1}^{M}}{\arg\min}\quad & \mathcal{Q}=\lambda_2\left(\mathcal{Q}_{\boldsymbol{F}}^{(M)}+\sum_{m=1}^{M-1}\alpha^{(m)}h\left(\mathcal{Q}_{\boldsymbol{F}}^{(m)}-\tau^{(m)}\right)\right)\\ & +\ \lambda_3\sum_{m=1}^{M}\left(\left\|\boldsymbol{W}^{(m)}\right\|_F^2+\left\|\boldsymbol{c}^{(m)}\right\|_F^2\right).\end{aligned}\tag{13}$$

This optimization problem can be solved by the stochastic gradient descent method, which learns the parameters $\{\boldsymbol{W}^{(m)},\boldsymbol{c}^{(m)}\}_{m=1}^{M}$. The gradients of the objective function in (13) with respect to the parameters $\boldsymbol{W}^{(M)}$ and $\boldsymbol{c}^{(M)}$ at the top layer are computed as follows:

$$\frac{\partial\mathcal{Q}}{\partial\boldsymbol{W}^{(M)}}=\lambda_2\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(M)}}{\partial\boldsymbol{W}^{(M)}}+2\lambda_3\boldsymbol{W}^{(M)}\tag{14}$$

$$\frac{\partial\mathcal{Q}}{\partial\boldsymbol{c}^{(M)}}=\lambda_2\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(M)}}{\partial\boldsymbol{c}^{(M)}}+2\lambda_3\boldsymbol{c}^{(M)}.\tag{15}$$

For the hidden layers with $m=1,2,\ldots,M-1$, the gradients are computed as follows:

$$\begin{aligned}\frac{\partial\mathcal{Q}}{\partial\boldsymbol{W}^{(m)}}=\ & \lambda_2\left(\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(M)}}{\partial\boldsymbol{W}^{(m)}}+\sum_{\ell=m}^{M-1}\alpha^{(\ell)}h'\left(\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}-\tau^{(\ell)}\right)\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}}{\partial\boldsymbol{W}^{(m)}}\right)\\ & +2\lambda_3\boldsymbol{W}^{(m)}\end{aligned}\tag{16}$$

$$\begin{aligned}\frac{\partial\mathcal{Q}}{\partial\boldsymbol{c}^{(m)}}=\ & \lambda_2\left(\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(M)}}{\partial\boldsymbol{c}^{(m)}}+\sum_{\ell=m}^{M-1}\alpha^{(\ell)}h'\left(\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}-\tau^{(\ell)}\right)\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}}{\partial\boldsymbol{c}^{(m)}}\right)\\ & +2\lambda_3\boldsymbol{c}^{(m)}\end{aligned}\tag{17}$$

where $h'(\cdot)$ is the derivative function of $h(\cdot)$ with the value at non-differentiable point $x=0$ as zero. For $1\le m\le\ell\le M$, we have

$$\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}}{\partial\boldsymbol{W}^{(m)}}=\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}}{\partial\boldsymbol{c}^{(m)}}\left(\boldsymbol{F}^{(m-1)}\right)^T.$$

For $1\le m=\ell<M$, the gradient of $\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}$ with respect to $\boldsymbol{c}^{(m)}$ is computed by

$$\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(m)}}{\partial\boldsymbol{c}^{(m)}}=\left((\boldsymbol{D}+\boldsymbol{D}^T)\boldsymbol{F}^{(m)}\right)\odot s'\left(\boldsymbol{Z}^{(m)}\right)$$

otherwise

$$\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(M)}}{\partial\boldsymbol{c}^{(M)}}=\left(\alpha^{(M)}\left(\boldsymbol{D}+\boldsymbol{D}^T\right)\boldsymbol{F}^{(M)}-\boldsymbol{B}+\boldsymbol{F}^{(M)}\right)\odot s'\left(\boldsymbol{Z}^{(M)}\right)$$

when $m=\ell=M$. And for $1\le m<\ell\le M$, the gradient could be obtained according to the following updating equation:

$$\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}}{\partial\boldsymbol{c}^{(m)}}=\left((\boldsymbol{W}^{(m+1)})^T\frac{\partial\mathcal{Q}_{\boldsymbol{F}}^{(\ell)}}{\partial\boldsymbol{c}^{(m+1)}}\right)\odot s'(\boldsymbol{Z}^{(m)}).$$

Here $\odot$ represents element-wise multiplication, $\boldsymbol{Z}^{(m)}=\boldsymbol{W}^{(m)}\boldsymbol{F}^{(m-1)}+\boldsymbol{c}^{(m)}$ is the inner state of each layer, and $s'(\cdot)$ is the derivative function of $s(\cdot)$.

Then, parameters are updated by using the following gradient descent algorithm until convergence:

$$\boldsymbol{W}^{(m)}=\boldsymbol{W}^{(m)}-\eta\frac{\partial\mathcal{Q}}{\partial\boldsymbol{W}^{(m)}}\tag{18}$$

$$\boldsymbol{c}^{(m)}=\boldsymbol{c}^{(m)}-\eta\frac{\partial\mathcal{Q}}{\partial\boldsymbol{c}^{(m)}}\tag{19}$$

where η is the learning rate. Algorithm 1 summarizes the detailed procedure of the network training.

$\boldsymbol{B}$-step: The $\boldsymbol{B}$-subproblem is formulated as

$$\begin{aligned}\arg\min_{\boldsymbol{B}}\mathcal{Q}=\ & \left\|\boldsymbol{L}-\boldsymbol{P}\boldsymbol{B}^T\right\|_F^2+\lambda_1\|\boldsymbol{B}-\boldsymbol{Y}\|_F^2\\ & +\ \lambda_2\left(\mathcal{Q}_{\boldsymbol{F}}^{(M)}+\sum_{m=1}^{M-1}\alpha^{(m)}h\left(\mathcal{Q}_{\boldsymbol{F}}^{(m)}-\tau^{(m)}\right)\right)\\ s.t.\quad & \boldsymbol{B}\in\{-1,1\}^{n\times r}.\end{aligned}\tag{20}$$

Algorithm 1: Network training.

Input: Training set $\boldsymbol{X}$, network layer number M, learning rate η, iterative number R, and parameters λ_2, λ_3.
Output: Network parameters $\{\boldsymbol{W}^{(m)}, \boldsymbol{c}^{(m)}\}_{m=1}^{M}$.
for $r = 1, 2, \ldots, R$ **do**
  for $m = 1, 2, \ldots, M$ **do**
    Compute $\boldsymbol{F}^{(M)}(\boldsymbol{X})$ according to (1).
  end
  for $m = M, M-1, \ldots, 1$ **do**
    Calculate the gradients with (14-17).
  end
  for $m = 1, 2, \ldots, M$ **do**
    Update $\boldsymbol{W}^{(m)}$ and $\boldsymbol{c}^{(m)}$ according to (18,19).
  end
  Calculate $\mathcal{Q}$ using (13).
  If $r > 1$ and $|\mathcal{Q}_r - \mathcal{Q}_{r-1}| < \varepsilon$, go to **Return**.
end
Return: $\{\boldsymbol{W}^{(m)}, \boldsymbol{c}^{(m)}\}_{m=1}^{M}$.

With simple mathematical derivation, one can arrive at the equivalent formulation

$$\arg\min_{\boldsymbol{B}} \mathcal{Q} = tr(\boldsymbol{P}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{P}^T) - 2tr(\boldsymbol{B}^T\boldsymbol{U}) \tag{21}$$

where $\boldsymbol{U} = \boldsymbol{L}^T\boldsymbol{P} + \lambda_1\boldsymbol{Y} + \lambda_2\boldsymbol{F}^{(M)}$.

Since the binary constraint of $\boldsymbol{B} \in \{-1, +1\}^{n\times r}$ makes the objective discontinuous or indifferentiable, it is more challenging to solve this subproblem. Instead of the continuous relaxation, we consider fixing all but one column $\boldsymbol{b}_i$ of $\boldsymbol{B}$, and optimize the original objective with respect to this column $\boldsymbol{b}_i$. This optimization problem can provide an exact and optimal update to this column. Then, the binary code matrix is updated column by column in the cyclic coordinate descending scheme [22], [34].

Let $\boldsymbol{b}_i (0 \le i \le r)$ be the ith column of $\boldsymbol{B}$ and $\hat{\boldsymbol{B}}$ be the matrix of $\boldsymbol{B}$ excluding $\boldsymbol{b}_i$. Similarly, $\boldsymbol{p}_i$ and $\hat{\boldsymbol{P}}$ could be defined. We can have

$$tr(\boldsymbol{P}\boldsymbol{B}^T\boldsymbol{B}\boldsymbol{P}^T) = \text{const.} + 2\boldsymbol{p}_i^T\hat{\boldsymbol{P}}\hat{\boldsymbol{B}}^T\boldsymbol{b}_i.$$

Similarly, we can define $\boldsymbol{u}_i$ and derive that

$$tr(\boldsymbol{B}^T\boldsymbol{U}) = \text{const.} + \boldsymbol{u}_i^T\boldsymbol{b}_i.$$

With these derivations, the minimization problem with respect to the ith column of $\boldsymbol{B}$ is formulated as

$$\arg\min_{\boldsymbol{b}_i}(\boldsymbol{p}_i^T\hat{\boldsymbol{P}}\hat{\boldsymbol{B}}^T - \boldsymbol{u}_i^T)\boldsymbol{b}_i.$$

This problem has the closed-form solution

$$\boldsymbol{b}_i = \text{sgn}(\boldsymbol{u}_i - \hat{\boldsymbol{B}}\hat{\boldsymbol{P}}^T\boldsymbol{p}_i). \tag{22}$$

The optimal solution of (20) can be obtained by iteratively solving the following r sequential problems:

$$\arg\min_{\boldsymbol{b}_i}(\boldsymbol{p}_i^T\hat{\boldsymbol{P}}\hat{\boldsymbol{B}}^T - \boldsymbol{u}_i^T)\boldsymbol{b}_i, i = 1, 2, \ldots, r. \tag{23}$$

Algorithm 2: NDH.

Input: Training set $\boldsymbol{X}$, iterative number L, parameters λ_1,λ_2 and λ_3, and convergence error ε.
Output: Classification projection $\boldsymbol{P}$, and network parameters $\{\boldsymbol{W}^{(m)}, \boldsymbol{c}^{(m)}\}_{m=1}^{M}$.
Initialize $\boldsymbol{B}$, network parameters $\boldsymbol{W}, \boldsymbol{c}$
for $l = 1, 2, \ldots, L$ **do**
  Compute $\boldsymbol{P}$ according to (12).
  Train network according to Algorithm 1.
  Obtain the orthogonal matrix $\boldsymbol{Y}$ by (26).
  for $i = 1, 2, \ldots, r$ **do**
    Learn $\boldsymbol{B}$ with column by column update using the cyclic coordinate descending method with (22).
  end
  Calculate $\mathcal{Q}$ using (10).
  If $l > 1$ and $|\mathcal{Q}_l - \mathcal{Q}_{l-1}| < \varepsilon$, go to **Return**.
end
Return: $\boldsymbol{P}$ and $\{\boldsymbol{W}^{(m)}, \boldsymbol{c}^{(m)}\}_{m=1}^{M}$.

$\boldsymbol{Y}$-step: This has the following form:

$$\arg\min_{\boldsymbol{Y}} \mathcal{Q} = \lambda_1 \|\boldsymbol{B} - \boldsymbol{Y}\|_F^2. \tag{24}$$

This objective can be equivalently reformulated as

$$\arg\max_{\boldsymbol{Y}} \mathcal{Q} = \lambda_1 tr(\boldsymbol{B}^T\boldsymbol{Y}). \tag{25}$$

The solution of this subproblem can be achieved with the aid of the singular value decomposition (SVD) of $\boldsymbol{B}$

$$\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \sum_{k=1}^{r'}\sigma_k\boldsymbol{u}_k\boldsymbol{v}_k^T$$

where $r' \le r$ is the rank of $\boldsymbol{B}$, $\sigma_1, \ldots, \sigma_{r'}$ are the positive singular values, and $\boldsymbol{U} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{r'}]$ and $\boldsymbol{V} = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_{r'}]$ contain the left- and right-singular vectors, respectively. If $\boldsymbol{B}$ is full column rank, i.e. $r' = r$, then

$$\tilde{\boldsymbol{Y}} = \boldsymbol{U}\boldsymbol{V}^T$$

is the optimal solution to subproblem (25). Otherwise, the optimal solution could be obtained with the aid of Gram-Schmidt process. On the base of singular vectors in $\boldsymbol{U}$ and $\boldsymbol{V}$, it is easily to construct matrices $\hat{\boldsymbol{U}} \in \mathbb{R}^{n\times(r-r')}$ and $\hat{\boldsymbol{V}} \in \mathbb{R}^{r\times(r-r')}$ such that $\hat{\boldsymbol{U}}^T\hat{\boldsymbol{U}} = \boldsymbol{I}_{r-r'}$, $\boldsymbol{U}\hat{\boldsymbol{U}}^T = \boldsymbol{0}$, and $\hat{\boldsymbol{V}}^T\hat{\boldsymbol{V}} = \boldsymbol{I}_{r-r'}$, $\boldsymbol{V}\hat{\boldsymbol{V}}^T = \boldsymbol{0}$. Then, the matrix

$$\tilde{\boldsymbol{Y}} = [\boldsymbol{U}\hat{\boldsymbol{U}}][\boldsymbol{V}\hat{\boldsymbol{V}}]^T \tag{26}$$

is an optimal solution to subproblem (25). The full column rank condition can be seen as a special case of (26) by setting both $\hat{\boldsymbol{U}}$ and $\hat{\boldsymbol{V}}$ as $\boldsymbol{0}$.

Algorithm 2 summarizes our solution to the objective in Section III-B.

IV. Experiments

In this section, we conduct experiments on four datasets, namely CIFAR10 [35], MNIST [36], SUN397 [37], and

TABLE I
RESULTS ON THE CIFAR10 DATASET. THE SECOND TO FOURTH COLUMNS SHOW THE MEAN AVERAGE PRECISION RESULTS OF HAMMING RANKING PERFORMANCE. THE FIFTH TO SEVENTH COLUMNS SHOW THE AVERAGE PRECISION FOR THE TOP 500 RETURNED SAMPLES, WHILE THE LAST THREE COLUMNS SHOW THE RESULTS OF HAMMING LOOKUP PRECISION WITHIN RADIUS OF 2

Methods	Mean average precision(%)			Precision@500(%)			Precision@(radius==2)(%)		
	16	32	64	16	32	64	16	32	64
LSH [1]	12.63	13.70	14.62	15.32	17.23	19.36	16.67	6.35	0.1
SMLSH [8]	14.96	16.41	16.98	17.82	19.75	20.36	18.28	14.65	4.03
ITQ [10]	15.57	15.80	16.57	19.91	21.04	22.53	22.89	15.66	1.44
SPLH [18]	17.08	19.38	21.21	21.22	26.39	29.34	16.70	27.17	30.02
CCA-ITQ [10]	16.21	16.02	16.49	24.63	24.44	26.77	21.45	28.22	26.47
FastH [23]	27.94	33.09	36.55	37.74	43.13	46.84	37.76	34.42	11.64
SDH [24]	29.21	29.22	32.67	39.08	39.62	42.15	30.19	36.90	38.98
DeepH [25]	24.04	25.96	27.53	32.45	34.99	36.85	33.25	37.42	25.43
NDH	33.75	35.93	37.90	43.58	46.67	48.24	36.10	43.62	32.32

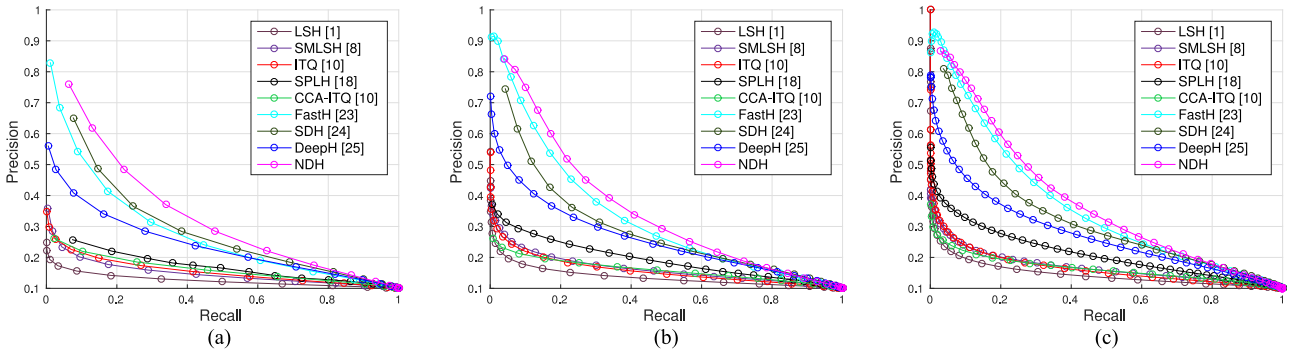

Fig. 2. Precision-recall curves on the CIFAR10 dataset for different binary code lengths. (a) 16 bits. (b) 32 bits. (c) 64 bits.

ILSVRC2012 [38], to evaluate the effectiveness of the proposed NDH approach by comparing with several state-of-the-art methods.

A. Datasets

1) CIFAR10: It is a set of 60000 manually labeled color images, which is sampled from the well-known 80M Tiny image benchmark [39]. They are from 10 classes, and each class has 6000 images. Each image is with the size of 32×32 and represented by a 512-D GIST feature vector [40]. Besides the hand-crafted GIST features, we also extract deep features with Convolutional Neural Networks (CNN) to evaluate different hashing methods. In particular, we follow the same settings of the CNN model in [41], which is pre-trained on ImageNet and finely tuned by the training set of CIFAR10, to extract a 1024 dimensional feature vector for each image.
2) MNIST: It is a handwritten digit dataset consisting of 70000 images with the size of 28×28. Each image is associated with a digit from '0' to '9' and represented as a 784-dimensional gray-scale feature vector by concatenating all pixels.
3) SUN397: This dataset contains 108 K images which are classified into 397 scene categories. Since images in this dataset are captured in more wild and uncontrolled conditions than the two above datasets, it is more challenging to retrieve semantic neighbors. Similarly, we also represent each image as a 4096-dimensional feature vector by using the CNN model in [41]. The model is pre-trained on the ImageNet [42] dataset and finely tuned with the training set of SUN397. To be specific, the output of the layer 'fc7' is extracted as the CNN feature as suggested in [43] so that the neuron activation values of internal layers of CNN are used as features.

TABLE II
COMPUTATION TIMES OF DIFFERENT HASHING METHODS ON THE CIFAR10 DATASET WHEN THE LENGTH OF HASH CODE IS SET TO BE 32 BITS

Methods	# of training samples	# of testing samples	Time	
			Train(s)	Test(ms)
LSH [1]	–	1000	–	22.4
SMLSH [8]	5000	1000	359.98	350.7
ITQ [10]	5000	1000	0.05	27.2
SPLH [18]	5000	1000	12.82	35.1
CCA-ITQ [10]	5000	1000	0.26	149.2
FastH [23]	5000	1000	74.65	5759.2
SDH [24]	5000	1000	1.33	1440.3
DeepH [25]	5000	1000	16.43	1181
NDH	5000	1000	25.23	937.7

TABLE III
COMPARISON OF RESULTS ON THE CIFAR10 DATASET WITH DIFFERENT NETWORK SETTINGS

# of layers	Mean average precision(%)			Precision@500(%)			Precision@(radius==2)(%)		
	16	32	64	16	32	64	16	32	64
Two	31.48	35.85	37.81	42.06	47.11	48.21	33.86	44.57	32.69
Three	33.75	36.59	38.25	43.58	46.89	48.50	36.10	43.74	31.75
Four	33.75	35.93	37.90	43.58	46.67	48.24	36.10	43.62	32.32
Five	32.46	35.85	37.93	43.10	46.33	48.23	35.61	44.22	32.67

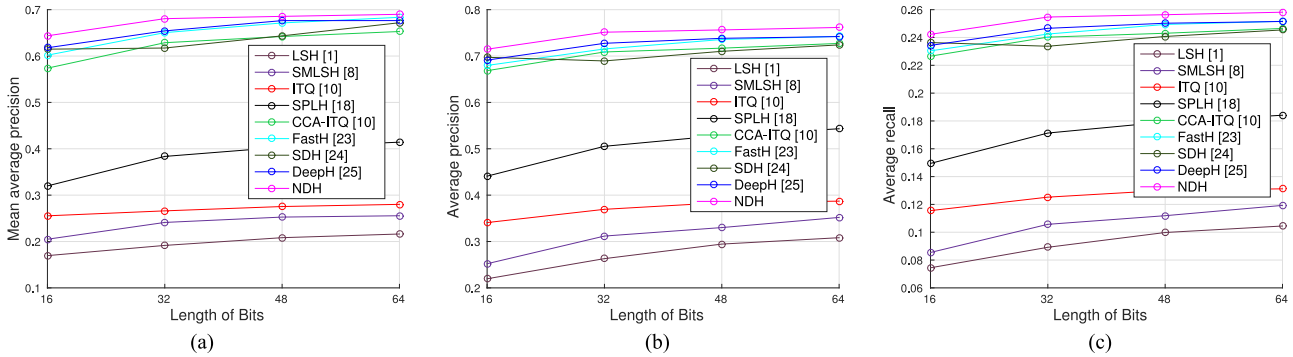

Fig. 3. Performance results of different hashing methods on the CIFAR10 dataset with CNN features. (a) Mean average precision. (b) Precision at top 2000. (c) Recall at top 2000.

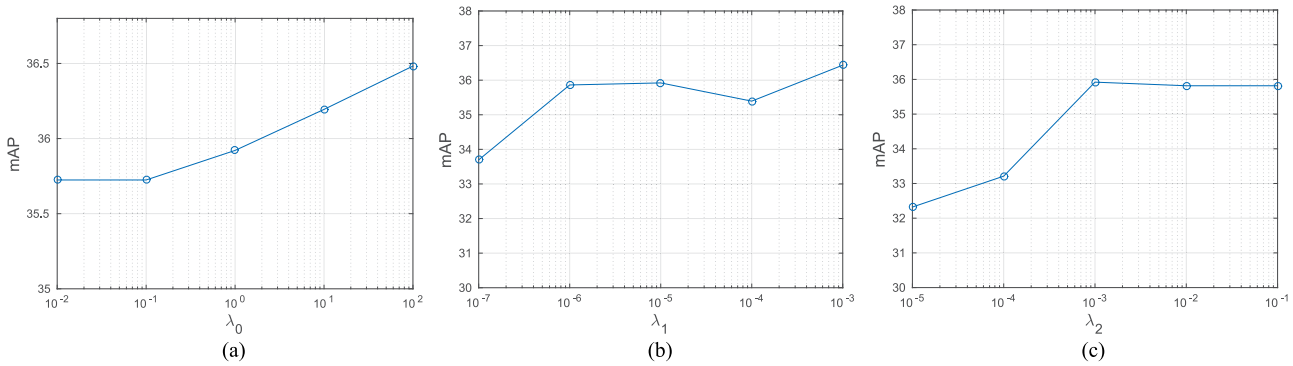

Fig. 4. mAP values with respect to the parameter variations on the CIFAR10 dataset with binary code length of 32 bits. (a) The impact of classification error. (b) The impact of bit independence. (c) The impact of nonlinear network.

4) ILSVRC2012: As a subset of ImageNet [42], this large dataset contains 1000 categories and 1.2 million images. Similarly, a 4096-dimensional feature vector is extracted for each image with the original CNN model provided in Caffe toolbox [44].

B. Evaluation Protocol

We use three metrics to measure the performance of different methods: mean average precision, precision at N samples, and Hamming lookup precision within a set Hamming radius r. The mean average precision presents an overall measurement of the retrieval performance by computing the area under the precision-recall curve, which delivers good discrimination and stability. Given top N returned samples, precision at N samples is calculated as the percentage of true neighbors. Hamming lookup precision measures the precision over all retrieved samples that fall into the buckets within a set Hamming radius, say $r = 2$, and a query with no returned point is treated as failed case. With the datasets fully annotated, the category information is used to define ground truth neighbors. All experiments are repeated 5 times and the averaged values are took as the final result.

C. Results on CIFAR10

Following the same setting in [18], we randomly sample 1000 images with 100 images per class to form the query set and use the remaining images as the database to test the hashing performance. And 5000 out of the 59000 database images are selected to construct the training set with 500 images per category. We normalize the data samples and compute the chi square distances to one thousand uniformly sampled points. The performance is

evaluated on binary codes with the lengths of 16, 32, and 64 bits.

The results of our NDH approach are compared with eight state-of-the-art hashing methods which cover both unsupervised method and supervised method. To be specific, LSH [1], SMLSH [8], ITQ [10], SPLH [18], CCA-ITQ [10], FastH [23], SDH [24], DeepH [25] are included. LSH and SMLSH are data independent hashing methods. While LSH is the classical hashing method, SMLSH is recently proposed to exploit more complex structure of data representation. ITQ and SPLH are the representatives of unsupervised and semi-supervised method, respectively. And the remaining ones are supervised methods, in which CCA-ITQ is the supervised extension of ITQ. While SDH and FastH are the recently proposed work to deal with the quantization optimization imposed by the binary constraint, the DeepH seeks a nonlinear transformation to learn compact binary codes. Due to the absence of public available implementation of SMLSH, we carefully reimplement it according to the description in [8] and set the number of feature scales as 3 and the size of the bit selection pool as 400% as suggested by the authors. For all the other compared methods, we use source codes kindly provided by the authors and adopt the suggested parameters of these methods from the corresponding authors. For our NDH method, we empirically set the network model as a 4-layer network ($M = 3$) with dimensions of [200 → 120 → 80 → 16], [200 → 120 → 80 → 32], and [200 → 120 → 80 → 64] for the bit length of 16, 32, and 64, respectively. The tanh function is taken as the nonlinear activation function in the network model. We initialize the biases $\{\boldsymbol{c}^{(m)}\}_{m=1}^{M}$ to be 0 and the weights $\{\boldsymbol{W}^{(m)}\}_{m=1}^{M}$ at each layer with the commonly used heuristic of uniform sampling [33]. Each element of $\boldsymbol{W}^{(m)}$ is uniformly sampled from the range $[-\frac{1}{\sqrt{col^{(m)}}}, \frac{1}{\sqrt{col^{(m)}}}]$, where $col^{(m)}$ is the number of columns of $\boldsymbol{W}^{(m)}$. The iteration numbers R and L are empirically set as 5 and 3. And we set coefficients $\alpha^{(1)}$ and $\alpha^{(2)}$ as 20, $\alpha^{(3)}$ as 100, the thresholds $\tau^{(1)}$ and $\tau^{(2)}$ as 1000, λ_1 as 1e-3, λ_2 as 1e-5, λ_3 as 1e-5 and learning rate η as 0.001. We tune the parameters through a 5 fold cross validation. At each run of the cross validation, one fifth of the training samples are selected as validation set. The same setting of our NDH method is adopted for all the other datasets.

1) Comparison With Baseline Methods: Table I shows the performance of different hashing methods in terms of mean average precision, top 500 precision, and Hamming lookup precision on the CIFAR10 dataset with respect to different length of binary code. We can see that our NDH method always outperforms all other methods by a large margin for the first two evaluation metrics and competitive to previous state-of-the-art methods. Besides our NDH method, no method could outperform the remaining competitors for any evaluation metric. We attribute this to the ability of local structure preservation by the multi-layer nonlinear transformation and discrete optimization with supervised information. We also notice that SMLSH shows performance improvement against LSH by data structure exploitation and supervised bit selection. Nevertheless, the last seven learning based hashing methods exhibit higher performance than those of data independent hashing methods, LSH and SMLSH. To get a deep observation of the mean average precision metric, the values of precision and recall for different Hamming distances are reported as the precision-recall curves in Fig. 2. As can be seen, our NDH method delivers higher performance than the remaining baseline methods.

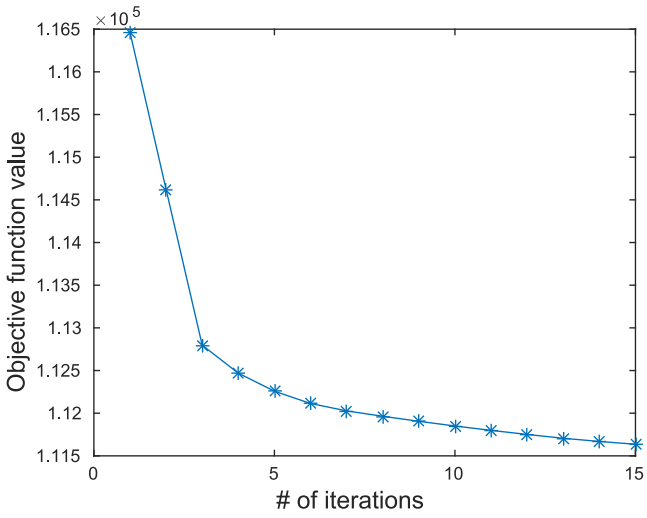

Fig. 5. Objective function value with respect to the number of iterations $\boldsymbol{L}$ of our NDH approach on the CIFAR10 dataset with binary code length of 32 bits.

Besides the performance, we also compare the time cost of our NDH method with those of other baseline methods. Both the training and testing times of all involved methods are reported in Table II for 32-bit binary codes on the CIFAR10 dataset. The computing platform is with 4.0 GHz Intel CPU and 32 GB RAM. From the table, we can see that both the training time and the testing time consumed by our NDH method are comparable to those of existing methods.

2) Effect of Different Layers: We evaluate the performance of our NDH method on network models with different numbers of layers. Specifically, we vary the network by setting the dimensions of the first M layers as [80], [120 → 80], [200 → 120 → 80], and [300 → 200 → 120 → 80] by setting M to 1-4, respectively. The results are summarized in Table III. We can observe that performance varies with different network settings and the best performance is achieved when the network model is with 2 and 3 layers rather than the largest 4 layers. This is interpreted as that a network model with 2 or 3 layer already have the ability to capture the local structure of the data samples. With larger layer number, the training process may lead to overfitting because there are more parameters to be trained.

3) Evaluation on Different Features: We also conduct experiments on the CIFAR10 dataset with the state-of-the-art deep CNN features. All experimental settings are the same as those of experiments with GIST features but with CNN features. Fig. 3 summarizes the performance of different hashing methods. Compared with the results with hand-crafted features, the performance of all hashing methods is substantially improved since the CNN features deliver stronger representations of the images. Nevertheless, our NDH method still show superior performance compared to baseline methods.

TABLE IV
RESULTS ON THE MNIST DATASET. THE THIRD TO FIFTH COLUMNS SHOW THE MEAN AVERAGE PRECISION RESULTS OF HAMMING RANKING PERFORMANCE. THE SIXTH TO EIGHTH COLUMNS SHOW THE AVERAGE PRECISION FOR THE TOP 500 RETURNED SAMPLES, WHILE THE LAST THREE COLUMNS SHOW THE RESULTS OF HAMMING LOOKUP PRECISION WITH RADIUS OF 2

Methods	# of training samples	Mean average precision(%)			Precision@500(%)			Precision@(radius==2)(%)		
		16	32	64	16	32	64	16	32	64
LSH [1]	–	18.51	25.41	32.78	28.08	38.56	48.39	33.66	39.69	3.8
SMLSH [8]	5000	31.68	38.28	43.42	41.93	49.16	55.14	46.36	53.37	13.41
ITQ [10]	5000	38.11	42.13	43.63	54.35	60.15	62.03	64.51	73.49	14.55
SPLH [18]	5000	48.67	49.38	48.71	59.69	60.57	63.06	61.23	73.87	48.67
CCA-ITQ [10]	5000	58.61	60.34	62.51	67.95	69.37	71.42	61.75	72.64	55.65
FastH [23]	5000	95.04	96.19	96.71	93.60	94.67	95.27	77.94	87.65	82.57
SDH [24]	5000	92.28	93.74	94.81	91.45	92.07	92.88	57.72	81.61	83.86
DeepH [25]	5000	70.91	74.10	76.34	76.75	79.13	81.55	72.72	80.08	69.82
NDH	5000	94.64	95.88	96.29	93.82	94.81	94.99	67.56	86.64	87.75

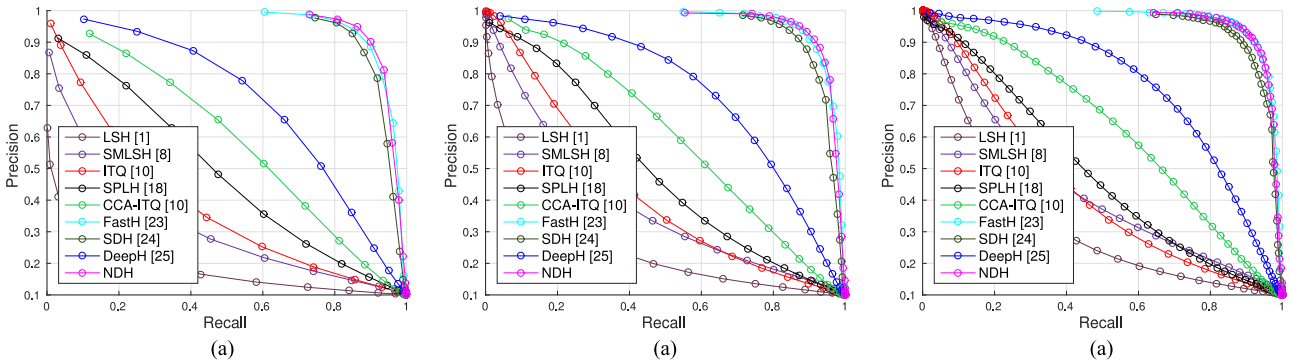

Fig. 6. Precision-recall curves on the MNIST dataset for different binary code lengths. (a) 16 bits. (b) 32 bits. (c) 64 bits.

TABLE V
RESULTS ON THE SUN397 DATASET. THE THIRD TO FIFTH COLUMNS SHOW THE MEAN AVERAGE PRECISION RESULTS OF HAMMING RANKING PERFORMANCE. THE SIXTH TO EIGHTH COLUMNS SHOW THE AVERAGE PRECISION FOR THE TOP 2000 RETURNED SAMPLES, WHILE THE LAST TWO COLUMNS SHOW THE RESULTS OF HAMMING LOOKUP PRECISION WITH RADIUS OF 2. HAMMING LOOKUP PRECISIONS ON 128 BIT BINARY CODES ARE NOT SHOWN BECAUSE IT IS IMPRACTICAL

Methods	# of training samples	Mean average precision(%)			Precision@2000(%)			Precision@(r==2)(%)	
		48	64	128	48	64	128	48	64
ITQ [10]	5000	5.16	5.58	6.73	6.14	6.43	6.98	2.03	0.29
SPLH [18]	5000	1.27	1.89	0.99	2.90	3.33	2.65	6.90	4.82
CCA-ITQ [10]	5000	7.22	6.38	6.08	6.21	5.90	5.56	3.37	1.31
FastH [23]	5000	2.71	4.98	8.28	2.90	3.90	5.22	0.11	0.14
SDH [24]	5000	9.87	9.65	11.85	7.57	7.81	8.52	11.71	8.00
DeepH [25]	5000	9.31	9.73	8.32	7.54	7.52	6.76	2.87	0.45
NDH	5000	11.39	12.96	13.86	7.81	8.32	9.05	8.81	6.23
ITQ [10]	20000	5.24	5.59	6.69	6.17	6.37	6.90	2.28	0.25
SPLH [18]	20000	4.57	2.81	1.79	4.94	3.04	2.93	1.05	5.67
CCA-ITQ [10]	20000	6.22	6.13	6.26	5.91	5.76	5.63	3.25	1.20
FastH [23]	20000	8.81	11.07	16.74	5.36	5.93	7.51	1.00	0.54
SDH [24]	20000	8.68	9.74	11.84	7.39	7.73	8.46	12.56	7.28
DeepH [25]	20000	12.41	14.35	14.20	9.34	9.59	9.50	6.45	1.73
NDH	20000	14.05	15.68	16.59	8.50	9.05	9.70	9.95	6.02

4) Parameter Analysis: We have also analyzed the impacts of parameter variations in the course of NDH training to see the sensitivity of the performance with respect to the parameters and the contributions of the three terms $\mathcal{Q}_{\boldsymbol{P}}$, $\mathcal{Q}_I$ and $\mathcal{Q}_{\boldsymbol{F}}$ in the objective function. By assuming a coefficient λ_0 (default as 1) for $\mathcal{Q}_{\boldsymbol{P}}$, we measure the performance with respect to λ_0, λ_1 and λ_2. Specifically, we measure the mAP values on the CIFAR10 dataset with the length of binary code as 32 bits, as shown in Fig. 4. Fig. 4(a) shows that increasing the proportion of the supervised classification error in the objective leads to performance improvement, which is achieved at the cost of overfitting and higher variation. Actually, the variation of λ_0 is equivalent to scaling the other parameters, λ_1, λ_2 and λ_3. As shown in Fig. 4(b), 4(c), the overall performance is improved

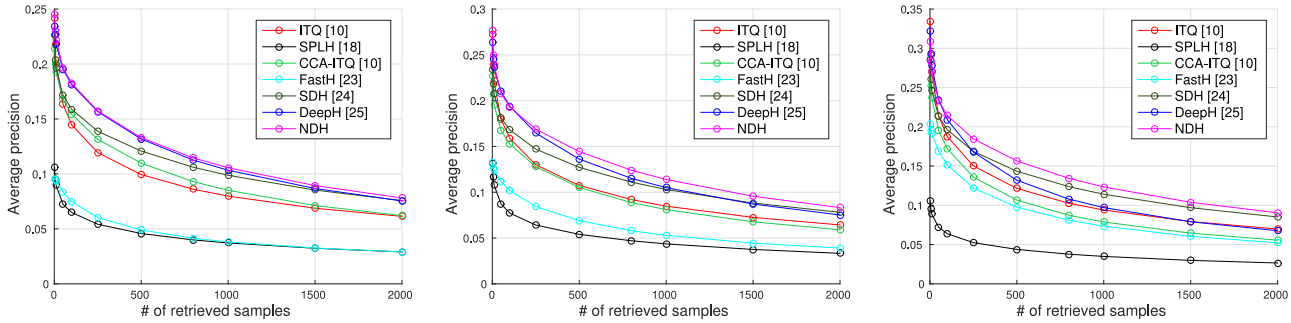

Fig. 7. Precision at top N retrieved samples on the SUN397 dataset with 5000 training samples. (a) 48 bits. (b) 64 bits. (c) 128 bits.

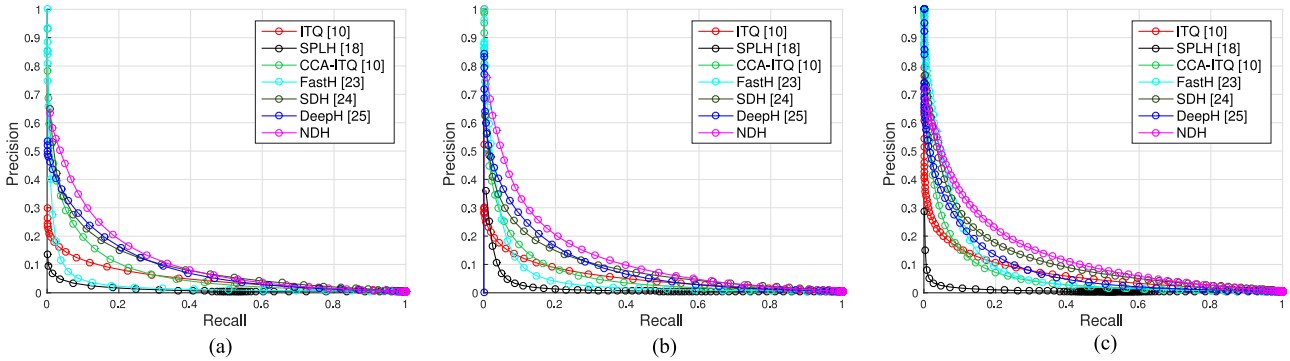

Fig. 8. Precision-recall curves on the SUN397 dataset with 5000 training samples for different binary code lengths. (a) 48 bits. (b) 64 bits. (c) 128 bits.

after introducing both the bit independence constraint and the nonlinear network.

5) Convergence Analysis: We have also analyzed the convergence of our NDH approach. As shown in Fig. 5, the objective in (10) converges quickly after several iterations of alternative update. This mainly benefits from the closed-form solution in the $\boldsymbol{B}$-step, which quickly passes on the feedback of the linear classifier $\boldsymbol{P}$ to the binary matrix $\boldsymbol{B}$. We notice that the objective function value changes slowly after 3 iterations.

D. Results on MNIST

Similar as the setting of experiments on the CIFAR10 dataset, we randomly set aside 100 images per class to form a query set of 1000 images and construct the database with the remaining images. The training set is with the size of 5000 images by selecting 500 images for each class. We adopt the same evaluation metrics on the CIFAR10 dataset to compare our NDH method with other baselines. Table IV reports the performance of different hashing methods on the MNIST dataset. From the reported performance on binary codes with 16, 32, and 64 bits lengths, we can lead to the observation that our NDH method rank top among all methods. This is explained by that the supervised information and data structure of this dataset are easier to be exploited and the recently proposed supervised hashing methods demonstrate the ability to realize the exploitation. Fig. 6 presents the precision-recall curves to give a detailed comparison among all methods. Compared to the results on the CIFAR10 dataset, all hashing methods deliver higher performance. We interpret this as the handwritten digit images in this dataset show simpler pattern than the objects in the CIFAR10 dataset. This leads to lower requirement on the hashing method and is in accord with the analysis above.

E. Results on SUN397

This dataset is more challenging than the above two datasets because this dataset is larger and with more categories and images in this dataset are captured under wild and uncontrolled conditions. Similar with the experimental settings above, 8000 images are randomly sampled as query images and the remaining images are left to form the database. Table V shows the performance of different hashing methods on the SUN397 dataset with different numbers of training samples. We notice that the overall performance is lower than those on the above CIFAR10 and MNIST datasets and our NDH method outperforms the best competitor in terms of the comprehensive mean average precision. We can also see that our NDH method presents superior performance for all evaluation metrics when using different numbers of training samples. The data independent hashing methods LSH and SMLSH are not evaluated because they are difficult to be used for such large dataset. Fig. 7 shows the precision of top N retrieved samples for each hashing method. The precision-recall curves with 5000 and 20000 training samples are demonstrated in Figs. 8 and 9, respectively. The curves in all above figures evidently show that our NDH method delivers the best performance.

F. Results on ILSVRC2012

ILSVRC2012 is a large and challenging dataset. Due to the absence of original test set, we follow the settings in [23] to substitute the query set with the validation set and use the provided

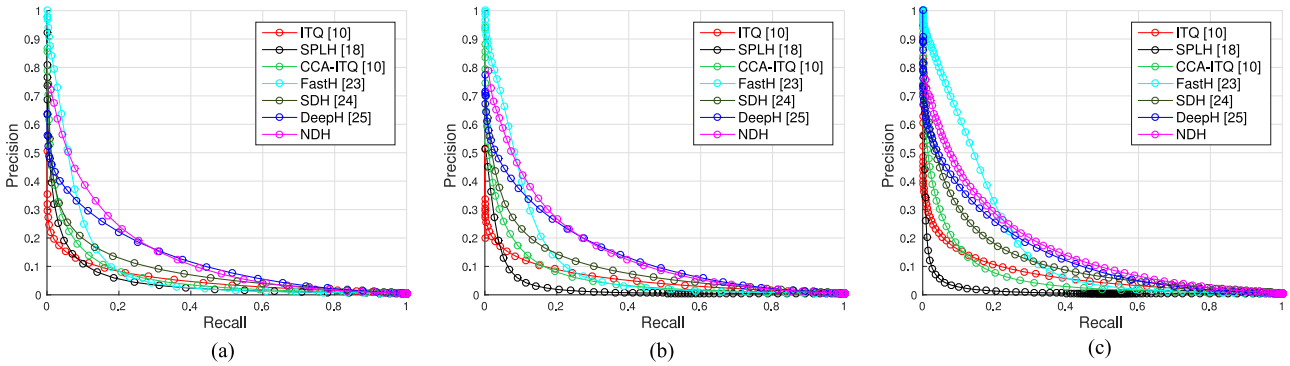

Fig. 9. Precision-recall curves on the SUN397 dataset with 20000 training samples for different binary code lengths. (a) 48 bits. (b) 64 bits. (c) 128 bits.

TABLE VI
RESULTS ON THE ILSVRC2012 DATASET. THE THIRD TO FIFTH COLUMNS SHOW THE MEAN AVERAGE PRECISION RESULTS OF HAMMING RANKING PERFORMANCE. THE SIXTH TO EIGHTH COLUMNS SHOW THE AVERAGE PRECISION FOR THE TOP 2000 RETURNED SAMPLES, WHILE THE LAST TWO COLUMNS SHOW THE RESULTS OF HAMMING LOOKUP PRECISION WITH RADIUS OF 2

Methods	# of training samples	Mean average precision(%)			Precision@2000(%)			Precision@(r==2)(%)	
		48	64	128	48	64	128	48	64
ITQ [10]	50000	3.92	4.99	7.03	8.70	10.15	12.77	9.84	3.47
SPLH [18]	50000	4.50	5.85	9.43	8.54	9.96	13.07	5.83	1.27
CCA-ITQ [10]	50000	4.45	5.94	10.97	9.44	11.12	15.32	11.57	8.69
FastH [23]	50000	0.55	0.93	2.98	1.79	3.64	4.71	0.26	0.03
SDH [24]	50000	7.03	8.30	10.85	10.14	11.69	14.30	12.20	7.59
DeepH [25]	50000	4.00	5.55	10.87	9.35	11.12	15.84	11.80	6.94
NDH	50000	7.31	8.66	11.04	10.69	12.17	14.46	13.24	8.26

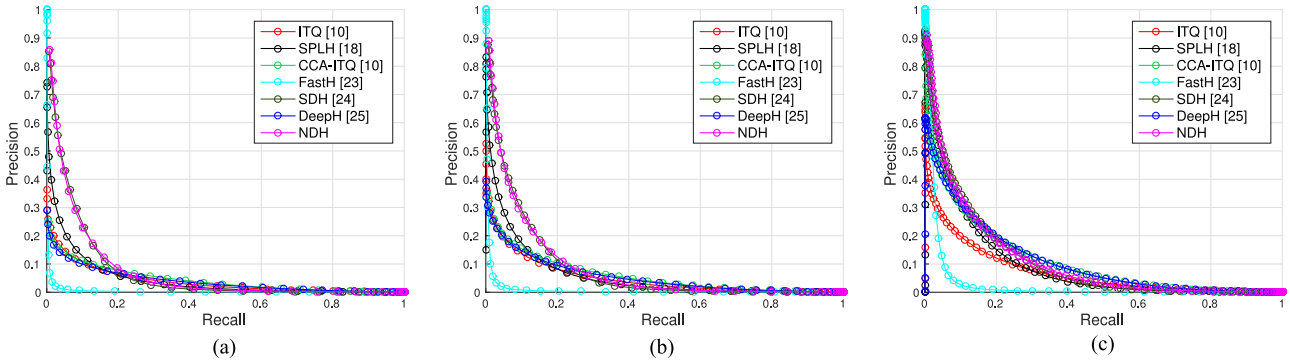

Fig. 10. Precision-recall curves on the ILSVRC2012 dataset with 50 000 training samples for different binary code lengths. (a) 48 bits. (b) 64 bits. (c) 128 bits.

training set as the database. Table VI summarizes the performance of different hashing methods, which indicates our NDH method is competitive to other methods. Since this dataset contains more than one million samples, the values are relatively low. Similar to that on the SUN397 dataset, we only show the results of leaning based hashing methods. Fig. 10 shows the precision-recall curves for 48, 64, and 128 bits binary codes, which demonstrate the superiority of our NDH method.

V. CONCLUSION

In this paper, we have proposed a nonlinear discrete hashing (NDH) approach to encode images as binary codes for scalable image retrieval. To preserve the local structure of data samples, the compact binary codes are learned through a network with multiple nonlinear transformations. To minimize the loss from the continuous results to the quantized codes, the discrete optimization problem is directly solved in the Hamming space to learn optimal binary codes. Extensive experimental results demonstrate the superiority of the proposed method.

REFERENCES

[1] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Commun. ACM*, vol. 51, no. 1, pp. 117–122, 2008.
[2] Q. Shi *et al.*, "Hash kernels for structured data," *J. Mach. Learn. Res.*, vol. 10, pp. 2615–2637, 2009.
[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proc. 20th Annu. Symp. Comput. Geom.*, 2004, pp. 253–262.
[4] M. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. 34th Annu. ACM Symp. Theory Comput.*, 2002, pp. 380–388.

[5] J. Ji *et al.*, "Batch-orthogonal locality-sensitive hashingfor angular similarity," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 10, pp. 1963–1974, Oct. 2014.
[6] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1509–1517.
[7] S. Kim and S. Choi, "Bilinear random projections for locality-sensitive binary codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 1338–1346.
[8] L. Weng *et al.*, "Supervised multi-scale locality sensitive hashing," in *Proc. 5th ACM Int. Conf. Multimedia Retrieval*, 2015, pp. 259–266.
[9] W. Liu, J. Wang, Y. Mu, S. Kumar, and S. Chang, "Compact hyperplane hashing with bilinear functions," in *Proc. 29th Int. Conf. Mach. Learn.*, 2012, pp. 17–24.
[10] 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.
[11] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 2938–2945.
[12] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 484–491.
[13] M. Á. Carreira-Perpiñán and R. Raziperchikolaei, "Hashing with binary autoencoders," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 557–566.
[14] R. Salakhutdinov and G. E. Hinton, "Semantic hashing," *Int. J. Approx. Reason.*, vol. 50, no. 7, pp. 969–978, 2009.
[15] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 353–360.
[16] G. Lin, C. Shen, and J. Wu, "Optimizing ranking measures for compact binary code learning," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 613–627.
[17] W. Kong and W. Li, "Double-bit quantization for hashing," in *Proc. 26th AAAI Conf. Artif. Intell.*, 2012, pp. 634–640.
[18] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.
[19] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, "LDAHash: Improved matching with smaller descriptors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 1, pp. 66–78, Jan. 2012.
[20] G. Lin, C. Shen, D. Suter, and A. van den Hengel, "A general two-step approach to learning-based hashing," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2013, pp. 2552–2559.
[21] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2012, pp. 2074–2081.
[22] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1042–1050.
[23] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2014, pp. 1971–1978.
[24] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 37–45.
[25] 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 Recog.*, Jun. 2015, pp. 2475–2483.
[26] M. Rastegari, C. Keskin, P. Kohli, and S. Izadi, "Computationally bounded retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2015, pp. 1501–1509.
[27] Y. Mu, J. Shen, and S. Yan, "Weakly-supervised hashing in kernel space," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3344–3351.
[28] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2008, pp. 1753–1760.
[29] W. Liu, J. Wang, S. Kumar, and S. Chang, "Hashing with graphs," in *Proc. 28th Int. Conf. Mach. Learn.*, 2011, pp. 1–8.
[30] F. Shen, C. Shen, Q. Shi, A. van den Hengel, and Z. Tang, "Inductive hashing on manifolds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2013, pp. 1562–1569.
[31] W. Liu, C. Mu, S. Kumar, and S. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3419–3427.
[32] C. Lee, S. Xie, P. W. Gallagher, Z. Zhang, and Z. Tu, "Deeply-supervised nets," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2015, pp. 562–570.
[33] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2010, pp. 249–256.
[34] B. Neyshabur, N. Srebro, R. Salakhutdinov, Y. Makarychev, and P. Yadollahpour, "The power of asymmetry in binary hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2823–2831.
[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Univ. Toronto, Toronto, ON, Canada, Tech. Rep. 001, 2009.
[36] L. Deng, "The MNIST database of handwritten digit images for machine learning research [best of the web]," *IEEE Signal Process. Mag.*, vol. 29, no. 6, pp. 141–142, Nov. 2012.
[37] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition from abbey to zoo," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2010, pp. 3485–3492.
[38] O. Russakovsky "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, 2015.
[39] A. Torralba, R. Fergus, and W. T. Freeman, "80 million tiny images: A large data set for nonparametric object and scene recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 11, pp. 1958–1970, Nov. 2008.
[40] 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.
[41] A. Vedaldi and K. Lenc, "MatConvNet: Convolutional neural networks for MATLAB," in *Proc. 23rd ACM Int. Conf. Multimedia*, 2015, pp. 689–692.
[42] J. Deng *et al.*, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, Jun. 2009, pp. 248–255.
[43] J. Donahue *et al.*, "DeCAF: A deep convolutional activation feature for generic visual recognition," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 647–655.
[44] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.

Zhixiang Chen received the B.S. degree in microelectronics from Xi'an Jiaotong University, Xi'an, China, in 2010, and is currently working toward the Ph.D. degree in the Department of Automation, Tsinghua University, Beijing, China.

His current research interests include large-scale visual search, binary embedding, and hashing learning.

Jiwen Lu (S'10–M'11–SM'15) received the B.Eng. degree in mechanical engineering and the M.Eng. degree in electrical engineering from Xi'an University of Technology, Xi'an, China, in 2003 and 2006, respectively, and the Ph.D. degree in electrical engineering from Nanyang Technological University, Singapore, in 2012.

From 2011 to 2015, he was a Research Scientist with the Advanced Digital Sciences Center, Singapore. He is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China. He has authored or coauthored more than 130 scientific papers, 33 of which were IEEE Transactions papers. His current research interests include computer vision, pattern recognition, and machine learning.

Prof. Lu is an Elected Member of the Information Forensics and Security Technical Committee of the IEEE Signal Processing Society. He serves as an Associate Editor of the *Pattern Recognition Letters, Neurocomputing,* and the IEEE ACCESS, a Managing Guest Editor of *Pattern Recognition and Image and Vision Computing*, a Guest Editor of *Computer Vision and Image Understanding* and *Neurocomputing*. He is the Workshop Chair, Special Session Chair, or Area Chair for more than ten international conferences. He was the recipient of the National 1000 Young Talents Plan Program Award in 2015.

Jianjiang Feng (M'10) received the B.S. and Ph.D. degrees from the School of Telecommunication Engineering, Beijing University of Posts and Telecommunications, Beijing, China, in 2000 and 2007, respectively.

From 2008 to 2009, he was a Postdoctoral Researcher with the Pattern Recognition and Image Processing Laboratory, Michigan State University, East Lansing, MI, USA. He is currently an Associate Professor with the Department of Automation, Tsinghua University, Beijing, China. His current research interests include fingerprint recognition and computer vision.

Prof. Feng is an Associate Editor of *Image and Vision Computing*.

Jie Zhou (M'01–SM'04) received the B.S. and M.S. degrees from the Department of Mathematics, Nankai University, Tianjin, China, in 1990 and 1992, respectively, and the Ph.D. degree from the Institute of Pattern Recognition and Artificial Intelligence, Huazhong University of Science and Technology, Wuhan, China, in 1995.

From 1995 to 1997, he was a Postdoctoral Fellow with the Department of Automation, Tsinghua University, Beijing, China. Since 2003, he has been a Full Professor with the Department of Automation, Tsinghua University. In recent years, he has authored more than 100 papers in peer-reviewed journals and conferences. Among them, more than 40 papers have been published in top journals and conferences, such as the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, the IEEE TRANSACTIONS ON IMAGE PROCESSING, and the IEEE Conference on Computer Vision and Pattern Recognition. His current research interests include computer vision, pattern recognition, and image processing.

Prof. Zhou is an Associate Editor of the *International Journal of Robotics and Automation* and two other journals. He was the recipient of the National Outstanding Youth Foundation of China Award.