Compact Hash Code Learning with Binary Deep Neural Network

Thanh-Toan Do, Tuan Hoang, Dang-Khoa Le Tan, Anh-Dzung Doan, Ngai-Man Cheung

***Abstract*—Learning compact binary codes for image retrieval problem using deep neural networks has recently attracted increasing attention. However, training deep hashing networks is challenging due to the binary constraints on the hash codes. In this paper, we propose deep network models and learning algorithms for learning binary hash codes given image representations under both unsupervised and supervised manners. The novelty of our network design is that we constrain one hidden layer to directly output the binary codes. This design has overcome a challenging problem in some previous works: optimizing non-smooth objective functions because of binarization. In addition, we propose to incorporate independence and balance properties in the direct and strict forms into the learning schemes. We also include a similarity preserving property in our objective functions. The resulting optimizations involving these binary, independence, and balance constraints are difficult to solve. To tackle this difficulty, we propose to learn the networks with alternating optimization and careful relaxation. Furthermore, by leveraging the powerful capacity of convolutional neural networks, we propose an end-to-end architecture that jointly learns to extract visual features and produce binary hash codes. Experimental results for the benchmark datasets show that the proposed methods compare favorably or outperform the state of the art.**

I. Introduction

Content-based image retrieval is an important and well studied problem in computer vision. It has many applications such as the visual search [1, 2, 3, 4, 5], place recognition [6, 7, 8], and camera pose estimation [9, 10, 11], to name a few. In the state-of-the-art image retrieval systems [1, 2, 3, 4, 5, 12], images are represented as high-dimensional feature vectors that can later be searched via the classical distance such as the Euclidean or Cosine distance. However, when the database is scaled up, there are two main requirements for retrieval systems, i.e., efficient storage and fast searching. Among solutions, binary hashing is an attractive approach for achieving those requirements [13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38]. Briefly, binary hashing learns a mapping (hashing) function that maps each original high dimensional vector $\mathbf{x} \in \mathbb{R}^D$ into a very compact binary vector $\mathbf{b} \in \{-1, 1\}^L$, where $L \ll D$. The distances between binary data points can be efficiently calculated by the bit operations, i.e., XOR and POPCOUNT. Furthermore, the binary representations also result in the sufficient storage.

The hashing methods can be divided into two groups, i.e., data-independent and data-dependent methods. The former ones [17, 18, 19, 20] rely on random projections to construct hash functions. The representative methods in this category are Locality Sensitive Hashing (LSH) [17] and its kernelized or discriminative extensions [18, 19]. The latter ones use the available training data to learn the hash functions in unsupervised [21, 22, 23, 24, 25, 26, 27, 28, 36, 37] or (semi-)supervised [29, 30, 31, 32, 33, 34, 35] manners. The representative unsupervised hashing methods, such as Spherical Hashing [24], Spectral Hashing [21], Iterative Quantization (ITQ) [22], and K-means Hashing [23], attempt to learn binary codes that preserve similar neighbors and the local structure of samples. The representative supervised hashing methods, such as, ITQ-CCA [22], Binary Reconstructive Embedding [34], Kernel Supervised Hashing [30], Two-step Hashing [33], and Supervised Discrete Hashing [35], attempt to learn binary codes that preserve the label similarity between samples. Extensive reviews of data-independent and data-dependent hashing methods can be found in recent surveys [13, 14, 15, 16].

One problem that makes the binary hashing difficult is the binary constraint on the codes, i.e., the outputs of the hash functions must be binary. Generally, this binary constraint leads to an NP-hard mixed-integer optimization problem. To overcome this difficulty, most of the aforementioned methods have relied on relaxation approaches that relax the constraint during the learning of hash functions. By using the relaxation approach, the continuous codes are first learned. Subsequently, the codes are binarized, for example, by thresholding. The relaxation significantly simplifies the original constrained binary problem. However, the solution can be suboptimal, i.e., the binary codes resulting from thresholded continuous codes could be inferior to those obtained by directly including the binary constraint in the learning.

In addition, as shown in the notable work *Spectral Hashing* [21], good binary codes should also have the following properties: (i) similarity preservation – (dis)similar inputs should likely have (dis)similar binary codes; (ii) independence – different bits in the binary codes are independent of each other so that no redundant information is captured; (iii) bit balance – each bit has a 50% chance of being 1 or -1. It is worth noting that direct incorporation of the independent and balance properties can complicate the learning. Previous works [22, 31] have used some relaxation or approximation to overcome the difficulties, but there may be some performance degradation.

Thanh-Toan Do is with the University of Liverpool, UK. Email: thanh-toan.do@liverpool.ac.uk.

Tuan Hoang, Dang-Khoa Le Tan, Ngai-Man Cheung are with Singapore University of Technology and Design, Singapore. Email: nguyenanhtuan_hoang@mymail.sutd.edu.sg, {letandang_khoa, ngaiman_cheung}@sutd.edu.sg.

Anh-Dzung Doan is with the University of Adelaide, Australia. Email: dung.doan@adelaide.edu.au.

Recently, deep learning has attracted great attention from the computer vision community due to its superiority in many vision tasks such as classification, detection, and segmentation [39, 40, 41]. Inspired by the success of deep learning in different vision tasks, recently, some researchers have used deep learning for joint learning image representations and binary hash codes in an end-to-end deep learning-based supervised hashing framework [42, 43, 44, 45]. However, learning binary codes in deep networks is challenging. This is because one has to deal with the binary constraint on the hash codes, i.e., one layer of the network should output binary codes. A naive solution is to adopt the $sign$ activation layer to produce binary codes. However, due to the lack of smoothness of the $sign$ function, it causes the *vanishing gradient* problem when training the network with standard back propagation [46].

Contributions: In this paper, first, we propose a novel deep network model and a learning algorithm for unsupervised hashing. Specifically, when learning binary codes, instead of involving the $sign$ or step function as in recent works [47, 48], our proposed network design constrains one layer to directly output the binary codes (therefore, the network is named as *Binary Deep Neural Network*). In addition, we propose to directly integrate the independence and balance properties into the objective function. Furthermore, we also include the similarity preserving property in our objective function. The resulting optimization with these binary and direct constraints is NP-hard. To overcome this challenge, we propose to attack the problem with alternating optimization and careful relaxation. Second, to increase the discriminative power of the binary codes, we extend our method to supervised hashing by leveraging the label information so that the binary codes preserve the semantic similarity between samples. Finally, to demonstrate the flexibility of our proposed method and to leverage the power of convolutional deep neural networks, we adapt our optimization strategy and the proposed supervised hashing model to an end-to-end deep hashing framework. Solid experiments on various benchmark datasets show the improvements of the proposed methods over state-of-the-art hashing methods.

A preliminary version of this work has been reported in [49]. In this work, we present a substantial extension to our previous work. In particular, the main extension is that we propose the end-to-end binary deep neural network framework which jointly learns the image features and binary codes. The experimental results show that the proposed end-to-end hashing framework significantly boosts the retrieval accuracy. Other extensions are more extensive experiments (e.g., new experiments on the SUN397 dataset [50], comparison to recent state-of-the-art end-to-end unsupervised and supervised hashing methods) to evaluate the effectiveness of the proposed methods.

The remainder of this paper is organized as follows. Section II presents related works. Section III presents and evaluates the proposed unsupervised hashing method. Section IV presents and evaluates the proposed supervised hashing method. Section V presents and evaluates the proposed end-to-end deep hashing network. Section VI concludes the paper.

II. Related work

In this section, we review related works that have also used neural networks as hash functions. We review both works that have used shallow network architectures [51, 47, 52, 48] and recent works which use end-to-end deep architectures [53, 42, 43, 44, 54, 55, 56].

In Semantic Hashing [51], the authors design a deep model by stacking Restricted Boltzmann Machines. That model does not consider the independence and balance of the codes. In Binary Autoencoder [48], the hash function is defined as a linear autoencoder. Because the Binary Autoencoder model [48] only uses one hidden layer, it may not well capture the input informations. We note that extension [48] with multiple, nonlinear layers is not easy due to the binary constraint. The model in [48] also does not consider the independence and balance of codes. In Deep Hashing [47, 52], the hash function is defined as a deep neural network. Nevertheless, the Deep Hashing model does not fully take into account the similarity preserving property. Furthermore, the authors also apply some relaxations in arriving at the independence and balance of codes. Those relaxations may degrade the performance. Recently, several works [53, 57, 58] have leveraged the Convolutional Neural Network (CNN) to learn more discriminative hash codes in the unsupervised manner. In DeepBit [53], the softmax layer of a pretrained network (i.e., VGG [59]) is replaced by a hash layer. Its loss function enforces several criteria on the codes produced by the hash layer, i.e., the output codes should: minimize the quantization loss; be distributed evenly; be invariant to rotation. The authors assume that the fully connected features produced by the pre-trained network are already sufficiently discriminative for image retrieval task. Hence, no similarity preserving criterion is considered on the hash codes. In Similarity-Adaptive Deep Hashing (SADH) [58], the authors propose to alternatively proceed over three training modules: deep hash model training, similarity graph updating and binary code optimization (with graph hashing [21]). In [57], the authors propose to analyze semantic informative deep features to obtain a semantic similarity matrix $\mathbf{S}$. The authors also relax the $sign$ function to the $tanh$ function to avoid the ill-posed gradient problem.

To handle the binary constraint, in Semantic Hashing [51], the authors first solves the relaxed problem by ignoring the binary constraints. Then, they apply thresholding on the continuous solution which results in binary codes. In Deep Hashing (DH) [47, 52], the binary codes are achieved by applying the $sign$ function on the outputs of the last layer, $\mathbf{H}^{(n)}$. The authors include a term in the objective function to reduce this binarization loss: $\left(sign(\mathbf{H}^{(n)}) - \mathbf{H}^{(n)}\right)$. However, due to the non-differentiability of the $sign$ function, solving the objective function of DH [47, 52] is difficult. The authors in [47, 52] assumed that the $sign$ function is differentiable everywhere, i.e., the derivative of $sign(x)$ equals zero for all values of x. In Binary Autoencoder (BA) [48], the binary codes are achieved by passing the outputs of the hidden layer into a step function. Incorporating the step function in the learning leads to a non-smooth objective function, i.e., a NP-complete problem. To handle this challenge, the authors [48] use binary

SVMs to learn the model parameters in the case when there is only a single hidden layer.

Joint learning image representations and binary hash codes in an end-to-end deep learning-based supervised hashing framework [42, 43, 44, 60, 55, 61, 54, 56] have shown a considerable boost in retrieval accuracy. By joint optimization, the produced hash codes are more sufficient to preserve the semantic similarity between images. In those works, the network architectures often consist of a feature extraction sub-network and a subsequent hashing layer to produce hash codes. Ideally, the hashing layer should adopt a $sign$ activation function to output exactly binary codes. However, due to the vanishing gradient difficulty of the $sign$ function, an approximation procedure must be employed. For example, $sign$ can be approximated by a tanh-like function $y = tanh(\beta x)$, where β is a free parameter controlling the trade off between the smoothness and the binary quantization loss [43]. However, it is non-trivial to determine an optimal β. A small β causes large binary quantization loss while a large β makes the output of the function close to the binary values, but the gradient of the function almost vanishes, making back-propagation infeasible. In [43], the β value is heuristically increased gradually reducing the smoothness as the training proceeds. Recently, similar to [43], HashNet [55] handles the non-smooth problem of the $sign$ function by continuation, i.e., starting the training with a smoothed objective function and gradually reducing the smoothness as the training proceeds. Furthermore, the above methods do not consider the independence and balance properties of the binary codes. The trade off problem between the smoothness and quatization loss persists when the logistic-like functions [42, 44] are used. In recent deep hashing works [61, 54], the absolute function and l_1 regularization are used to deal with the binary constraint on the codes. However, both absolute function and l_1 regularization are non-differentiable. The authors work around this difficulty by assuming that both are differentiable everywhere, but there may be some performance degradation. In Deep Pairwise-Supervised Hashing (DPSH) [56], the authors design a method to handle the binary constraint of the pairwise-supervised hashing objective function. Specifically, the outputs of the model are first computed, and then the corresponding binary codes are obtained by applying the $sign$ function to the outputs. By assuming the binary codes are fixed (to avoid the ill-posed gradient problem of the $sign$ function), the gradients are then computed to update the model weights. DPSH also does not consider the independence and balance properties, which are important for the hashing problem [21]. Recently, in [60] the authors proposed an binary encoder-decoder Recurrent Neural Network for video hashing. To handle the non-smooth problem of the *sign* function, the authors proposed to use the hinge loss to approximate the *sign* function. As will be discussed, our work proposes different formulations and new learning algorithms to deal with the binary constraints on the codes.

TABLE I
NOTATIONS AND THEIR CORRESPONDING MEANINGS.

Notation	Meaning
$\mathbf{X}$	$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m \in \mathbb{R}^{D\times m}$: set of m training samples; each column of $\mathbf{X}$ corresponds to one sample
$\mathbf{B}$	$\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^m \in \{-1,+1\}^{L\times m}$: binary code of $\mathbf{X}$
L	Number of required bits to encode a sample
n	Number of layers (including input and output layers)
s_l	Number of units in layer l
$f^{(l)}$	Activation function of layer l
$\mathbf{W}^{(l)}$	$\mathbf{W}^{(l)} \in \mathbb{R}^{s_{l+1}\times s_l}$: weight matrix connecting layer $l+1$ and layer l
$\mathbf{c}^{(l)}$	$\mathbf{c}^{(l)} \in \mathbb{R}^{s_{l+1}}$:bias vector for units in layer $l+1$
$\mathbf{H}^{(l)}$	$\mathbf{H}^{(l)} = f^{(l)}\left(\mathbf{W}^{(l-1)}\mathbf{H}^{(l-1)} + \mathbf{c}^{(l-1)}\mathbf{1}_{1\times m}\right)$: output values of layer l; convention: $\mathbf{H}^{(1)} = \mathbf{X}$
$\mathbf{1}_{a\times b}$	Matrix has a rows, b columns and all elements equal to 1

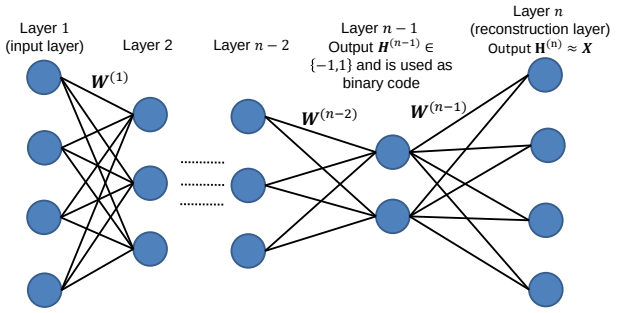


Fig. 1. The illustration of our UH-BDNN ($D = 4, L = 2$). In our proposed network design, the outputs of layer $n-1$ are constrained to $\{-1, 1\}$ and are used as the binary codes. During training, these codes are used to reconstruct the input samples at the final layer.

III. Unsupervised Hashing with Binary Deep Neural Network (UH-BDNN)

A. Formulation of UH-BDNN

For easy following, we first summarize the notations in Table I. In our work, the hash functions are defined by a deep neural network. In our proposed architecture, we use different activation functions in different layers. Specifically, we use the sigmoid function as the activation function for layers $2, \cdots, n-2$, and the identity function as the activation function for layer $n-1$ and layer n. Our idea is to learn the network such that the output values of the *penultimate layer* (layer $n-1$) can be used as the binary codes. We introduce constraints in the learning algorithm such that the output values at the layer $n-1$ have the following desirable properties: *(i)* belonging to $\{-1, 1\}$; *(ii)* similarity preserving; *(iii)* independence and *(iv)* balancing. Fig. 1 illustrates our network for the case $D = 4, L = 2$.

Let us start with the first two properties of the codes, i.e., belonging to $\{-1, 1\}$ and similarity preserving. To achieve binary codes having these two properties, we propose to optimize the following constrained objective function

$$\min_{\mathbf{W},\mathbf{c}} J = \frac{1}{2m}\left\|\mathbf{X} - \left(\mathbf{W}^{(n-1)}\mathbf{H}^{(n-1)} + \mathbf{c}^{(n-1)}\mathbf{1}_{1\times m}\right)\right\|^2 + \frac{\lambda_1}{2}\sum_{l=1}^{n-1}\left\|\mathbf{W}^{(l)}\right\|^2 \quad (1)$$

$$\text{s.t. } \mathbf{H}^{(n-1)} \in \{-1, 1\}^{L\times m} \quad (2)$$

Constraint (2) is to ensure the first property. As the activation function for the last layer is the identity function, the term $\left(\mathbf{W}^{(n-1)}\mathbf{H}^{(n-1)}+\mathbf{c}^{(n-1)}\mathbf{1}_{1\times m}\right)$ is the output of the last layer. The first term of (1) ensures that the binary code gives a good reconstruction of $\mathbf{X}$. It is worth noting that the reconstruction criterion has been used as an indirect approach for preserving the similarity in state-of-the-art unsupervised hashing methods [22, 48, 51], i.e., it encourages (dis)similar inputs to be mapped to (dis)similar binary codes. The second term is a regularization that tends to decrease the magnitude of the weights, which helps to prevent overfitting. Note that in our proposed design, we constrain the network to directly output the binary codes at one layer, which avoids the difficulty of the *sign* / *step* function which is non-differentiability. Our formulation with (1) under the binary constraint (2) is difficult to solve. It is a mixed-integer problem that is NP-hard. To address the problem, we propose to introduce an auxiliary variable $\mathbf{B}$ and use alternating optimization. Consequently, we reformulate the objective function (1) under constraint (2) as follows:

$$\begin{aligned}\min_{\mathbf{W},\mathbf{c},\mathbf{B}} J \quad = \quad & \frac{1}{2m}\left\|\mathbf{X}-\mathbf{W}^{(n-1)}\mathbf{B}-\mathbf{c}^{(n-1)}\mathbf{1}_{1\times m}\right\|^2 \\ & +\frac{\lambda_1}{2}\sum_{l=1}^{n-1}\left\|\mathbf{W}^{(l)}\right\|^2 \qquad (3)\\ \text{s.t. } & \mathbf{B}=\mathbf{H}^{(n-1)}, \qquad (4)\\ & \mathbf{B}\in\{-1,1\}^{L\times m}. \qquad (5)\end{aligned}$$

The advantage of introducing the auxiliary variable $\mathbf{B}$ is that the difficult constrained optimization problem (1) can be decomposed into two simpler sub-optimization problems. Consequently, we are able to iteratively solve the optimization problem by using alternating optimization with respect to $(\mathbf{W},\mathbf{c})$ and $\mathbf{B}$ while holding the other fixed. Inspired from the quadratic penalty method [62], we relax the equality constraint (4) by converting it into a penalty term. We achieve the following constrained objective function

$$\begin{aligned}\min_{\mathbf{W},\mathbf{c},\mathbf{B}} J \quad = \quad & \frac{1}{2m}\left\|\mathbf{X}-\mathbf{W}^{(n-1)}\mathbf{B}-\mathbf{c}^{(n-1)}\mathbf{1}_{1\times m}\right\|^2 \\ & +\frac{\lambda_1}{2}\sum_{l=1}^{n-1}\left\|\mathbf{W}^{(l)}\right\|^2+\frac{\lambda_2}{2m}\left\|\mathbf{H}^{(n-1)}-\mathbf{B}\right\|^2 \qquad (6)\\ \text{s.t. } & \mathbf{B}\in\{-1,1\}^{L\times m} \qquad (7)\end{aligned}$$

in which, the third term in (6) measures the (equality) constraint violation. By setting the penalty parameter λ_2 sufficiently large, we penalize the constraint violation severely, thereby forcing the minimizer of the penalty function (6) closer to the feasible region of the original constrained function (3).

Now let us consider the independence and balance properties of the codes. We note that the independence and balance can be constrained in $\mathbf{B}$. However, this makes the optimization on $\mathbf{B}$ difficult. Thus, for independence and balance, we constrain on $\mathbf{H}^{(n-1)}$. In contrast to previous works that use some relaxation or approximation on the independence and balance properties [22, 47, 31], in this work, we propose to encode these properties strictly and directly based on the outputs of the layer $n-1$. In particular, we encode the independence and balance properties of the codes by introducing the fourth and the fifth terms respectively in the following constrained objective function

$$\begin{aligned}\min_{\mathbf{W},\mathbf{c},\mathbf{B}} J \quad = \quad & \frac{1}{2m}\left\|\mathbf{X}-\mathbf{W}^{(n-1)}\mathbf{B}-\mathbf{c}^{(n-1)}\mathbf{1}_{1\times m}\right\|^2 \\ & +\frac{\lambda_1}{2}\sum_{l=1}^{n-1}\left\|\mathbf{W}^{(l)}\right\|^2+\frac{\lambda_2}{2m}\left\|\mathbf{H}^{(n-1)}-\mathbf{B}\right\|^2 \\ & +\frac{\lambda_3}{2}\left\|\frac{1}{m}\mathbf{H}^{(n-1)}(\mathbf{H}^{(n-1)})^T-\mathbf{I}\right\|^2+\frac{\lambda_4}{2m}\left\|\mathbf{H}^{(n-1)}\mathbf{1}_{m\times 1}\right\|^2 \qquad (8)\\ \text{s.t. } & \mathbf{B}\in\{-1,1\}^{L\times m}. \qquad (9)\end{aligned}$$

The objective function (8) under constraint (9) is our final formulation.

B. Optimization

To solve (8) under constraint (9), we propose to use alternating optimization over $(\mathbf{W},\mathbf{c})$ and $\mathbf{B}$.

1) $(\mathbf{W},\mathbf{c})$ *step:* When fixing $\mathbf{B}$, the problem becomes the unconstrained optimization. This is solved by using the *L-BFGS* [63] optimizer with backpropagation. The gradients of the objective function J (8) w.r.t. different parameters are computed as follows.

At $l=n-1$, we have

$$\frac{\partial J}{\partial \mathbf{W}^{(n-1)}}=\frac{-1}{m}(\mathbf{X}-\mathbf{W}^{(n-1)}\mathbf{B}-\mathbf{c}^{(n-1)}\mathbf{1}_{1\times m})\mathbf{B}^T+\lambda_1\mathbf{W}^{(n-1)} \qquad (10)$$

$$\frac{\partial J}{\partial \mathbf{c}^{(n-1)}}=\frac{-1}{m}\left((\mathbf{X}-\mathbf{W}^{(n-1)}\mathbf{B})\mathbf{1}_{m\times 1}-m\mathbf{c}^{(n-1)}\right) \qquad (11)$$

For other layers, let us define

$$\begin{aligned}\Delta^{(n-1)} \quad = \quad & \left[\frac{\lambda_2}{m}\left(\mathbf{H}^{(n-1)}-\mathbf{B}\right)\right. \\ & +\frac{2\lambda_3}{m}\left(\frac{1}{m}\mathbf{H}^{(n-1)}(\mathbf{H}^{(n-1)})^T-\mathbf{I}\right)\mathbf{H}^{(n-1)} \\ & \left.+\frac{\lambda_4}{m}\left(\mathbf{H}^{(n-1)}\mathbf{1}_{m\times m}\right)\right]\odot f^{(n-1)'}(\mathbf{Z}^{(n-1)}) \qquad (12)\end{aligned}$$

$$\Delta^{(l)}=\left((\mathbf{W}^{(l)})^T\Delta^{(l+1)}\right)\odot f^{(l)'}(\mathbf{Z}^{(l)}),\forall l=n-2,\cdots,2 \qquad (13)$$

where $\mathbf{Z}^{(l)}=\mathbf{W}^{(l-1)}\mathbf{H}^{(l-1)}+\mathbf{c}^{(l-1)}\mathbf{1}_{1\times m}$, $l=2,\cdots,n$; $\odot$ denotes the Hadamard product.

Then, $\forall l=n-2,\cdots,1$, we have

$$\frac{\partial J}{\partial \mathbf{W}^{(l)}}=\Delta^{(l+1)}(\mathbf{H}^{(l)})^T+\lambda_1\mathbf{W}^{(l)} \qquad (14)$$

$$\frac{\partial J}{\partial \mathbf{c}^{(l)}}=\Delta^{(l+1)}\mathbf{1}_{m\times 1} \qquad (15)$$

2) $\mathbf{B}$ *step:* When fixing $(\mathbf{W},\mathbf{c})$, we can rewrite problem (8) as

$$\begin{aligned}\min_{\mathbf{B}} J \quad = \quad & \left\|\mathbf{X}-\mathbf{W}^{(n-1)}\mathbf{B}-\mathbf{c}^{(n-1)}\mathbf{1}_{1\times m}\right\|^2 \\ & +\lambda_2\left\|\mathbf{H}^{(n-1)}-\mathbf{B}\right\|^2 \qquad (16)\\ \text{s.t. } & \mathbf{B}\in\{-1,1\}^{L\times m}. \qquad (17)\end{aligned}$$

We adaptively use the recent method *discrete cyclic coordinate descent* [35] to iteratively solve $\mathbf{B}$, i.e., row by row. The advantage of this method is that if we fix $L-1$ rows of $\mathbf{B}$ and only solve for the remaining row, we can achieve the closed-form solution for that row.

Algorithm 1 Unsupervised Hashing with Binary Deep Neural Network (UH-BDNN)

Input:
$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m \in \mathbb{R}^{D\times m}$: training data; L: code length; T: maximum iteration number; n: number of layers; $\{s_l\}_{l=2}^n$: number of units of layers $2 \to n$ (note: $s_{n-1} = L$, $s_n = D$); $\lambda_1, \lambda_2, \lambda_3, \lambda_4$.

Output:
Parameters $\{\mathbf{W}^{(l)}, \mathbf{c}^{(l)}\}_{l=1}^{n-1}$

1: Initialize $\mathbf{B}_{(0)} \in \{-1, 1\}^{L\times m}$ using ITQ [22]
2: Initialize $\{\mathbf{c}^{(l)}\}_{l=1}^{n-1} = \mathbf{0}_{s_{l+1}\times 1}$. Initialize $\{\mathbf{W}^{(l)}\}_{l=1}^{n-2}$ by getting the top s_{l+1} eigenvectors from the covariance matrix of $\mathbf{H}^{(l)}$. Initialize $\mathbf{W}^{(n-1)} = \mathbf{I}_{D\times L}$
3: Fix $\mathbf{B}_{(0)}$, compute $(\mathbf{W}, \mathbf{c})_{(0)}$ with $(\mathbf{W}, \mathbf{c})$ step using initialized $\{\mathbf{W}^{(l)}, \mathbf{c}^{(l)}\}_{l=1}^{n-1}$ (line 2) as starting point for L-BFGS.
4: **for** $t = 1 \to T$ **do**
5: Fix $(\mathbf{W}, \mathbf{c})_{(t-1)}$, compute $\mathbf{B}_{(t)}$ with $\mathbf{B}$ step
6: Fix $\mathbf{B}_{(t)}$, compute $(\mathbf{W}, \mathbf{c})_{(t)}$ with $(\mathbf{W}, \mathbf{c})$ step using $(\mathbf{W}, \mathbf{c})_{(t-1)}$ as starting point for L-BFGS.
7: **end for**
8: Return $(\mathbf{W}, \mathbf{c})_{(T)}$

Let $\mathbf{V} = \mathbf{X} - \mathbf{c}^{(n-1)}\mathbf{1}_{1\times m}$; $\mathbf{Q} = (\mathbf{W}^{(n-1)})^T\mathbf{V} + \lambda_2\mathbf{H}^{(n-1)}$. For $k = 1, \cdots L$, let $\mathbf{w}_k$ be k^{th} column of $\mathbf{W}^{(n-1)}$; $\mathbf{W}_1$ be matrix $\mathbf{W}^{(n-1)}$ excluding $\mathbf{w}_k$; $\mathbf{q}_k$ be k^{th} column of $\mathbf{Q}^T$; $\mathbf{b}_k^T$ be k^{th} row of $\mathbf{B}$; $\mathbf{B}_1$ be the matrix of $\mathbf{B}$ excluding $\mathbf{b}_k^T$. We have the closed-form for $\mathbf{b}_k^T$ as

$$\mathbf{b}_k^T = sign(\mathbf{q}^T - \mathbf{w}_k^T\mathbf{W}_1\mathbf{B}_1). \tag{18}$$

The proposed UH-BDNN method is summarized in Algorithm **1**. In Algorithm **1**, $\mathbf{B}_{(t)}$ and $(\mathbf{W}, \mathbf{c})_{(t)}$ are values of $\mathbf{B}$ and $\{\mathbf{W}^{(l)}, \mathbf{c}^{(l)}\}_{l=1}^{n-1}$ at iteration t, respectively.

C. Evaluation of Unsupervised Hashing with Binary Deep Neural Network (UH-BDNN)

In this section, we conduct experiments to evaluate the effectiveness of the proposed method UH-BDNN. We compare UH-BDNN with other unsupervised hashing methods: Binary Autoencoder (BA) [48], Spectral Hashing (SH) [21], Iterative Quantization (ITQ) [22], Spherical Hashing (SPH) [24], and K-means Hashing (KMH) [23], which are the state-of-the-art unsupervised hashing methods. For all compared methods, we use the released implementations and the suggested parameters provided by the authors.

1) Dataset, evaluation protocol, and implementation notes: We evaluate and compare methods on three benchmarking datasets: CIFAR10 [64], MNIST [65], and SIFT1M [66].

a) Dataset: The MNIST [65] dataset contains 70,000 handwritten digit images of 10 classes. We use the original split of the dataset. The training set (also used as the database for retrieval) consists of 60,000 images. The query set consists of 10,000 images. Each image is represented by a 784 dimensional feature vector by using its intensity in gray-scale.

The CIFAR10 [64] dataset contains 60,000 images of 10 classes. We use the original split of the dataset. The provided test set of 10,000 images is used as the query set. The remaining 50,000 images are used as the training set and the database for the retrieval. Each image is represented by an 800-dimensional feature vector extracted by PCA from the 4096-dimensional CNN feature produced by AlexNet [39].

TABLE II
THE EFFECTS OF OUR PROPOSED INDEPENDENCE (IND) AND BALANCE (BAL) TERMS ON RETRIEVAL PERFORMANCES (MAP). THE EXPERIMENTS ARE CONDUCTED ON CIFAR10 AND MNIST DATASETS.

	CIFAR10			MNIST		
Obj. function	16	24	32	16	24	32
With IDN+BAL	**5.80**	**9.11**	**11.97**	**10.57**	**18.32**	**24.90**
No BAL	5.44	9.02	11.60	10.42	17.99	24.84
No IND	5.13	8.89	11.28	9.91	17.85	24.75
No IND+BAL	4.89	8.57	10.93	9.81	17.55	24.57

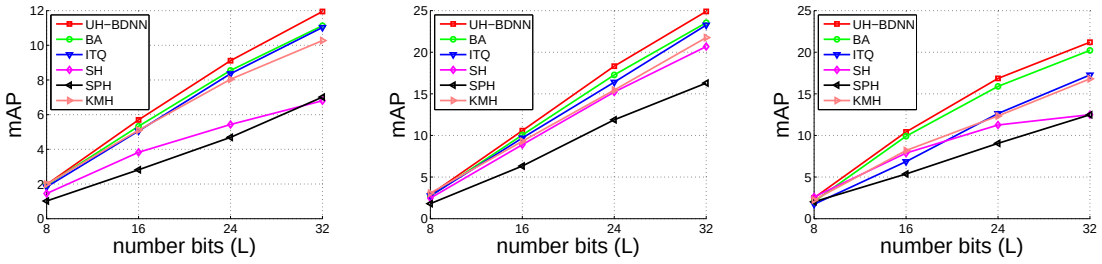
The SIFT1M [66] dataset is used to evaluate the proposed method on a large scale. The dataset contains 128 dimensional SIFT vectors [67]. The original split of this dataset consists of three separated sets of 1M, 100K and 10K vectors. These three sets correspond to the database, training, and query set respectively.

b) Evaluation protocol: In the state-of-the-art unsupervised hashing [22, 24, 23, 48], during the evaluation, instead of using the class labels, the Euclidean nearest neighbors are used as the ground truths for the queries. Hence, we follow this setting to evaluate the proposed method. Specifically, the number of ground truths are set as in [48]. For each query image of the CIFAR10 and MNIST datasets, we use its 50 Euclidean nearest neighbors as the ground truths. For the large scale dataset SIFT1M, we use 10,000 Euclidean nearest neighbors as the ground truths for each query. To evaluate the retrieval performance, we follow the state of the art [22, 48, 47] which use the following evaluation metrics: 1) mean Average Precision (mAP); 2) precision of Hamming radius 2 (precision@2) which measures precision on retrieved images with a Hamming distance to query ≤ 2 (note that we report zero precision in the case of no satisfactory image). Because computing mAP is slow for the large dataset SIFT1M, we consider the top 10,000 returned neighbors when computing mAP.

c) Implementation notes: In our deep model, we use $n = 5$ layers. More specifically, for the code lengths of 8, 16, 24 and 32 bits, the numbers of units in hidden layers $2, 3$, and 4 are empirically set as $[90 \to 20 \to 8]$, $[90 \to 30 \to 16]$, $[100 \to 40 \to 24]$ and $[120 \to 50 \to 32]$ respectively. Additionally, the parameters λ_1, λ_2, λ_3 and λ_4 are empirically set by cross validation as 10^{-5}, 5×10^{-2}, 10^{-2} and 10^{-6}, respectively. Finally, we empirically set the max iteration number T to 10.

2) Retrieval results:

a) Effects of independence and balance terms: First, we investigate the contributions of independence and balance terms in our proposed method. The quantitative results shown in Table II clearly confirm the importance of independence and balance terms. More specifically, when including the proposed independence and balance terms, we achieve improvement on the retrieval performance (mAP) (i.e., $> 0.5\%$ in the majority of experiments). Additionally, the experimental results also show that the independence term plays a more important role than the balance term, i.e., the performance drops are larger without the independence term than those without the balance term.



(a) CIFAR10 (b) MNIST (c) SIFT1M

Fig. 2. mAP comparison between UH-BDNN and state-of-the-art unsupervised hashing methods on CIFAR10, MNIST, and SIFT1M.

TABLE III
PRECISION AT HAMMING DISTANCE $r = 2$ COMPARISON BETWEEN UH-BDNN AND STATE-OF-THE-ART UNSUPERVISED HASHING METHODS ON CIFAR10, MNIST, AND SIFT1M.

	CIFAR10				MNIST				SIFT1M			
L	8	16	24	32	8	16	24	32	8	16	24	32
UH-BDNN	0.55	5.79	22.14	18.35	0.53	6.80	29.38	38.50	4.80	25.20	62.20	80.55
BA [48]	0.55	5.65	20.23	17.00	0.51	6.44	27.65	35.29	3.85	23.19	61.35	77.15
ITQ [22]	0.54	5.05	18.82	17.76	0.51	5.87	23.92	36.35	3.19	14.07	35.80	58.69
SH [21]	0.39	4.23	14.60	15.22	0.43	6.50	27.08	36.69	4.67	24.82	60.25	72.40
SPH [24]	0.43	3.45	13.47	13.67	0.44	5.02	22.24	30.80	4.25	20.98	47.09	66.42
KMH [23]	0.53	5.49	19.55	15.90	0.50	6.36	25.68	36.24	3.74	20.74	48.86	76.04

b) Comparison with the state of the art: The comparative performances in terms of mAP and the precision of Hamming radius 2 (precision@2) are shown in Fig. 2 and Table III, respectively. We find the following observations are consistent across all three datasets. In term of mAP, the proposed UH-BDNN is comparable to or outperforms other methods at all code lengths. At high code lengths, i.e., $L = 24, 32$, we observe clearer improvements. The best competitor for UH-BDNN is Binary Autoencoder (BA) [48]. In comparison to BA, at high code lengths, UH-BDNN consistently outperforms BA on all datasets. Regarding the precision@2, at $L = 8, 16$, UH-BDNN is comparable to other methods, while at $L = 24, 32$, UH-BDNN achieve considerably better performance than other methods. Specifically, the improvements of UH-BDNN over the best competitor BA [48] are clearly observed at $L = 32$ on the MNIST and SIFT1M datasets. These comparative results again confirm the advantages of our proposed method, i.e., directly enforcing the independence and balance properties on the binary outputs and carefully relaxing the binary constraint.

Comparison with Deep Hashing (DH) [47, 52]: Because the implementation of DH [47] has not been released, to make a fair comparison between UH-BDNN and DH, we configure the experiments on CIFAR10 and MNIST similar to [47]. Specifically, for each dataset, 1,000 images (i.e., 100 images per class) are randomly sampled as the query set; the remaining images are used as training and database sets. Similar to DH [47], GIST descriptors [68] are used to represent CIFAR10 images. Additionally, the class labels are used as the ground truths for the queries[1]. We present the comparative results in term of mAP and precision@2 in Table IV. The results show that the proposed UH-BDNN outperforms DH [47] at all compared code lengths, in both mAP and precision@2.

Comparison with DeepBit [53]: Here, we compare the proposed UH-BDNN with the recent end-to-end unsupervised hashing DeepBit [53]. As reported in [53], DeepBit uses the pre-trained VGG network [59] and fine-tunes the VGG using 50,000 training samples of CIFAR10. Because DeepBit is an unsupervised method, it does not use the data label during fine-tuning. The comparative results between DeepBit and other methods on the top 1,000 returned images (with the class labels ground truth) on the testing set of CIFAR10 is cited at the top part of Table V.

It is worth noting that in DeepBit [53], when reporting the results of ITQ, KMH, and SPH, the authors use GIST features for these methods. To make a fair comparison, we evaluate those three hashing methods on the features extracted from the activations of the last fully connected layer of the same pre-trained VGG [59] under the same setting. The results of those three methods, which are noted as ITQ-CNN, KMH-CNN, and SPH-CNN, are presented at the bottom part of Table V. The results show that using fully-connected features instead of GIST, ITQ-CNN, KMH-CNN, and SPH-CNN provides significant improvements. To evaluate the proposed UH-BDNN, we also use the same fully-connected features. The results of UH-BDNN with fully-connected features, which are noted as UH-BDNN-CNN, are presented in the last row of Table V. They show that with the same code length, UH-BDNN significantly outperforms the recent end-to-end unsupervised hashing DeepBit [53]. Furthermore, UH-BDNN also outperforms ITQ-CNN, KMH-CNN, and SPH-CNN with a fair margin.

[1]It is worth noting that in the evaluation of unsupervised hashing, instead of using the class label as ground truths, most state-of-the-art methods [22, 24, 23, 48] use Euclidean nearest neighbors as ground truths for the queries.

TABLE IV
COMPARISON WITH DEEP HASHING (DH) [47].

	CIFAR10				MNIST			
	mAP		precision@2		mAP		precision@2	
L	16	32	16	32	16	32	16	32
DH [47]	16.17	16.62	23.33	15.77	43.14	44.97	66.10	73.29
UH-BDNN	17.83	18.52	24.97	18.85	45.38	47.21	69.13	75.26

TABLE V
COMPARISON BETWEEN DEEPBIT [53] AND OTHER UNSUPERVISED HASHING METHODS ON CIFAR10. THE RESULTS IN THE FIRST FOUR ROWS ARE CITED FROM [53], WHICH WE HAVE ALSO REPRODUCED.

Method	16 bits	32 bits	64 bits
ITQ [22]	15.67	16.20	16.64
KMH [23]	13.59	13.93	14.46
SPH [24]	13.98	14.58	15.38
DeepBit [53]	19.43	24.86	27.73
ITQ-CNN	38.52	41.39	44.17
KMH-CNN	36.02	38.18	40.11
SPH-CNN	30.19	35.63	39.23
UH-BDNN-CNN	40.79	44.63	46.75

IV. SUPERVISED HASHING WITH BINARY DEEP NEURAL NETWORK (SH-BDNN)

One advantage of the proposed UH-BDNN is its flexibility. It can be extended to the supervised version when the label information for the data is available. In this section, to enhance the discriminative power of the binary codes, we extend UH-BDNN to supervised hashing by leveraging the label information.

To exploit the label information, we follow the approach proposed in Kernel-based Supervised Hashing (KSH) [30]. The advantage of this approach is that it directly encourages the Hamming distances between binary codes of within-class samples equal to 0, and the Hamming distances between binary codes of between-class samples equal to L. To achieve this goal, it enforces the Hamming distances between learned binary codes to be highly correlated with the pre-computed pairwise label matrix.

Generally, the network structure of SH-BDNN is similar to that of UH-BDNN, excluding the removal of the last layer preserving the reconstruction of UH-BDNN. The layer $n-1$ in UH-BDNN becomes the last layer in SH-BDNN. All desirable properties, i.e., semantic similarity preservation, independence, and balance, in SH-BDNN are constrained on the outputs of its last layer.

A. Formulation of SH-BDNN

We define the pairwise label matrix $\mathbf{S}$ as

$$\mathbf{S}_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are same class} \\ -1 & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ are not same class} \end{cases} \tag{19}$$

To achieve the semantic similarity preserving property, we learn the binary codes such that the Hamming distance between learned codes highly correlates with the matrix $\mathbf{S}$, i.e., we want to minimize the quantity $\left\|\frac{1}{L}(\mathbf{H}^{(n)})^T\mathbf{H}^{(n)} - \mathbf{S}\right\|^2$. In addition, to achieve the independence and balance properties of codes, we want to minimize the quantities $\left\|\frac{1}{m}\mathbf{H}^{(n)}(\mathbf{H}^{(n)})^T - \mathbf{I}\right\|^2$ and $\left\|\mathbf{H}^{(n)}\mathbf{1}_{m\times 1}\right\|^2$, respectively.

Algorithm 2 Supervised Hashing with Binary Deep Neural Network (SH-BDNN)

Input:
$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^m \in \mathbb{R}^{D\times m}$: labeled training data; L: code length; T: maximum iteration number; n: number of layers; $\{s_l\}_{l=2}^n$: number of units of layers $2 \to n$ (note: $s_n = L$); $\lambda_1, \lambda_2, \lambda_3, \lambda_4$.

Output:
Parameters $\{\mathbf{W}^{(l)}, \mathbf{c}^{(l)}\}_{l=1}^{n-1}$

1: Compute pairwise label matrix $\mathbf{S}$ using (19).
2: Initialize $\mathbf{B}_{(0)} \in \{-1, 1\}^{L\times m}$ using ITQ [22]
3: Initialize $\{\mathbf{c}^{(l)}\}_{l=1}^{n-1} = \mathbf{0}_{s_{l+1}\times 1}$. Initialize $\{\mathbf{W}^{(l)}\}_{l=1}^{n-1}$ by getting the top s_{l+1} eigenvectors from the covariance matrix of $\mathbf{H}^{(l)}$.
4: Fix $\mathbf{B}_{(0)}$, compute $(\mathbf{W}, \mathbf{c})_{(0)}$ with $(\mathbf{W}, \mathbf{c})$ step using initialized $\{\mathbf{W}^{(l)}, \mathbf{c}^{(l)}\}_{l=1}^{n-1}$ (line 3) as starting point for L-BFGS.
5: **for** $t = 1 \to T$ **do**
6: Fix $(\mathbf{W}, \mathbf{c})_{(t-1)}$, compute $\mathbf{B}_{(t)}$ with $\mathbf{B}$ step
7: Fix $\mathbf{B}_{(t)}$, compute $(\mathbf{W}, \mathbf{c})_{(t)}$ with $(\mathbf{W}, \mathbf{c})$ step using $(\mathbf{W}, \mathbf{c})_{(t-1)}$ as starting point for L-BFGS.
8: **end for**
9: Return $(\mathbf{W}, \mathbf{c})_{(T)}$

Follow the same reformulation and relaxation as UH-BDNN (Sec. III-A), we solve the following constrained optimization which ensures the binary constraint, the semantic similarity preserving, the independence, and the balance properties of codes.

$$\begin{aligned} \min_{\mathbf{W},\mathbf{c},\mathbf{B}} J \quad = \quad & \frac{1}{2m}\left\|\frac{1}{L}(\mathbf{H}^{(n)})^T\mathbf{H}^{(n)} - \mathbf{S}\right\|^2 + \frac{\lambda_1}{2}\sum_{l=1}^{n-1}\left\|\mathbf{W}^{(l)}\right\|^2 \\ & + \frac{\lambda_2}{2m}\left\|\mathbf{H}^{(n)} - \mathbf{B}\right\|^2 + \frac{\lambda_3}{2}\left\|\frac{1}{m}\mathbf{H}^{(n)}(\mathbf{H}^{(n)})^T - \mathbf{I}\right\|^2 \\ & + \frac{\lambda_4}{2m}\left\|\mathbf{H}^{(n)}\mathbf{1}_{m\times 1}\right\|^2 \end{aligned} \tag{20}$$

$$\text{s.t. } \mathbf{B} \in \{-1, 1\}^{L\times m} \tag{21}$$

(20) under constraint (21) is our formulation for supervised hashing. The main difference in formulation between UH-BDNN (8) and SH-BDNN (20) is that the reconstruction term preserving the neighbor similarity in UH-BDNN (8) is replaced by the term preserving the label similarity in SH-BDNN (20).

B. Optimization

To solve (20) under constraint (21), we use alternating optimization, which comprises two steps over $(\mathbf{W}, \mathbf{c})$ and $\mathbf{B}$.

1) $(\mathbf{W}, \mathbf{c})$ *step:* When fixing $\mathbf{B}$, (20) becomes unconstrained optimization. We used *L-BFGS* [63] optimizer with backpropagation for solving. The gradients of the objective function J (20) w.r.t. different parameters are computed as follows.

Let us define

$$\Delta^{(n)} = \left[\frac{1}{mL}\mathbf{H}^{(n)}\left(\mathbf{V}+\mathbf{V}^T\right)+\frac{\lambda_2}{m}\left(\mathbf{H}^{(n)}-\mathbf{B}\right) + \frac{2\lambda_3}{m}\left(\frac{1}{m}\mathbf{H}^{(n)}(\mathbf{H}^{(n)})^T-\mathbf{I}\right)\mathbf{H}^{(n)} + \frac{\lambda_4}{m}\left(\mathbf{H}^{(n)}\mathbf{1}_{m\times m}\right)\right]\odot f^{(n)'}(\mathbf{Z}^{(n)}) \quad (22)$$

where $\mathbf{V}=\frac{1}{L}(\mathbf{H}^{(n)})^T\mathbf{H}^{(n)}-\mathbf{S}$.

$$\Delta^{(l)} = \left((\mathbf{W}^{(l)})^T\Delta^{(l+1)}\right)\odot f^{(l)'}(\mathbf{Z}^{(l)}), \forall l = n-1,\cdots,2 \quad (23)$$

where $\mathbf{Z}^{(l)} = \mathbf{W}^{(l-1)}\mathbf{H}^{(l-1)}+\mathbf{c}^{(l-1)}\mathbf{1}_{1\times m}$, $l=2,\cdots,n$; $\odot$ denotes the Hadamard product.

Then $\forall l = n-1,\cdots,1$, we have

$$\frac{\partial J}{\partial \mathbf{W}^{(l)}} = \Delta^{(l+1)}(\mathbf{H}^{(l)})^T+\lambda_1\mathbf{W}^{(l)} \quad (24)$$

$$\frac{\partial J}{\partial \mathbf{c}^{(l)}} = \Delta^{(l+1)}\mathbf{1}_{m\times 1} \quad (25)$$

2) $\mathbf{B}$ step: When fixing $(\mathbf{W},\mathbf{c})$, we can rewrite problem (20) as

$$\min_{\mathbf{B}} J = \left\|\mathbf{H}^{(n)}-\mathbf{B}\right\|^2 \quad (26)$$

$$\text{s.t. } \mathbf{B}\in\{-1,1\}^{L\times m} \quad (27)$$

It is easy to see that the optimal solution for (26) under constraint (27) is $\mathbf{B}=sign(\mathbf{H}^{(n)})$.

The proposed SH-BDNN method is summarized in Algorithm **2**. In Algorithm **2**, $\mathbf{B}_{(t)}$ and $(\mathbf{W},\mathbf{c})_{(t)}$ are values of $\mathbf{B}$ and $\{\mathbf{W}^{(l)},\mathbf{c}^{(l)}\}_{l=1}^{n-1}$ at iteration t, respectively.

C. Evaluation of Supervised Hashing with Binary Deep Neural Network

In this section, we evaluate and compare our proposed SH-BDNN to the state-of-the-art supervised hashing methods including ITQ-CCA [22], Kernel-based Supervised Hashing (KSH) [30], Binary Reconstructive Embedding (BRE) [34], and Supervised Discrete Hashing (SDH) [35]. We use the released implementations and the suggested parameters provided by the authors for all compared methods.

1) Dataset, evaluation protocol, and implementation notes:

a) Dataset: We evaluate and compare methods on the CIFAR-10, MNIST, and SUN397 datasets. The descriptions of the first two datasets are presented in section III-C1.

The SUN397 [50] dataset contains approximately 108,000 images from 397 scene categories. We select 42 categories that contain more than 500 images, which results in a dataset of approximately 35,000 images in total. We then randomly sample 100 images per class from the dataset to form a query set of 4,200 images. The remaining images are used as the training set and the database set. Each image is represented by an 800-dimensional feature vector extracted by PCA from 4096-dimensional CNN feature produced by AlexNet [39].

b) Evaluation protocol: We report the retrieval performances by using the two standard metrics in the literature [35, 22, 30]: precision of Hamming radius 2 (precision@2) and mean Average Precision (mAP).

c) Implementation notes: The network configuration of SH-BDNN is similar to the configuration of UH-BDNN except the final layer is removed. The parameters λ_1, λ_2, λ_3, and λ_4 are empirically set using cross validation as 10^{-3}, 5, 1, and 10^{-4}, respectively. The max iteration number T is empirically set to 5.

Following the settings in ITQ-CCA [22] and SDH [35], all training samples are used in the learning for these two methods. For SH-BDNN, KSH [30] and BRE [34] where the label information is leveraged by the pairwise label matrix, we randomly select 300 training samples from each class to form a training set. In the supervised setting, the class label is used to define the ground truths of queries.

2) Retrieval results: Fig. 3 and Table VI show comparative results between the proposed SH-BDNN and other supervised hashing methods on the CIFAR10, MNIST, and SUN397 datasets.

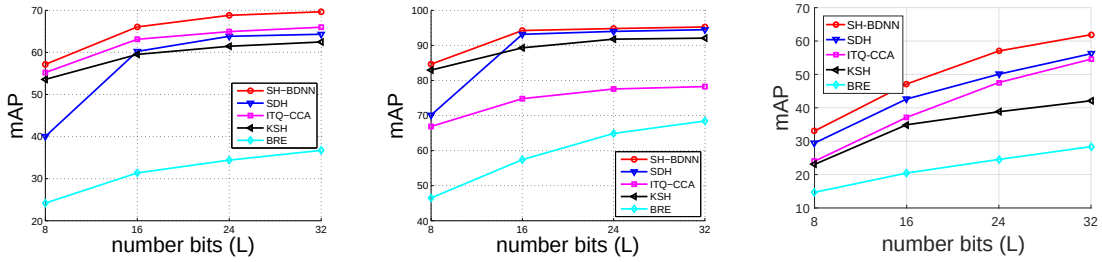
On the CIFAR10 dataset, Fig. 3(a) and Table VI show that our proposed SH-BDNN clearly outperforms all compared methods at all code lengths by a fair margin in both evaluation metrics, i.e., mAP and precision@2. The best competitor to SH-BDNN on this dataset is CCA-ITQ [22]. The more improvements of SH-BDNN over CCA-ITQ are observed at high code lengths, i.e., SH-BDNN outperforms CCA-ITQ by approximately 4% at $L=24$ and $L=32$.

On the MNIST dataset, Fig. 3(b) and Table VI show that the proposed SH-BDNN significant outperforms the current state-of-the-art SDH [35] at low code lengths, i.e., $L=8$. At higher code lengths, however, the performances of SH-BDNN and SDH [35] are comparable. Moreover, SH-BDNN significantly outperforms other methods, i.e., ITQ-CCA [22], KSH [30], and BRE [34], on both mAP and precision@2.

On the SUN397 dataset, the proposed SH-BDNN outperforms other competitors at all code lengths in terms of both mAP and precision@2. The best competitor to SH-BDNN on this dataset is SDH [35]. At high code lengths (e.g., $L=24,32$), SH-BDNN achieves more improvements over SDH.

V. Supervised Hashing with End-to-End Binary Deep Neural Network (E2E-BDNN)

Even though the proposed SH-BDNN can significantly enhance the discriminative power of the binary codes, similar to other hashing methods, its capability is partially dependent on the discriminative power of the image features. The recent end-to-end deep learning-based supervised hashing methods [42, 43, 44] have shown that joint learning image representations and binary hash codes in an end-to-end fashion can boost the retrieval accuracy. Therefore, in this section, we propose to extend the proposed SH-BDNN to an end-to-end framework. Specifically, we integrate the convolutional neural network (CNN) with our supervised hashing network (SH-BDNN) into an unified end-to-end deep architecture, namely, the End-to-End Binary Deep Neural Network (E2E-BDNN), which can jointly learn visual features and binary representations of images. In the following, we first introduce our proposed network architecture. We then describe the training



(a) CIFAR10 (b) MNIST (c) SUN397

Fig. 3. mAP comparison between SH-BDNN and state-of-the-art supervised hashing methods on CIFAR10, MNIST and SUN397 datasets.

TABLE VI
PRECISION AT HAMMING DISTANCE $r = 2$ COMPARISON BETWEEN SH-BDNN AND STATE-OF-THE-ART SUPERVISED HASHING METHODS ON CIFAR10, MNIST, AND SUN397 DATASETS.

	CIFAR10				MNIST				SUN397			
L	8	16	24	32	8	16	24	32	8	16	24	32
SH-BDNN	54.12	67.32	69.36	69.62	84.26	94.67	94.69	95.51	15.52	41.98	52.53	56.82
SDH[35]	31.60	62.23	67.65	67.63	36.49	93.00	93.98	94.43	13.89	40.39	49.54	53.25
ITQ-CCA[22]	49.14	65.68	67.47	67.19	54.35	79.99	84.12	84.57	13.22	37.53	50.07	53.12
KSH[30]	44.81	64.08	67.01	65.76	68.07	90.79	92.86	92.41	12.64	40.67	49.29	46.45
BRE[34]	23.84	41.11	47.98	44.89	37.67	69.80	83.24	84.61	9.26	26.95	38.36	40.36

process. Finally, we present experiments on various benchmark datasets.

A. Network architecture

The network consists of three main components: *(i)* a feature extractor, *(ii)* a dimensional reduction layer, and *(iii)* a binary optimizer component. We utilize the AlexNet model [39] as the feature extractor component of the E2E-BDNN. In our configuration, we remove the last layer of AlexNet, namely the `softmax` layer, and consider its last fully connected layer (`fc7`) as the image representation.

The dimensional reduction component (the DR layer) involves a fully connected layer for reducing the high dimensional image representations outputted by the feature extractor component into lower dimensional representations. We use the identity function as the activation function for this DR layer. Thus, the DR layer performs a linear projection to reduce the dimension of AlexNet features. The reduced representations are then used as inputs for the following binary optimizer component.

The binary optimizer component of E2E-BDNN is similar to SH-BDNN. Thus, we also constrain the output codes of E2E-BDNN to be binary. These codes also have the desired properties such as semantic similarity preservation, independence, and balance. By using the same design as SH-BDNN for the last component of E2E-BDNN, it allows us to observe the advantages of the end-to-end architecture over SH-BDNN.

The training data for the E2E-BDNN are labelled images, contrasting with SH-BDNN which uses visual features such as GIST [68], SIFT [67] or deep features from convolutional deep networks. Given the input labeled images, we aim to learn binary codes with the aforementioned desired properties, i.e., semantic similarity preservation, independence, and balance. To achieve these properties on the codes, we use a similar objective function as SH-BDNN. However, it is important to mention that in SH-BDNN, by its non end-to-end architecture, we can feed the whole training set into the network at a time during training, which does not hold for E2E-BDNN. Due to the memory consumption of the end-to-end architecture, we can only feed a minibatch of images into the network at a time during training. Technically, let $\mathbf{H}$ be the output of the last fully connected layer of E2E-BDNN for a minibatch of size m_b; $\mathbf{S}$ be the similarity matrix defined over the minibatch (using equation (19)); and $\mathbf{B}$ serve as an auxiliary variable. Similar to SH-BDNN, we train the network to minimize the following constrained loss function

$$\begin{aligned}\min_{\mathbf{W},\mathbf{B}} J \quad = \quad & \frac{\lambda_1}{2m_b}\left\|\frac{1}{L}\mathbf{H}^T\mathbf{H}-\mathbf{S}\right\|^2 \\ & +\frac{\lambda_2}{2m_b}\left\|\mathbf{H}-\mathbf{B}\right\|^2+\frac{\lambda_3}{2}\left\|\frac{1}{m_b}\mathbf{H}\mathbf{H}^T-\mathbf{I}\right\|^2 \\ & +\frac{\lambda_4}{2m_b}\left\|\mathbf{H}\mathbf{1}_{m_b\times 1}\right\|^2 \qquad (28)\end{aligned}$$

$$\text{s.t. } \mathbf{B}\in\{-1,1\}^{L\times m_b} \qquad (29)$$

B. Training

The training procedure for E2E-BDNN is presented in Algorithm **3**. In Algorithm **3**, $\mathbf{X}_{(t)}$ and $\mathbf{B}_{(t)}\in\{-1,1\}^{L\times m_b}$ are a minibatch sampled from the training set at iteration t and its corresponding binary codes, respectively. $\mathbf{B}^{(k)}$ is the binary codes of the whole training set $\mathbf{X}$ at iteration k; and $\mathbf{W}_{(t)}^{(k)}$ is the network weight when learning up to iterations t and k.

At first (line 1 in Algorithm **3**), we initialize the network parameter $\mathbf{W}_{(0)}^{(0)}$ as follows. *(i)* The feature extractor component is initialized by the pretrained AlexNet model [39]. *(ii)* The dimensional reduction (DR) layer is initialized by the top eigenvectors extracted from the covariance matrix of the

Algorithm 3 End-to-End Binary Deep Neural Network (E2E-BDNN) Learning

Input:
$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{m}$: labeled training images; m_b: minibatch size; L: code length; K, T: maximum iteration. $\lambda_1, \lambda_2, \lambda_3, \lambda_4$: hyperparameters.
Output:
Network parameters $\mathbf{W}$

1: Initialize the network $\mathbf{W}_{(0)}^{(0)}$
2: Initialize $\mathbf{B}^{(0)} \in \{-1, 1\}^{L \times m}$ via ITQ [22]
3: **for** $k = 1 \rightarrow K$ **do**
4: **for** $t = 1 \rightarrow T$ **do**
5: A minibatch $\mathbf{X}_{(t)}$ is sampled from $\mathbf{X}$
6: Compute the corresponding similarity matrix $\mathbf{S}_{(t)}$
7: From $\mathbf{B}^{(k-1)}$, sample $\mathbf{B}_{(t)}$ corresponding to $\mathbf{X}_{(t)}$
8: Fix $\mathbf{B}_{(t)}$, optimize $\mathbf{W}_{(t)}^{(k)}$ via SGD
9: **end for**
10: Update $\mathbf{B}^{(k)}$ by $\mathbf{W}_{(T)}^{(k)}$
11: **end for**
12: Return $\mathbf{W}_{(T)}^{(K)}$

AlexNet features (i.e., the outputs of `fc7` layer) of the training set. *(iii)* The binary optimizer component is initialized by the trained SH-BDNN models (in Section IV).

We then initialize the binary code matrix of the whole dataset $\mathbf{B}^{(0)} \in \{-1, 1\}^{L \times n}$ via ITQ [22] (line 2 in the Algorithm **3**). Here, AlexNet features are used as training inputs for ITQ.

In each iteration t of Algorithm **3**, we only sample a minibatch $\mathbf{X}_{(t)}$ from the training set to feed into the network (line 5 in Algorithm **3**). Thus, after T iterations, we exhaustively sample all training data. In each iteration t, we first create the similarity matrix $\mathbf{S}_{(t)}$ (using equation (19)) corresponding to $\mathbf{X}_{(t)}$, as well as the $\mathbf{B}_{(t)}$ matrix (lines 6 and 7 in Algorithm **3**). Since $\mathbf{B}_{(t)}$ has already been computed, we can fix that variable and optimize the network parameter $\mathbf{W}_{(t)}^{(k)}$ by standard backpropagation with Stochastic Gradient Descent (SGD) (line 8 in Algorithm **3**). After T iterations, since the network was exhaustively learned from the whole training set, we update $\mathbf{B}^{(k)} = sign(\mathbf{F})$ (line 10 in the Algorithm **3**), where $\mathbf{F}$ is the outputs of the last fully connected layer for all training samples. We then repeat the optimization procedure until it reaches a criterion, i.e., after K iterations.

Implementation details The proposed E2E-BDNN is implemented in MATLAB with MatConvNet library [69]. All experiments are conducted on a workstation machine with a GPU Titan X. Regarding the hyperparameters of the loss function, we empirically set $\lambda_1 = 10^{-1}$, $\lambda_2 = 10^{-2}$, $\lambda_3 = 10^{-2}$ and $\lambda_4 = 10^{-3}$. The learning rate is set to 10^{-4} and the weight decay is set to 5×10^{-4}. The minibatch size is 256.

C. Evaluation of End-to-End Binary Deep Neural Network (E2E-BDNN)

Since we have already compared SH-DBNN to other supervised hashing methods in Section IV-C, in this experiment we focus on comparing E2E-BDNN with SH-BDNN. We also compare the proposed E2E-BDNN to other end-to-end hashing methods [43, 42, 44, 45, 54, 56].

a) Comparison between SH-BDNN and E2E-BDNN: Table VII presents the comparative mAP between SH-BDNN and E2E-BDNN. The results shows that E2E-BDNN consistently improves over SH-BDNN at all code lengths on all datasets. The large improvements of E2E-BDNN over SH-BDNN are observed on the CIFAR10 and MNIST datasets, especially at low code lengths, i.e., on the CIFAR10 dataset, E2E-BDNN outperforms SH-BDNN by 7.7% and 5% at $L = 8$ and $L = 16$, respectively; on the MNIST dataset, E2E-BDNN outperforms SH-BDNN by 4.2% and 3.8% at $L = 8$ and $L = 16$, respectively. On the SUN397 dataset, the improvements of E2E-BDNN over SH-BDNN are clearer at high code lengths, i.e., E2E-BDNN outperforms SH-BDNN by 2.5% and 2.7% at $L = 24$ and $L = 32$, respectively. The improvements of E2E-BDNN over SH-BDNN confirm the effectiveness of the proposed end-to-end architecture for learning discriminative binary codes.

b) Comparison between E2E-BDNN and other end-to-end supervised hashing methods: We also compare our proposed deep networks SH-BDNN and E2E-BDNN with other end-to-end supervised hashing architectures, i.e., Hashing with Deep Neural Network (DNNH) [44], Deep Hashing Network (DHN) [54], Deep Quantization Network (DQN) [45], Deep Semantic Ranking Hashing (DSRH) [42], Deep Pairwise Supervised Hashing (DPSH) [56], and Deep Regularized Similarity Comparison Hashing (DRSCH) [43]. In those works, the authors propose the frameworks in which the image features and hash codes are simultaneously learned by combining a CNN and a binary quantization layer into a unified model. However, their binary mapping layer only applies a simple operation, e.g., an approximation of the $sign$ function ($logistic$ [42, 44], $tanh$ [43]), l_1 norm approximation of binary constraints [54]. Our SH-BDNN and E2E-BDNN advances over those works in the way to map the image features to the binary codes. Furthermore, our learned codes ensure good properties, i.e. independence and balance, while [44, 43, 45, 54, 56] do not consider such properties, and [42] only considers the balance of codes. It is worth noting that, in [44, 42, 43, 54, 56], different settings are used for evaluation. For a fair comparison, following those works, we setup two different experimental settings on CIFAR10 as follows

- Setting 1: following [44, 45, 54], we randomly sample 100 images per class to form 1K testing images. The remaining 59K images are used as database images. Furthermore, 500 images per class are sampled from the database to form 5K training images.
- Setting 2: following [42, 43], we randomly sample 1K images per class to form 10K testing images. The remaining 50K images serve as the training set. In the test phase, each test image is searched through the test set by the leave-one-out procedure.

Table VIII shows the comparative mAP between our methods and DNNH [44], DQN [45], DPSH [56], and DHN [54] on the CIFAR10 dataset with setting 1. The results show that even with the non end-to-end approach, our SH-BDNN outperforms DNNH and DQN and is comparable to DPSH at all code lengths. With the end-to-end approach, it helps

TABLE VII
MAP COMPARISON BETWEEN SH-BDNN AND E2E-BDNN ON CIFAR10, MNIST, AND SUN397 DATASETS.

	CIFAR10				MNIST				SUN397			
L	8	16	24	32	8	16	24	32	8	16	24	32
SH-BDNN	57.15	66.04	68.81	69.66	84.65	94.24	94.80	95.25	33.06	47.13	57.02	61.89
E2E-BDNN	64.83	71.02	72.37	73.56	88.82	98.03	98.16	98.21	34.15	48.21	59.51	64.58

TABLE VIII
MAP COMPARISON BETWEEN E2E-BDNN, SH-BDNN AND DNNH[44], DQN[45], DPSH [56], DHN [54] ON CIFAR10 (SETTING 1).

L	24	32	48
E2E-BDNN	60.02	61.35	63.59
SH-BDNN	57.30	58.66	60.08
DNNH[44]	56.60	55.80	58.10
DQN[45]	55.80	56.40	58.00
DPSH [56]	57.57	58.54	60.17
DHN[54]	59.40	60.30	62.10

TABLE IX
MAP COMPARISON BETWEEN E2E-BDNN, SH-BDNN AND DSRH[42], DRSCH[43] ON CIFAR10 (SETTING 2).

L	24	32	48
E2E-BDNN	67.16	68.72	69.23
SH-BDNN	65.21	66.22	66.53
DSRH [42]	61.08	61.74	61.77
DRSCH [43]	62.19	62.87	63.05

to boost the performance of the SH-BDNN. The proposed E2E-BDNN outperforms all compared methods, DNNH [44], DHN [54], DQN [45], and DPSH [56]. It is worth noting that in [44], increasing the code length does not necessarily boost the retrieval accuracy, i.e., [44] reports a mAP of 55.80 at $L = 32$, while a higher mAP, i.e., 56.60 is reported at $L = 24$. In contrast to [44], both SH-BDNN and E2E-BDNN improve mAP when increasing the code length. Additionally, we also observe that both DPSH and DHN maximize log-likelihood objective functions to ensure that (dis)similar input pairs result in (dis)similar output pairs. Hence, the large performance gaps between DHN and DPSH show that the optimization method proposed in DPSH does not well handle the binary constraint. Specifically, DPSH resorts to the $sign$ function to obtain the binary codes during optimization but ignores its ill-posed gradient problem. More importantly, the superior performance of our proposed method over the compared methods confirms the effectiveness of the proposed approach in dealing with binary constrains and the provision of desired properties such as independence and balance on the produced codes.

Table IX presents the comparative mAP between the proposed SH-BDNN, E2E-BDNN and the competitors DSRH [42], DRSCH [43] on the CIFAR10 dataset with setting 2. The results clearly show that the proposed E2E-BDNN significantly outperforms DSRH [42] and DRSCH [43] at all code lengths. Compared with the best competitor DRSCH [43], the improvements of E2E-BDNN over DRSCH range from 5% to 6% at different code lengths. Furthermore, we can see that even with the non end-to-end approach, the proposed SH-BDNN also outperforms DSRH [42] and DRSCH [43].

VI. CONCLUSION

In this paper, we propose three deep hashing neural networks for learning compact binary presentations. Firstly, we introduce UH-BDNN and SH-BDNN for unsupervised and supervised hashing respectively. In our novel designs, the networks are constrained to directly produce binary codes at one layer. The designs also ensure good properties for produced codes, i.e., similarity preservation, independence, and balance. Together with the designs, we also propose alternating optimization schemes that allow us to effectively deal with binary constraints on the codes. We then propose to extend SH-BDNN to an end-to-end deep hashing framework (E2E-BDNN) that jointly learns the image representations and the binary codes. The solid experimental results on various benchmark datasets show that the proposed methods compare favorably or outperform state-of-the-art hashing methods.

ACKNOWLEDGEMENT

This research was supported by the National Research Foundation Singapore under its AI Singapore Programme (Award number: AISG-100E-2018-005). This work was also supported by both ST Electronics and the National Research Foundation (NRF), Prime Minister's Office, Singapore under Corporate Laboratory at University Scheme (Programme Title: STEE Infosec - SUTD Corporate Laboratory).

REFERENCES

[1] R. Arandjelovic and A. Zisserman, "All about VLAD," in *CVPR*, 2013.
[2] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *CVPR*, 2010.
[3] T.-T. Do and N.-M. Cheung, "Embedding based on function approximation for large scale image search," *TPAMI*, 2017.
[4] H. Jégou and A. Zisserman, "Triangulation embedding and democratic aggregation for image search," in *CVPR*, 2014.
[5] T.-T. Do, Q. Tran, and N.-M. Cheung, "FAemb: a function approximation-based embedding method for image retrieval," in *CVPR*, 2015.
[6] A. Torii, R. Arandjelovic, J. Sivic, M. Okutomi, and T. Pajdla, "24/7 place recognition by view synthesis," *TPAMI*, pp. 257–271, 2018.
[7] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: CNN architecture for weakly supervised place recognition," *TPAMI*, pp. 257–271, 2018.
[8] A. Torii, J. Sivic, M. Okutomi, and T. Pajdla, "Visual place recognition with repetitive structures," *TPAMI*, pp. 2346–2359, 2015.

[9] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt, "Image retrieval for image-based localization revisited," in *BMVC*, 2012.

[10] A. Irschara, C. Zach, J. Frahm, and H. Bischof, "From structure-from-motion point clouds to fast location recognition," in *CVPR*, 2009.

[11] N. Tran, D. L. Tan, A. Doan, T.-T. Do, T. Bui, M. Tan, and N.-M. Cheung, "On-device scalable image-based localization via prioritized cascade search and fast one-many RANSAC," *TIP*, 2019.

[12] T. Hoang, T.-T. Do, D.-K. L. Tan, and N.-M. Cheung, "Selective deep convolutional features for image retrieval," in *ACM Multimedia 2017*, 2017.

[13] K. Grauman and R. Fergus, "Learning binary hash codes for large-scale image search," *Machine Learning for Computer Vision*, 2013.

[14] J. Wang, H. T. Shen, J. Song, and J. Ji, "Hashing for similarity search: A survey," *CoRR*, 2014.

[15] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *TPAMI*, 2017.

[16] J. Wang, W. Liu, S. Kumar, and S. Chang, "Learning to hash for indexing big data - A survey," *Proceedings of the IEEE*, vol. 104, no. 1, pp. 34–57, 2016.

[17] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *VLDB*, 1999.

[18] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *ICCV*, 2009.

[19] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *NIPS*, 2009.

[20] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *TPAMI*, pp. 2143–2157, 2009.

[21] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *NIPS*, 2008.

[22] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *TPAMI*, pp. 2916–2929, 2013.

[23] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *CVPR*, 2013.

[24] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-e. Yoon, "Spherical hashing," in *CVPR*, 2012.

[25] W. Kong and W.-J. Li, "Isotropic hashing," in *NIPS*, 2012.

[26] X. Zhu, X. Li, S. Zhang, Z. Xu, L. Yu, and C. Wang, "Graph PCA hashing for similarity search," *IEEE Trans. Multimedia*, pp. 2033–2044, 2017.

[27] D. Wang, P. Cui, M. Ou, and W. Zhu, "Learning compact hash codes for multimodal representations using orthogonal deep structure," *IEEE Trans. Multimedia*, pp. 1404–1416, 2015.

[28] X. Mao, Y. Yang, and N. Li, "Hashing with pairwise correlation learning and reconstruction," *IEEE Trans. Multimedia*, pp. 382–392, 2017.

[29] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, "LDAhash: Improved matching with smaller descriptors," *TPAMI*, pp. 66–78, 2012.

[30] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *CVPR*, 2012.

[31] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for large-scale search," *TPAMI*, pp. 2393–2406, 2012.

[32] M. Norouzi, D. J. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *NIPS*, 2012.

[33] G. Lin, C. Shen, D. Suter, and A. van den Hengel, "A general two-step approach to learning-based hashing," in *ICCV*, 2013.

[34] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *NIPS*, 2009.

[35] F. Shen, C. Shen, W. Liu, and H. Tao Shen, "Supervised discrete hashing," in *CVPR*, 2015.

[36] T.-T. Do, A.-D. Doan, D.-T. Nguyen, and N.-M. Cheung, "Binary hashing with semidefinite relaxation and augmented lagrangian," in *ECCV*, 2016.

[37] T.-T. Do, K. Le, T. Hoang, H. Le, T. V. Nguyen, and N. Cheung, "Simultaneous feature aggregating and hashing for compact binary code learning," *TIP*, 2019.

[38] T.-T. Do, T. Hoang, D. L. Tan, H. Le, T. Nguyen, and N.-M. Cheung, "From selective deep convolutional features to compact binary representations for image retrieval," *ACM Transactions on Multimedia Computing, Communications, and Applications*, 2019.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012.

[40] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *CVPRW*, 2014.

[41] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *CVPR*, 2015.

[42] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *CVPR*, 2015.

[43] 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," *TIP*, pp. 4766–4779, 2015.

[44] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *CVPR*, 2015.

[45] Y. Cao, M. Long, J. Wang, H. Zhu, and Q. Wen, "Deep quantization network for efficient image retrieval," in *AAAI*, 2016.

[46] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, pp. 1527–1554, 2006.

[47] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *CVPR*, 2015.

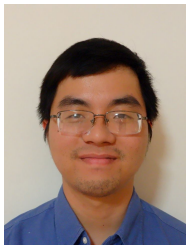
[48] M. A. Carreira-Perpinan and R. Raziperchikolaei, "Hashing with binary autoencoders," in *CVPR*, 2015.

[49] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *ECCV*, 2016.

[50] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba, "SUN database: Large-scale scene recognition

from abbey to zoo," in *CVPR*, 2010.
[51] R. Salakhutdinov and G. E. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, pp. 969–978, 2009.
[52] J. Lu, V. E. Liong, and J. Zhou, "Deep hashing for scalable image search," *TIP*, 2017.
[53] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *CVPR*, 2016.
[54] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for efficient similarity retrieval," in *AAAI*, 2016.
[55] Z. Cao, M. Long, J. Wang, and P. S. Yu, "Hashnet: Deep learning to hash by continuation," in *ICCV*, 2017.
[56] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *IJCAI*, 2016.
[57] E. Yang, C. Deng, T. Liu, W. Liu, and D. Tao, "Semantic structure-based unsupervised deep hashing," in *IJCAI-18*, 2018, pp. 1064–1070.
[58] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization," *IEEE TPAMI*, vol. 40, no. 12, pp. 3034–3044, 2018.
[59] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, 2014.
[60] H. Zhang, M. Wang, R. Hong, and T. Chua, "Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing," in *ACM MM*, 2016.
[61] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, 2016.
[62] J. Nocedal and S. J. Wright, *Numerical Optimization, Chapter 17*, 2nd ed. World Scientific, 2006.
[63] D. C. Liu and J. Nocedal, "On the limited memory bfgs method for large scale optimization," *Mathematical Programming*, vol. 45, pp. 503–528, 1989.
[64] A. Krizhevsky, "Learning multiple layers of features from tiny images," University of Toronto, Tech. Rep., 2009.
[65] Y. Lecun and C. Cortes, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist/.
[66] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *TPAMI*, pp. 117–128, 2011.
[67] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *IJCV*, pp. 91–110, 2004.
[68] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *IJCV*, pp. 145–175, 2001.
[69] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *ACM Multimedia*, 2015.

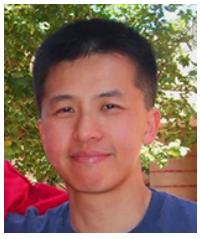
Thanh-Toan Do is currently a lecturer at the Department of Computer Science, University of Liverpool (UoL), United Kingdom. He obtained a Ph.D. in Computer Science from INRIA, Rennes, France in 2012. Before joining UoL, he was a Research Fellow at the Singapore University of Technology and Design, Singapore (2013 - 2016) and the University of Adelaide, Australia (2016 - 2018). His research interests include Computer Vision and Machine Learning.



Tuan Hoang is currently a Ph.D. student at Singapore University of Technology and Design (SUTD). Before joining SUTD, he received the bachelor degree in Electrical Engineering from Portland State University, in 2014. His research interests are content-based image retrieval and image hashing.

Khoa Le received a BSc for an honours degree from the University of Science, Vietnam National University, in 2015. Since 2016, he has been a research assistant at Singapore University of Technology and Design (SUTD). His current research interests are deep learning and image retrieval.

Anh-Dzung Doan is currently a Ph.D. student at the University of Adelaide (UoA). He received a BSc for an honours degree from the University of Science, Vietnam National University, in 2013. Before joining UoA, he was a research assistant at Singapore University of Technology and Design (SUTD) (2014-2017). His research interests include 3D computer vision, robotic vision, and image retrieval.



Ngai-Man Cheung received a Ph.D. degree in electrical engineering from the University of Southern California, Los Angeles, CA, in 2008. He is currently an Associate Professor with the Singapore University of Technology and Design (SUTD). From 2009 - 2011, he was a postdoctoral researcher with the Image, Video and Multimedia Systems group at Stanford University, Stanford, CA. He has also held research positions with Texas Instruments Research Center Japan, Nokia Research Center, IBM T. J. Watson Research Center, HP Labs Japan, Hong Kong University of Science and Technology (HKUST), and Mitsubishi Electric Research Labs (MERL). His work has resulted in 10 U.S. patents granted with several pending. His research interests include signal, image, and video processing, and computer vision.