Unsupervised Deep Hashing With Pseudo Labels for Scalable Image Retrieval

Haofeng Zhang , Li Liu, Yang Long, and Ling Shao , *Senior Member, IEEE*

***Abstract*— In order to achieve efficient similarity searching, hash functions are designed to encode images into low-dimensional binary codes with the constraint that similar features will have a short distance in the projected Hamming space. Recently, deep learning-based methods have become more popular, and outperform traditional non-deep methods. However, without label information, most state-of-the-art unsupervised deep hashing (DH) algorithms suffer from severe performance degradation for unsupervised scenarios. One of the main reasons is that the ad-hoc encoding process cannot properly capture the visual feature distribution. In this paper, we propose a novel unsupervised framework that has two main contributions: 1) we convert the unsupervised DH model into supervised by discovering pseudo labels; 2) the framework unifies likelihood maximization, mutual information maximization, and quantization error minimization so that the pseudo labels can maximumly preserve the distribution of visual features. Extensive experiments on three popular data sets demonstrate the advantages of the proposed method, which leads to significant performance improvement over the state-of-the-art unsupervised hashing algorithms.**

***Index Terms*— Image retrieval, unsupervised hashing, pseudo labels.**

I. Introduction

FAST similarity search is one of the fundamental requirements for large-scale visual information retrieval applications [1]–[10]. Hashing, the most widely used method, usually exploits high-dimensional visual feature space to find a similarity-preserved low-dimensional Hamming space. Hash codes enormously reduce the requirement of storage space and improve the computational efficiency. Recently, there are many hashing methods [11]–[18] that have achieved excellent performance.

The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Xiaochun Cao. *(Haofeng Zhang and Li Liu contributed equally to this work.) (Corresponding author: Ling Shao.)*

H. Zhang is with the School of Computer Science and Engineering, Nanjing University of Science and Technology and University of East Anglia, Nanjing 210094, China (e-mail: zhanghf@njust.edu.cn).

L. Liu is with JD Artificial Intelligence Research (JDAIR), Beijing 100176, China (e-mail: liuli1213@gmail.com).

Y. Long is with the Open Lab, School of Computing, University of Newcastle, Newcastle upon Tyne NE4 5TG, U.K. (e-mail: yang.long@ieee.org).

L. Shao is with JD Artificial Intelligence Research (JDAIR), Beijing 100176, China, and also with the School of Computing Sciences, University of East Anglia, Norwich NR4 7TJ, U.K. (e-mail: ling.shao@ieee.org).

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

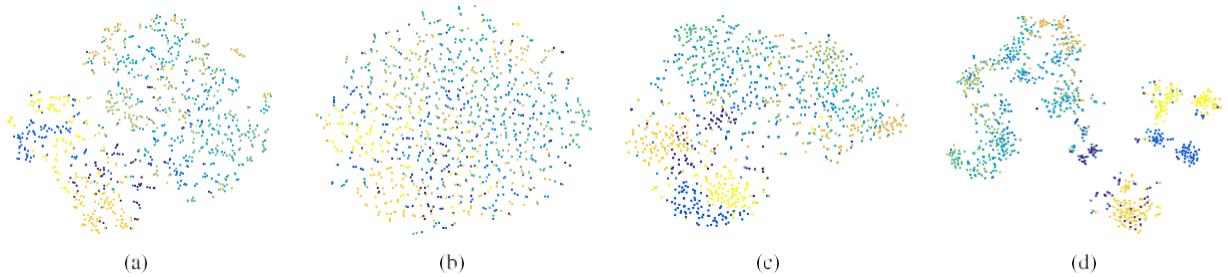
Digital Object Identifier 10.1109/TIP.2017.2781422

Existing unsupervised hashing algorithms can be roughly categorized into data-independent methods and data-dependent methods. For the former one, the most widely used algorithm is Locality-Sensitive Hashing (LSH) [13]. LSH differs from conventional cryptographic hash functions because it aims to maximise the probability of a collision for similar items. However, with the growing data size and the performance requirement, the LSH related methods are proved to be not robust in many challenging scenarios.

To generate compact and effective binary codes, data-dependent methods have gained increasing attention. Unsupervised methods like Spectral Hashing (SH) [19], Principle Component Analysis Hashing (PCAH) [20], Anchor Graph Hashing (AGH) [21], Iterative Quantization (ITQ) [22], and Kernel Hashing [23], [24] can achieve impressive performance. Even though these methods can get good performance, they still can not satisfy the demanded accuracy of image retrieval.

Therefore, lots of supervised methods were proposed to improve retrieval accuracy, e.g. Canonical Correlation Analysis ITQ (CCA-ITQ) [22], Supervised Semantics-preserving Deep Hashing (SSDH) [26], Quantization Based Hashing (QBH) [27], which are all trained with labelled samples and can achieve great improvement in performance. However, these works are all supervised with real labels. Since human annotation is expensive to acquire, it is unrealistic to expect the large-scaled data to be well-labelled. Thus, unsupervised hashing methods still remain the focus.

Earlier unsupervised binary coding methods often use hand-craft features such as GIST [1], [28] and SIFT [29], [30], which can make hashing methods achieve good performance, but still not very satisfactory for our applications. Fortunately, at near recent, on account of great development in the area of deep learning [31]–[34], hashing methods turn to extract deep features automatically using Convolutional Neural Network (CNN) descriptors [35]–[38], which inherit high discriminative property and often get state-of-the-art performance. However, most state-of-the-art unsupervised deep hashing algorithms [39]–[41] suffer from severe performance degradation due to lack of label information, *e.g.* DH [39] and Deepbit [40] use quantisation loss and evenly distribution loss, which can only make the binary codes have large variance, but can not get the real distribution of the data, UH-BDNN [41] adds a reconstruction layer after the binary code layer, which can obtain the semantics of the images, but still can not get the category information, which is more important than the



(a) (b) (c) (d)

Fig. 1. Comparison of learnt binary codes of state-of-the-art methods on the CIFAR-10 dataset using t-SNE [25] with true class labels. (a) AGH. (b) PCAH. (c) ITQ. (d) Ours.

semantics for image retrieval, and the experimental results reported in their original literatures and this paper can also show this point. Therefore, it is necessary to find out the visual feature distribution for unsupervised hashing.

In this paper, a novel hashing method is proposed to discover pseudo labels to train a supervised deep network to solve the unsupervised problem. This method integrates four advantages of current algorithms, **1) Likelihood Maximisation** methods assume the instances in different classes share some latent attributes, in order to ensure the pseudo labels can truly reflect the distribution of the real class labels, likelihood maximisation is used to maximise the joint distribution between visual features and pseudo labels. **2) Quantisation Error Minimisation** methods [20] are proposed for better quantization. One of the state-of-the-art method ITQ [22], which used orthogonal rotation matrix $\boldsymbol{R}$ to project each data point to the nearest vertex of the binary hypercube, and then $\boldsymbol{R}$ is updated to minimise the quantization loss according to this assignment. **3) Correlation Maximisation** approaches, *e.g.* ITQ-CCA [22], aim to maximise the correlation between binary code and visual feature. **4) Deep Hashing** avoids to use hand-crafted features such as GIST [1], SIFT [29], and make the feature extraction and encoding into a unified end-to-end model.

Although promising progress has been achieved, for a realistic scenario, in Fig. 1 we show that only a single method can hardly achieve stable and satisfied performance. One of the main reasons is that a single method might be designed for a particular scenario, which may suffer from severe performance degradations on other tasks under different assumptions. The AGH (Fig. 1 (a)) preserves the likelihood of input visual feature but cannot make the learnt binary code more discriminative. On the contrary, PCAH (Fig. 1 (b)) makes all of the instances are discriminative to each other, but completely loses the structural information. The state-of-the-art ITQ demonstrates a balance between structure preserving and discrimination. However, without label information, ITQ could not further improve the binary code according to real class distributions. Towards more robust hashing model, this paper proposes a unified framework that can effectively incorporate the advantages of these methods. The resultant hash code can simultaneously capture various preferred characteristics, as shown in Fig. 1 (d). Furthermore, by proposing pseudo labels, our binary code is not only robust but can approximate the distribution of true classes labels, *i.e.* most of the clusters are in the same colour. Our contributions can be summarised as follows.

1) In order to leverage the power of supervised deep hashing model on unsupervised scenarios, this paper proposes to discover pseudo labels for a number of grouped instances with a particular distribution.
2) A unified framework is proposed so that the discovered pseudo labels are more close to the true labels of instances.
3) The proposed unified framework can incorporate different favourable learning objectives, which can make the learnt hash code robust to more generalised tasks.

In this paper, we select three criteria to learn the hash codes, which correspond to three different hashing methods: maximising the likelihood to get the approximate distribution of real visual features [42]; maximising the correlation between the input visual features and the variance so that the learnt hash code can have safer boundaries in the Hamming distance space [22]; and the power of supervised Deep model [39]. Our method have some similar steps to CCA-ITQ, but actually they are different, since our method is unsupervised while CCA-ITQ is supervised. Furthermore, quantisation error minimisation and correlation maximisation are just partial steps of our method, and the aim of our method is to combine the advantages of state-of-the-art methods to train a deep network for hashing. Moreover, in extensive experiments, we prove that a single one of the above methods cannot fully address all of the scenarios. By incorporating these criteria into our unified framework, the resultant hash code is more robust for large-scale retrieval tasks on different benchmarks.

The rest of this paper is organised as follows. In Section II, we give a brief review of recent hashing methods. The details of our method are described in Section III. Section IV reports the experimental results. Finally, we conclude this paper in Section V.

II. Related Works

Learning based hashing methods compute binary code by exploiting the training data and its corresponding semantic labels and can avoid the disadvantages of data-independent methods such as LSH [13], which encodes input data so that similar data were projected into the same buckets with high probability, and this type of methods were proved not robust. The learning based hashing methods can be divided into three

categories according to the degree of semantic labels used: unsupervised, semi-supervised, and supervised.

For the first category, label information of training data is not required in the entire training process of unsupervised methods, which try to keep the similarity information between training samples in the original space when the samples were projected into Hamming space. Representative algorithms such as SH [19], PCAH [20], ITQ [22], Kernelized Locality-Sensitive Hashing (KLSH) [43], AGH [21], Spherical Hashing (SpH) [44], K-Means Hashing (KMH) [42], can get pretty good performance. SH [19] generates efficient binary codes by spectral graph partitioning. Additionally, PCAH [20] and its orthogonal projection variant [22] have been proposed for better quantization. The method ITQ exploits a simple and efficient alternating minimisation scheme to find a rotation of zero-centred data to minimise the quantization error of mapping this data to the vertexes of a zero-centred binary hypercube. Kulis and Grauman [43] proposed a kernelised version of LSH, namely KLSH, which also computes random projections, as standard LSH does, but these random projections are constructed using only the kernel function and a sparse set of examples from the database itself, which is different from standard LSH. To achieve less time complexity and better applicability, and also inspired by the spectral hashing method, AGH [21] was introduced to utilise Anchor Graphs to obtain tractable low-rank adjacency matrices and it can automatically discover the neighbourhood inherent structure in the training data. Heo *et al.* proposed a hyperplane-based hash function, called Spherical Hashing [44], which minimises the spherical distance between the original real-valued features and the learned binary codes, this distance metrics is also called spherical Hamming distance. KMH [42] uses an affinity-preserving K-means algorithm to simultaneously performs k-means clustering and learns the binary indices of the quantized cells and exploits the Hamming distance of the cell indices to approximate the distance between these cells.

For the second category, semi-supervised methods use information from both labelled training data and unlabelled training data. One of the typical methods of this category is Semi-supervised Hashing (SSH) [45], which minimises empirical error over the pairwise labelled data and maximises the variance over both labelled and unlabelled sets.

For the third category, the semantic label information of each sample is fully used in learning more efficient binary representations, thus better performance than the unsupervised and the semi-supervised methods can be achieved. For example, Binary Reconstructive Embedding (BRE) [46] utilises pairwise relations between data samples to minimise the reconstruction error between the original Euclidean space and the learned Hamming space. QBH [27] incorporates quantization error reduction methods into conventional property preserving hashing methods. Norouzi and Fleet [47] presented a hash function called Minimal Loss Hashing (MLH) to compute binary codes by minimising the empirical loss between the learned Hamming distance and the quantization error. ITQ also has its supervised version called CCA-ITQ [22], which utilises Canonical Correlation Analysis (CCA) with label information to reduce the dimension of original features while preserving the semantic information, and then exploits ITQ to minimise the quantization error. The Supervised Discrete Hashing (SDH) [11] compute the hash codes with the preservation of semantics with the intention for classification. Based on SDH, X. Shi *et al.* proposed to exploit kernel based method Kernel-based Supervised Discrete Hashing (KSDH) [48] to improve the performance. To solve the problem of NP-hard optimisation and computational complexity in graph based method, Asymmetric Discrete Graph Hashing (ADGH) [49] was proposed by preserving the asymmetric discrete constraint and building an asymmetric affinity matrix to learn compact binary codes. Additionally, SSDH [26] constructs hash functions as a latent layer in a deep network and the binary codes are learned by minimising an objective function defined over classification error.

A. Deep Hashing Methods

In learning based hashing methods, deep learning based methods form a special group in this area, so we discuss it here separately. To our knowledge, Semantic Hashing (SmH) [50] is the earliest effort to use deep network to learn hash function. It compute hash codes by passing an unlabelled image through a stacked Restricted Boltzmann Machines (RBMs).

The unsupervised deep hashing methods, Deep Hashing (DH) [39] and Deep compact binary descriptor (Deepbit) [40] also use deep network for learning compact binary code. DH develop a deep neural network to seek multiple hierarchical non-linear transforms to learn binary codes. In this method, DH employs three constraints as follow at the top layer of the deep network: 1) the loss between the original real-valued feature and the learned binary vector is minimised, 2) the binary codes distribute evenly on each bit, and 3) different bits are as independent as possible. DH also has its supervised version, which includes one discriminative term into objective function and simultaneously minimises the intra-class differences and maximises the inter-class difference of learned binary code. On the basis of DH, Deepbit adds another constraint, rotation invariant, to the final loss function, and achieve a slight boost on the performance. Do *et al.* [41] propose a Binary Deep Neural Network (BDNN), which has both unsupervised and supervised versions. BDNN introduces one hidden layer to directly output the binary codes and then utilises the binary code layer to reconstruct the original feature. Through optimising the sum of reconstruction loss, independence constraint loss, and balance constraint loss, BDNN can get a little improvement with respect to DH. The recent state-of-art method is Supervised Semantic-preserving Deep Hashing (SSDH) [26], which assumes that the semantic labels are governed by several latent attributes. Based on this assumption, SSDH constructs hash function as a latent layer in a deep network and learns the binary codes by minimising an objective function combined by classification error, independent property, and balance property like those in method BDNN.

III. METHODOLOGY

Let $\boldsymbol{X} = [\boldsymbol{x}_1; \ldots ; \boldsymbol{x}_n; \ldots ; \boldsymbol{x}_N] \in \mathsf{R}^{N \times w \times h}$ be the training set which contains N image samples, where $\boldsymbol{x}_n \in \mathsf{R}^{w \times h}$ is

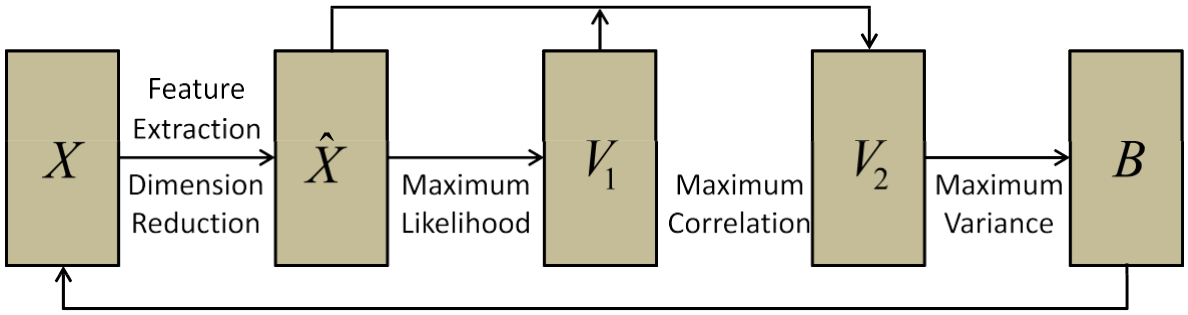


Fig. 2. Main Framework of Unsupervised hashing method with Pseudo Labels. The discovered pseudo label codes are used as supervision for fine-tuning the deep hashing network like existing supervised methods. The total procedure is executed for T times recurrently to find better network model.

the n-th image sample in $\boldsymbol{X}$, and has the dimension of $w \times h$. Hashing methods aim to learn a hash function $\boldsymbol{H}: \boldsymbol{X} \rightarrow \boldsymbol{B} \in \{-1, 1\}^{N \times k}$ to project each sample into a compact binary code $\boldsymbol{b}_n = \boldsymbol{H}(\boldsymbol{x}_n)$, where k is the target binary code length.

Figure 2 illustrates the framework of our method. The fundamental issue is how to obtain class labels during training. In supervised methods, the class labels are used to guide intra-class instances to be encoded closer in $\boldsymbol{B}$. For improved performance under unsupervised scenarios, this paper proposes to discover latent pseudo labels that can maximumly reflect the data distribution. Correspondingly, we adopt three of the most important criteria to align the latent labels multiple times. Firstly, the latent labels $\boldsymbol{V}_1 \in \mathbb{R}^{N \times k_0}$ maximise the likelihood to the visual feature $\hat{\boldsymbol{X}} \in \mathbb{R}^{N \times d}$, which is extracted from visual sample $\boldsymbol{X}$. However, in most scenarios, stabler performance often results from a smaller number of latent labels, *i.e.* $k_0 < k$, which may lead to severe information loss. Therefore, the next step discovers an adequate number of latent classes $\boldsymbol{V}_2 \in \mathbb{R}^{N \times k_1}$ that can preserve the maximum mutual information, where $k_1 = k$ is used as the length of final hash codes. Finally, the latent labels are converted into binary codes that can preserve the distribution of V_2 in the meantime maximising the overall variance so that the final hash codes are more discriminative. Hereafter, we can recurrently train deep hashing models using the discovered latent label codes like supervised methods.

A. Deep Hashing Model

Our deep hashing model is initialised by using the pre-trained VGG-16 parameters. Although the model is trained on the large-scale ImageNet dataset, the extracted visual features are not readily applied to learn hashing codes. Firstly, on a different dataset, some of the involved classes are not present in ImageNet, thus only using pre-trained model to extract features may degrade the performance. Secondly, the initial classification model may not perform the best for learning binary codes. Therefore, we consider to trim the model using the followed loss function:

$$J = \|\boldsymbol{H}(\boldsymbol{X}; \copyright) - \boldsymbol{B}\|_F^2, \quad \text{s.t. } \boldsymbol{B} \in \{-1, +1\}^{N \times k}, \qquad (1)$$

where © is the parameter space of the deep hashing model. Specifically, our model uses all the convolution layers *conv1_1–conv5_3*(each one followed by a *ReLU* layer), pooling layers and first two fully connected layers *fc6*, and *fc7*. The *ReLU* layer after *fc7* is removed. Then, the final *fc8* layer is substituted by the objective binary code followed by a *tanh* function to constrain the learnt codes to fall within $(-1, 1)$. In order to retain the excellent training model of ImageNet, we fix *conv1_1–conv5_3* and release *fc6–fc8* to be trainable. The detailed configuration of the network is illustrated in Fig. 3.

However, the fundamental difficulty is the lack of supervision. In other words, the vital objective $\boldsymbol{B}$ of the above learning function is missing. Random code allocation, *e.g.* [39] can make the resultant code completely lose the original data structure. Motivated by this, we investigate a unified framework to discover pseudo labels that can maximumly preserve the original distribution, which will be introduced next.

B. Pseudo Labels Discovery

Pseudo labels of latent classes can be achieved by linear or non-linear combinations of the real classes according to the intrinsic data distribution. Ideally, instances from the same class should be closer in the pseudo label space. However, without the guidance of true class labels, conventional unsupervised hashing methods often suffer from the severe discrepancy between the pseudo and real labels. One of the main reasons is that a preferred encoding procedure often requires ensuring different criteria at the same time, which makes the optimisation intractable to solve. In this paper, we propose a unified framework that sequentially discovers latent classes that are aligned with the visual data at multiple time. Each step corresponds to an optimisation based on a specific criterion.

1) Feature Extraction and Dimension Reduction: We first extract the visual features from training images using the *fc7* layers of the pre-trained Model. The dimensionality of visual features is 4096 that is much larger than the required length of binary codes. To enhance the discriminative property and reduce the computational cost, we first generate a compact feature where the variance of every variable is maximised, the variables are uncorrelated pair-wisely, and the redundancy and noise are removed. This can be realised by Principle Component Analysis (PCA) easily. However, in order to

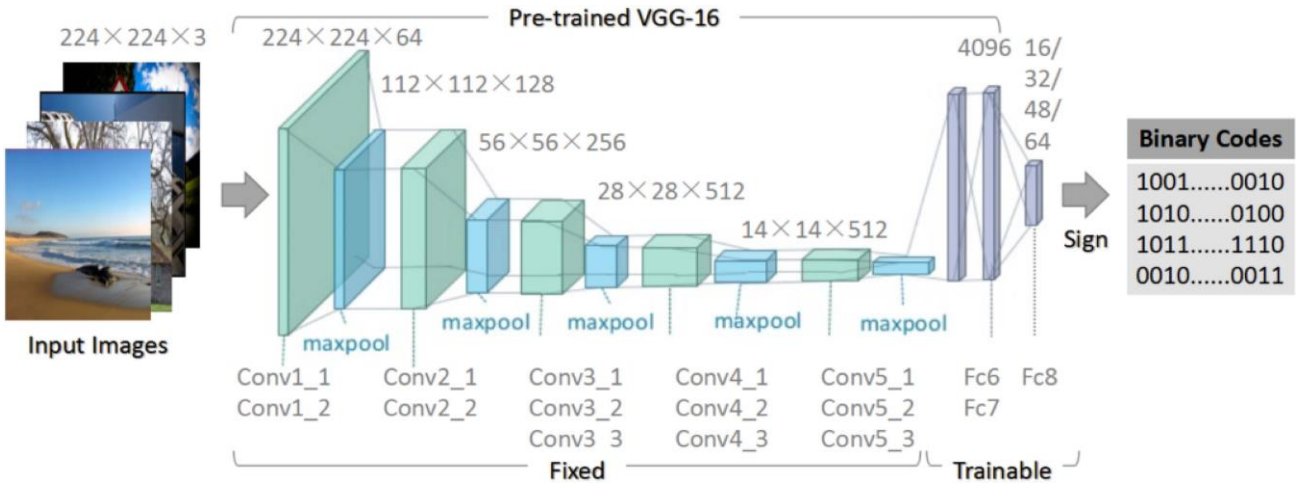


Fig. 3. Configuration of the deep hash model. The convolutional layers are fixed and the full connection layers are released to be trainable.

achieve the expected length of codes, conventional methods, such as PCA-Hashing [20], directly reduce the dimension from 4096 to binary code length k regardless the loss of important property-preserving information. In comparison, our scheme allows to retain as many PCA components as possible, and shorter codes can be obtained in later stages. Therefore, we use the percentage of retained variance as criterion like the most traditional usage of PCA. The resultant representation is the extracted visual feature $\hat{\boldsymbol{X}}$ of our framework.

2) Pseudo Labels With Maximum Likelihood: Due to the widely existing intra-class variance, a class label often corresponds to several different latent distributions. Also, some classes may share the same latent distribution. For example, a 'plane' and a 'bird' share the features of 'wings'. Therefore, the ideal number of latent labels may not always equal to that of real classes. A typical solution is to find a set of latent classes $\boldsymbol{Z}$ $\boldsymbol{z}_1$ $\rightleftharpoons$. [$\boldsymbol{z}_c$;.. $\boldsymbol{z}_{k_0}$ that can maximise the likelihood to the visual data distribution, *i.e.* to generate visual features using several latent distributions. Suppose there is a k_0 number of latent classes, where each visual instance belongs to one of the latent class. By assuming uniformed prior $\mathrm{p}(\boldsymbol{Z})$ $\perp$, taking the Bayes' rule, we have the likelihood function:

$$\mathrm{p}(\boldsymbol{Z}, \hat{\boldsymbol{X}}) = \mathrm{p}(\hat{\boldsymbol{X}}|\boldsymbol{Z}) = \sum_{n=1}^{N} \log \sum_{c=1}^{k_0} \mathrm{p}(\hat{\boldsymbol{x}}_n|\boldsymbol{z}_c). \tag{2}$$

Using the $\boldsymbol{Z}$ with the maximised likelihood, we can assign each of the visual features $\hat{\boldsymbol{X}}$ to a latent class in $\boldsymbol{Z}$ by maximising the marginal probability mass function: $L_1 = \mathrm{p}(\hat{\boldsymbol{X}}, \boldsymbol{Z}|\boldsymbol{V}_1)$, where $\boldsymbol{V}_1 = [v_n] \in [1, k_0]$ is the pseudo label space with k_0 latent classes. The each paired v_n of $\hat{\boldsymbol{x}}_n$ can be achieved by:

$$v_n = \arg \max_{\in[1,k_0} \mathrm{p}(\boldsymbol{z}_c|\boldsymbol{x}_n). \tag{3}$$

3) Maximum Correlation Label Embedding: The above $\boldsymbol{V}_1$ only contains generative distribution, *i.e.* shared similarities between real classes. In the second stage, we discover a latent label embedding space so that the single-valued pseudo label can be aligned to corresponding visual feature space to preserve both the discriminative and intrinsic data distribution. We first extend pseudo labels into one-hot vectors: $\hat{\boldsymbol{V}}_1 = [\hat{v}^i_n] \in \{0, 1\}^{N \times k_0}$, where $\hat{v}^i_n = 1$ for $i = v_n$ and $\hat{v}^i_n = 0$ otherwise. Given the visual features after PCA $\hat{\boldsymbol{X}} \in \mathrm{R}^{N \times d}$, we maximise the correlation function:

$$L_2 = \max_{W_1, W_2} Corr(\hat{\boldsymbol{X}}W_1, \hat{\boldsymbol{V}}_1 W_2) = \max_{W_1, W_2} \frac{< \hat{\boldsymbol{X}}W_1, \hat{\boldsymbol{V}}_1 W_2 >}{\|\hat{\boldsymbol{X}}W_1\| \cdot \|\hat{\boldsymbol{V}}_1 W_2\|}, \tag{4}$$

where $W_1 \in \mathrm{R}^{d \times k}$, $W_2 \in \mathrm{R}^{k_0 \times k}$ are the parameters for linear transformations. Hereby, the visual features can be converted into the embedded pseudo label space in the final desired dimensionality k:

$$\boldsymbol{V}_2 = \hat{\boldsymbol{X}}W_1. \tag{5}$$

4) Maximum Variance: In the last stage, we convert $\boldsymbol{V}_2$ into binary codes. In order to minimise the quantisation error $\|sgn(\boldsymbol{v}) - \boldsymbol{v}\|_F^2$ between generated binary codes $\boldsymbol{B}$ and $\boldsymbol{V}_2$, we adopt the ITQ algorithm to find a rotation matrix $\boldsymbol{R} \in \mathrm{R}^{k \times k}$ that can align the binary codes to the nearest vertex of the hypercube:

$$L_3 = \|\boldsymbol{B} - \boldsymbol{V}_2 \boldsymbol{R}\|_F^2, \quad \text{s.t. } \boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I}, \tag{6}$$

where, $\|\cdot\|_F$ denotes the Frobenius norm; $\boldsymbol{V}_2$ is zero-centred; $\boldsymbol{I}$ is the identity matrix, *i.e.* $\boldsymbol{I}_{i,j} = 1$ if and only if $i = j$. Considering all of the three losses, Eq. 1 can be modified into the following objective function:

$$\min_{©, \boldsymbol{B}, \boldsymbol{R}, W_1} \|H(\boldsymbol{X}; ©) - \boldsymbol{B}\|_F^2 + \|\boldsymbol{B} - \hat{\boldsymbol{X}}W_1\boldsymbol{R}\|_F^2,$$
$$\text{s.t. } \boldsymbol{B} \in \{-1, +1\}^{N \times k}, \quad \boldsymbol{R}\boldsymbol{R}^\top = \boldsymbol{I}. \tag{7}$$

C. Optimisation

The above objective function has four variables ©, $\boldsymbol{B}$, $\boldsymbol{R}$, W_1, and the function is a non-convex problem with the binary and orthogonal constraints. As we have known that there is no direct method to solve the problem of Eq. 7.

Therefore, for each loop, we propose an alternating strategy that in turn optimises W_1, interactively updates $\boldsymbol{B}$ and $\boldsymbol{R}$, and finally optimise ©. The overall optimisation usually takes a few iterations for a specific bit size which we summarise as follows.

. W_1 **Step**: By first fix the parameters of the pre-trained deep model ©, we can discover latent classes that can maximise the likelihood of the extracted visual features in Eq. 2. In this paper, we simply adopt the K -means algorithm as the generative model to solve the equation. This can be simply implemented by finding k_0 centroids $\boldsymbol{Z} = [\boldsymbol{z}_1, \ldots, \boldsymbol{z}_c, \ldots, \boldsymbol{z}_{k_0}] \in \mathsf{R}^{k_0 \times d}$. Pseudo labels $\boldsymbol{V}_1$ then can be assigned by:

$$v_n = \arg \min_{c \in [1, k_0]} \|\hat{\boldsymbol{x}}_n - \boldsymbol{z}_c\|_2^2. \tag{8}$$

After extending $\boldsymbol{V}_1$ to one-hot vectors $\hat{\boldsymbol{V}}_1 = [\hat{\boldsymbol{v}}^i]_n \in \{0, 1\}^{N \times k_0}$, we can estimate its correlation to the original extracted features $\hat{\boldsymbol{X}}$ in Eq. 4. A kernel matrix can be used to estimate the correlations between each pair of dimensions in $\hat{\boldsymbol{V}}_1$ and $\hat{\boldsymbol{X}}$. This paper adopts the simplest linear kernel in Eq. 4 that can result in the following covariance matrix before the linear transformation W_1 and W_2:

$$K(\hat{\boldsymbol{X}}, \hat{\boldsymbol{V}}_1) = \mathsf{E}([\hat{\boldsymbol{X}}, \hat{\boldsymbol{V}}_1]^\mathsf{T} [\hat{\boldsymbol{X}}, \hat{\boldsymbol{V}}_1]) = \begin{matrix} K_{xx}, & K_{xv} \\ K_{vx}, & K_{vv} \end{matrix}, \tag{9}$$

where $E(\cdot)$ denotes a normalisation using the empirical expectation. K_{xx}, K_{vv} are inner-space correlations and K_{xv}, K_{vx} are inter-space. We leave the problem as a generalised function so that existing kernel methods can be directly applied. In this paper, we simply use the linear kernel with an analytic solution to achieve W_1:

$$K_{xy} K_{yy}^{-1} K_{yx} W_1 = \lambda^2 K_{xx} W_1, \tag{10}$$

which is an eigenproblem that can be efficiently addressed by Matlab toolbox *eig(.)*, where λ is the eigenvalue corresponding to W_1.

. $\boldsymbol{BR}$ **Step:** By fixing ©, W_1, we interactively solve $\boldsymbol{B}$ and $\boldsymbol{R}$ in a inner-loop, which is same as that did in ITQ [22]. Initialise $\boldsymbol{R}$ as a random orthogonal matrix, and the Eq. (7) can be easily solved as,

$$\boldsymbol{B} = sgn(H(\boldsymbol{X}; ©) + \hat{\boldsymbol{X}} W_1 \boldsymbol{R}), \tag{11}$$

where, $sgn(x) = 1$ if $x > 0$ and -1 otherwise. With fixed $\boldsymbol{B}$, we can efficiently optimise $\boldsymbol{R}$ using singular value decomposition (SVD):

$$\begin{aligned} \boldsymbol{B}^\mathsf{T} \hat{\boldsymbol{X}} W_1 &= \boldsymbol{S} O \tilde{\boldsymbol{S}}^\mathsf{T} \\ \boldsymbol{R} &= \tilde{\boldsymbol{S}} \boldsymbol{S}^\mathsf{T}, \end{aligned} \tag{12}$$

where $\boldsymbol{S} \in \mathsf{R}^{k \times k}$ and $\tilde{\boldsymbol{S}} \in \mathsf{R}^{k \times k}$ are two unitary matrices, and O is a rectangular diagonal matrix with non-negative real numbers on the diagonal. We alternating the above $\boldsymbol{B}$ and $\boldsymbol{R}$ steps iteratively until convergence. An example of the objective function convergence is shown in Fig. 4. Hereafter, we can finally get the final binary code $\boldsymbol{B}$ as the supervision for the deep hashing model.

. © **Step:** Using the $\boldsymbol{B}$ obtained from the above three stages, we solve the minimisation problem in Eq. 1 by fine-tuning the deep network with Adam optimization [51], where the hashing loss is minimised with mini-batch back-propagation. The network parameter © is updated simultaneously and determined in the current loop after convergence. Then, the network model can generate new image features for the next loop. Those parameters are updated sequentially for T loops and finally the hash model can be constructed.

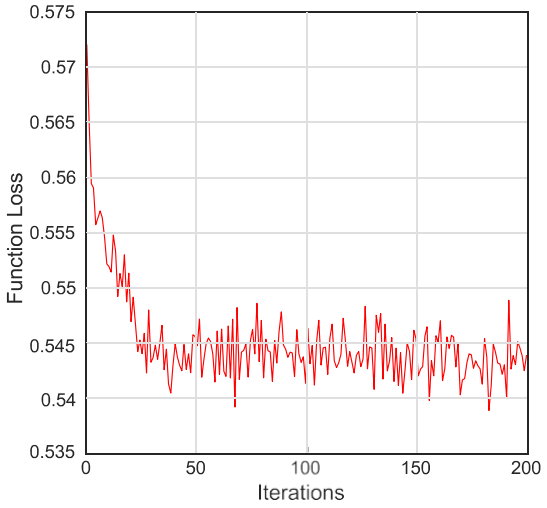


Fig. 4. An example of the objective function convergence on the CIFAR-10 dataset.

D. Scalable Image Retrieval

Using the achieved hashing model, images in the gallery can be efficiently converted into binary codes by $sgn(H(x; ©))$. Given a query image $\tilde{x}$, we can carry out the retrieval by ranking its Hamming distances to the images in the gallery by $\arg\min_i \|sgn(H(x_i; ©)) - sgn(H(\tilde{x}; ©))\|_F^2$. The overall method is summarized in Algorithm(1).

IV. Experiments

We provide a comprehensive comparison with various baselines and state-of-the-art methods on three challenging datasets, the CIFAR-10 color images [52], the NUS-WIDE dataset [53], and the MIRFLICKR-25K [54] dataset.

A. Datasets

1) CIFAR-10 Dataset: [52] contains 10 classes and each class consists of 6,000 images, leading to a total of 60,000 images. And the dataset is split into two sets, training set and testing set, with 50,000 and 10,000 images respectively. We randomly select 100 images from each class, totally 1,000 images, as the testing set, and set the rest 59,000 images as the training set.

2) NUS-WIDE Dataset: [53] is a real-world web image dataset, which contains about 270K web images. It is a multi-label dataset, in which each image is associated with one or multiple class labels from 81 classes, the number of images in each class ranges from 5,000 from 30,000. Because the dataset was download from the Internet, and some of its URLs are

Algorithm 1 Unsupervised Deep Hashing With Pseudo Labels

Input:

Training image set $\boldsymbol{X}$,Hyper-parameters: k, k_0, T, Initialised $\boldsymbol{R}$;
Pre-trained net model inlcuding initialized Θ;

Output:

Network parameters Θ.

1: **for** each $i \in [1, T]$ **do**
2: Feature Extraction and Dimension Reduction:
Extract *fc7* feature from $\boldsymbol{X}$ using net parameters Θ,
Reduce dimension of *fc7* feature from 4096 to d with PCA, and get $\hat{\boldsymbol{X}}$;
3: Maximum Likelihood:
Latent class discovery using Eq. 2;
Compute $\boldsymbol{V}_1$ using Eq. 8;
4: Maximum Correlation:
Compute W_1 using Eq. 9, 10;
Compute $\boldsymbol{V}_2$ using Eq. 5;
5: Maximum Variance:
Repeat until converge
Fix $\boldsymbol{R}$, update $\boldsymbol{B}$ using Eq. 11;
Fix $\boldsymbol{B}$, update $\boldsymbol{R}$ using Eq. 12;
6: Deep Hashing:
Update the parameters of *fc6-fc8* Θ using Eq. 1;
7: **end for**
8: **return** Θ;

invalid, furthermore, we use the settings as in [57], the most frequent 21 concept classes, we can get 159,579 images finally. We randomly select 2,100 images from the dataset as the test set, and the remaining images are left as the training set.

3) MIRFLICKR-25K Dataset: [54] contains 25,000 images with 38 categories as well as their tags collected from Flickr. Each of images is annotated with more than one label. We randomly select 1,000 images from the dataset as the test set, and the left 24,000 images were kept as the training set.

B. Experimental Settings

1) Baselines: We systematically compare our method with eight state-of-the-art non-deep methods: ITQ [22], PCAH [20], LSH [13], DSH [55], SpH [44], SH [19], AGH [21], and SELVE [56], and three deep methods: DH [39], Deepbit [40], UH-BDNN [41] for retrieval task, all these eleven methods are unsupervised. All of the non-deep methods and UH-BDNN [41] in our experiments use the same VGG [58] *fc7* feature as that in our method, and DH [39] and Deepbit [40] are based the same settings like that in their original papers.

2) Key Hyper-Parameters: The energy ratio of PCA is set as 0.98, and the number of clusters k_0 is set to 10 for dataset CIFAR-10 and 5 for dataset MIRFLICKR and NUS-WIDE in main comparisons. For deep training process of our method, we set the mini-training batches as 20, learning rate as 2×10^{-6}. The iteration time of solving maximum variance method is set as 50, an example of convergence is illustrated in Fig. 4, same as the original definition by Gong *et al.* [22]. The iteration time of our deep training in each loop was set to 1×10^5, and the total loop times T is set to 10.

C. Main Comparisons With State-of-the-Art Methods

In this section, we evaluate our method by four common metrics and provide comprehensive comparison to the eleven state-of-the-art baselines: 1) mean average precision(mAP), which evaluates the overall performance of hashing methods; 2) precision-recall curve, which describes the relationship between retrieval precision and recall rate; 3) precision at N retrieved samples, which means the percentage of ground truth images among top N retrieved samples; and 4) recall rate at N retrieved samples, which measures the percentage of ground truth images in retrieved images among all ground truth images in the query dataset. In this experiment, we use the hashing toolbox supplied by Yuan *et al.* [59] and Lu *et al.* [60] to compute the mAP value and draw P-R, P-N, and R-N curves, and give the detailed analysis about the results in the following subsections.

1) Mean Average Precision: In the query phase, we compute the mean average precision (mAP) to evaluate the retrieval performance of our algorithm on all three datasets. The mAP is defined as:

$$mAP = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \left(\frac{1}{r} \sum_{j=1}^{r} P(i, j) \right) \tag{13}$$

where, $|Q|$ is the size of the query image set, r is the number of retrieved images from the dataset related to the ith query image, and $P(i, j)$ is the precision of the top jth retrieved image of ith query image. In addition, all of the performances of the baseline and our method are evaluated on four different length of binary code $\{16, 32, 48, 64\}$.

Table I shows our results of mAP on all of the three datasets. Our method significantly outperforms all of the compared state-of-the-art methods, which manifest the robustness of our hash code. The success can be understood by comparing to the results of three methods in the table. First is the power of deep model. Compared to LSH, the deep model UH-BDNN achieved 5-8% improvement. However, due its model is based on an auto-encoder-like scheme, the learnt hash code is not structural-aware enough. In comparison, the AGH preserves data structure by the graph-regularization and achieves better performance than that of UH-BDNN. The second best method is ITQ, which performs stabler than AGH on long codes, such as 48 and 64-bits. This is because the maximised variance in ITQ benefits the exploration of a higher-dimensional Hamming space, which makes the safety-bounds between classes wider. Our framework effectively absorbs the advantages of all the above mentioned methods. Therefore, both of our shorter and longer hashing codes achieve the best performance.

From Table 1, we observe that our method is not very sensitive to the length of the code. From 16 bits to 64 bits, the performances just vary marginally. Another observation is that longer code may perform slightly worse, especially for the dataset NUS-WIDE. The potential reason is related to the number of real classes and latent classes. Since the NUS-WIDE is a multi-label dataset, and the number of latent classes

TABLE I

IMAGE RETRIEVAL RESULTS (MEAN AVERAGE PRECISION(mAP)) WITH 16 BITS, 32 BITS, 48 BITS AND 64 BITS ON THE CIFAR-10, NUS-WIDE AND MIRFLICKR-25K DATASETS. THE SCALES OF TEST SET ARE 1K, 2.1K, 1K RESPECTIVELY, AND THE mAPs ARE COMPUTED USING ALL THE TRAINING SETS. THE PROPOSED METHOD OUTPERFORMS ALL THE STATE-OF-THE-ART METHODS LISTED IN THIS TABLE

Algorithm	CIFAR-10				NUS-WIDE				MIRFLICKR-25K			
	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits	16bits	32bits	48bits	64bits
ITQ [22]	0.3115	0.3286	0.3387	0.3476	0.5117	0.5165	0.5183	0.5245	0.6357	0.6677	0.6636	0.6657
PCAH [20]	0.2121	0.1863	0.1749	0.1687	0.4102	0.3929	0.3826	0.3763	0.6021	0.6015	0.5932	0.5882
LSH [13]	0.1798	0.2142	0.2170	0.2405	0.4102	0.4067	0.4242	0.4389	0.5724	0.5861	0.5972	0.6043
DSH [55]	0.2466	0.2629	0.2833	0.2923	0.5012	0.4910	0.4982	0.5180	0.6270	0.6816	0.6699	0.6775
SpH [44]	0.2041	0.2374	0.2667	0.2591	0.4180	0.4558	0.4735	0.4735	0.5972	0.6292	0.6376	0.6438
SH [19]	0.1828	0.1818	0.1728	0.1664	0.3459	0.3579	0.3606	0.3650	0.5954	0.5940	0.5919	0.5911
SELVE [56]	0.3090	0.2807	0.2606	0.2391	0.4672	0.4621	0.4442	0.4323	0.6411	0.6414	0.6340	0.6171
AGH [21]	0.3013	0.2698	0.2539	0.2384	0.5649	0.5218	0.5182	0.4782	0.6681	0.6436	0.6323	0.6221
DH [39]	0.1937	0.1939	0.1929	0.1883	0.3855	0.3926	0.3833	0.3767	0.5750	0.5847	0.5984	0.5881
UH-BDNN [41]	0.2643	0.2815	0.2884	0.2947	0.4708	0.4725	0.4778	0.4811	0.6489	0.6442	0.6493	0.6522
Deepbit [40]	0.2076	0.2093	0.2282	0.2445	0.4099	0.4091	0.4398	0.4608	0.6182	0.6015	0.5931	0.6028
Our Method	**0.3512**	**0.3791**	**0.3901**	**0.3841**	**0.5738**	**0.5623**	**0.5260**	**0.5380**	**0.6711**	**0.6872**	**0.6885**	**0.6797**

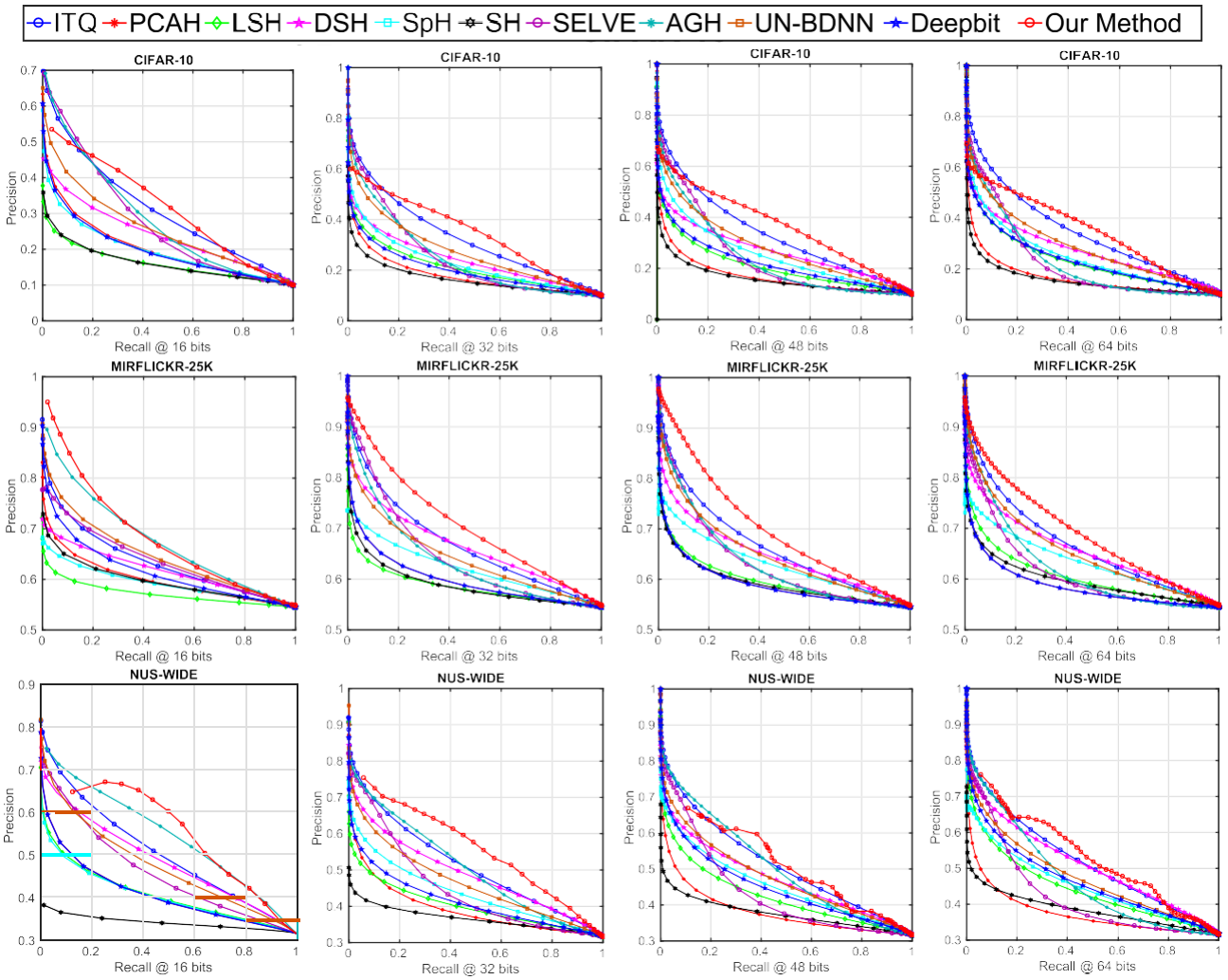


Fig. 5. Results of Precision VS Recall Curves on all of the three datasets.

is fixed to be 5 (the reason why we choose 5 latent classes will be analysed in the following subsection), longer codes make the intermediate representation non-compact and result in redundant dimensions, which is same as that in PCAH [20]. In addition, this phenomenon of mAP reduction with code length growing also appears in AGH [21] and SELVE [56].

2) Precision-Recall: Another popular evaluation protocol is the precision-recall (PR) curve which plots the precision and recall rates at different searching Hamming radius $r = 0, 1, 2, \ldots, k$, where k is the length of the binary codes. Fig. 5 shows the precision-recall curve of each method with 16, 32, 48, and 64 bits. The first point of the curve stands

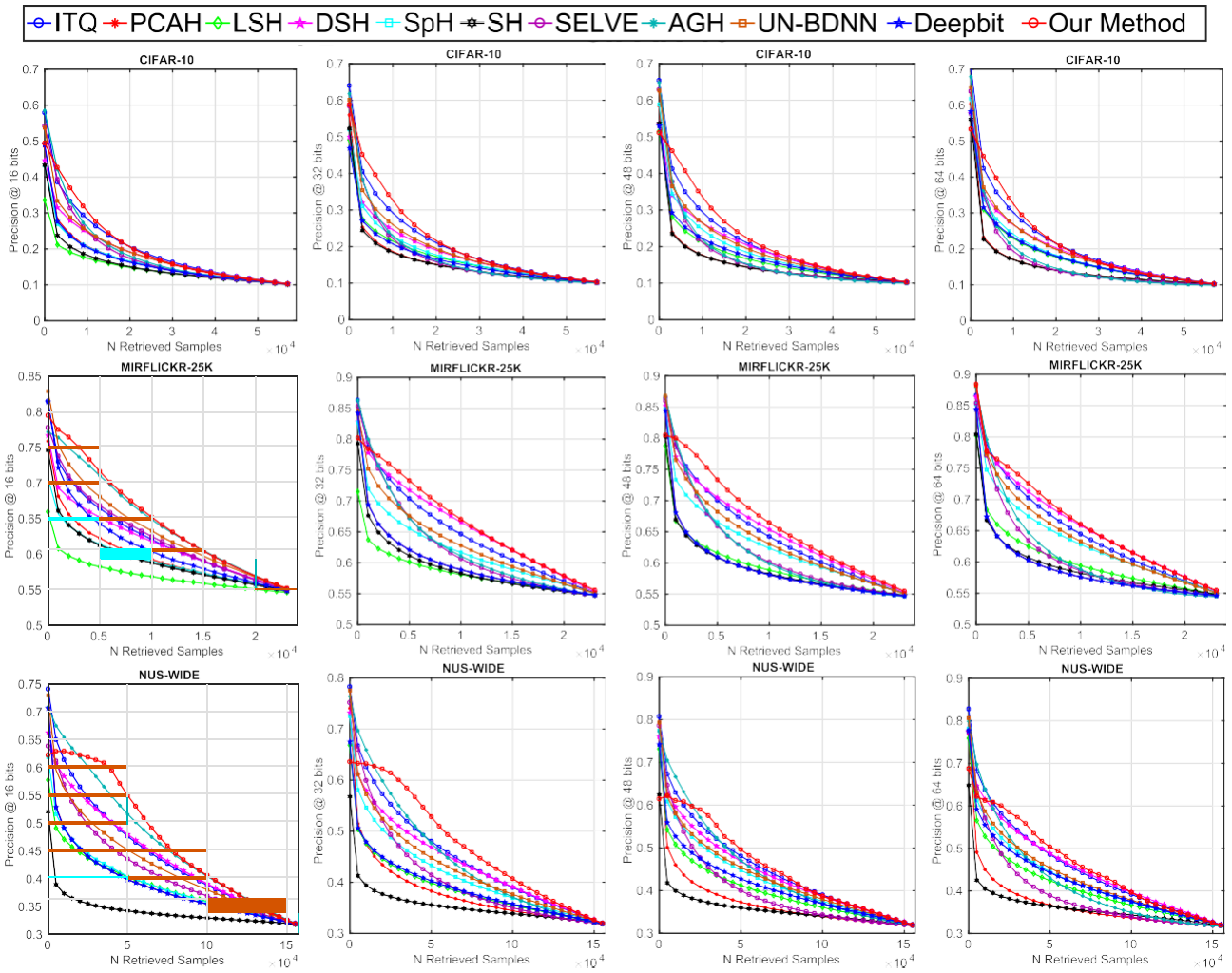


Fig. 6. Results of Precision @ N Retrieved images on all of the three datasets.

for the precision and recall rate at Hamming radius ~~≠~~ 0; the second point means the precision and recall rate at Hamming radius *r* ~~1,~~ and so on.

It is noticeable that the points at different searching Hamming radius are not aligned. Most methods cannot make similar objects are encoded exactly the same, which leads to very low recall rate but high precision with short Hamming radius *r* ~~0~~, 1, 2. In contrast, albeit slightly lower precision, our method can often achieve significantly higher recall rates although the Hamming radius is short, *e.g.* CIFAR-10 @ 16bits, and all of the PR curves on NUS-WIDE. On the MIRFLICKR-25K dataset, our method outperforms all the other methods on all ~~{~~16, 32, 48, 6~~4~~ bits length at every Hamming radius, in terms of both precision and recall rates.

3) Precision @ N Retrieved Samples: The next two evaluations separately observe precision and recall rates. Firstly, we fix the number of retrieved sample and testifies the corresponding precision. In Fig. 6, the curves were drawn with 3,000, 1,000, 5,000 retrieved images per dot on the datasets of CIFAR-10, IMRFlICKR-25K, NUS-WIDE, respectively. For example, on CIFAR-10, the first point shows the precision when only one image is returned and the second point stands for the precision when 3001 images are returned. Our methods achieve significantly higher precisions than baselines on CIFAR-10. Only at the point of at 1000 returned images, the precision of our method is imperceptibly lower than that of ITQ, SELVE, and AGH for the code length of 32 and 64 bits. The result curves on the MIRFLICKER-25K dataset show that our method can outperform all the current state-of-the-art algorithms listed in table (I) for the code length of 16 and 48 bits. When the code lengths are 32 and 64 bits, our method performs just a little worse than AGH, ITQ and SELVE for the number of returned images less than 1,000, otherwise the performance of our method can surpass that of all the other methods. The result curves on the NUS-WIDE dataset demonstrate that our algorithm can get the better precision from the returned images of 20,000, 15,000, 20,000, 15,000 on the code length of 16, 32, 48, 64 respectively. Although some sacrifice of the precession of that when the number of returned samples are small and our method performs slightly worse than the state-of-art algorithms, such as ITQ and AGH, our method can exceed all the listed methods when the number of retrieved images is larger than 20,000.

4) Recall @ N Retrieved Samples: The points of recall curves are in pairs with that of the precision curves, with each step of 3,000, 1,000, and 5,000 on the three datasets, respectively, which are shown in Fig. 7. On CIFAR-10 dataset, for the code length of 48, our algorithm outperforms all other

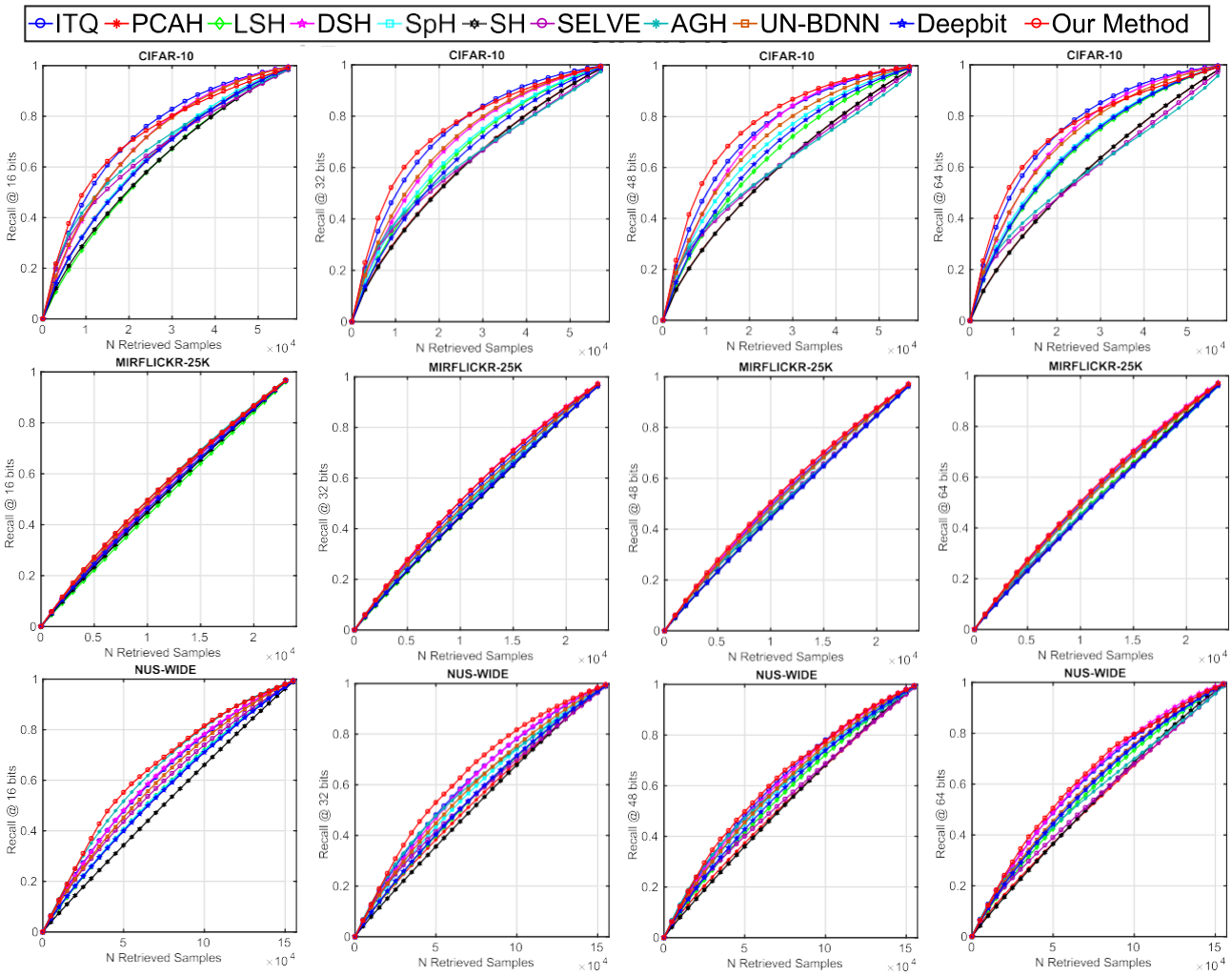


Fig. 7. Results of Recall rate @ N retrieved images on all of the three datasets.

algorithms at any number of retrieved images. For the code length of 16, 32, and 64, the recall rate of our algorithm exceeds the recall rate of others when the retrieved images were less than 20,000, which indicates that our algorithm can get more true-positive images from the earlier returned images. On MIRFLICKER-25K dataset, the recall rates of all of the methods perform very close. Therefore, our highest precision rates discussed earlier can manifest that our method achieves the best performance. For the 32 bits code length on the dataset of NUS-WIDE, the recall rate of our method can exceed all other compared algorithms at any number of returned images. When the retrieved images are less than 120,000 using 16 bits code, less than 90,000 using 48 bits code, and less than 100,000 using 64 bits code, our recall rates are significantly higher than that of other start-of-art methods. In short, our method achieves higher recall rates with fewer returned images than that of baselines, and we believe that the main reason of such good performance of our method is due to the ability of capturing the real distribution of the data.

D. Detailed Analysis

To further understand the success of our method, we analyse three technical issues. First, we discuss a key parameter of our method, *i.e.* how does the number of latent classes affect the final performance, and then we exhibit some retrieved images using our method and other baselines. Last but not least, we give the performances of pre-trained model to show the effectiveness of our method.

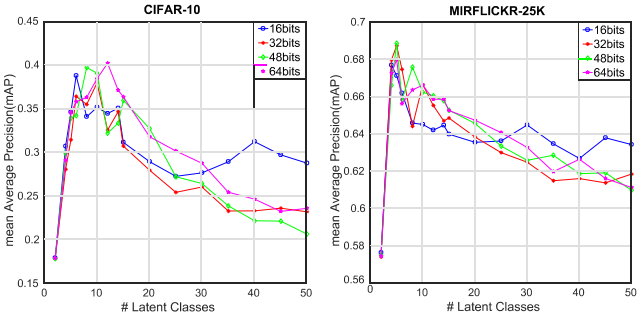


Fig. 8. Retrieval performance with respect to the number of discovered latent classes.

1) Number of Latent Classes: The fundamental technique of our framework is to find a number of latent classes so that the deep model can be trained like a supervised method. Therefore, it is important to investigate how many latent classes is the best for learning the hash code. In Fig. 8,

Fig. 9. Qualitative results on CIFAR-10 Dataset with queried images of *aeroplane* and *bird*. Returned samples with red boxes are false-positive.

we show the mAP with respect to the selected k_0 latent classes on CIFAR-10 and MIRFLICKR-25K datasets, which are single-label and multi-label retrieval tasks, respectively. In this experiment, we choose 16 different latent classes k_0 2, 4, 5, 6, 8, 10, 12, 14, 15, 20, 25, 30, 35, 40, 45, 50 to test the performance.

For the dataset CIFAR-10, the best results 0.3873, 0.3791, 0.3966 and 0.4015 appear at the latent class numbers of k_0 6, 10, 8, 12 of 16 bits, 32 bits, 48 bits, and 64 bits, respectively, and for the dataset MIRFLICKR-25K, the best results are 0.6768, 0.6872, 0.6885 and 0.6797, which appear at the latent class numbers of k_0 4, 5, 5, 5 respectively. For the fair of comparison, we just choose k_0 = 10 for CIFAR-10 and k_0 =5 for MIRFLICKR-25K to compute the final results in table I. And for the dataset NUS-WIDE, we also use the same parameter k_0 5.

The best k_0 of CIFAR-10 is around 10, which is interestingly the same as the total number of the classes. In comparison, although the total number of classes in MIRFLICKR is much higher than that of CIFAR-10 (38 vs 10), the multi-label retrieval task only requires at least one of the queried labels to be matched. Therefore, the performance results from a smaller number of latent classes, in which correlated real labels are clustered together so that the learnt hash codes can be very close in Hamming distance.

2) Qualitative Results: In Fig. 9, we select two query images from the CIFAR-10 dataset as examples and exhibit top ten retrieved images by different methods. It can be seen that the proposed method can achieve more satisfying results than other methods. Among the retrieved images, our method just gets only one false-positive image for class *bird* and zero failure for class *car*. Due to the *frog* is standing by the water, the reflection and the frog look like a *bird*. Such a failure case also supports that our resultant binary code can capture high-level visual features. In comparison, the reason behind the failure cases of other methods are easier to be found out, *e.g. car* is a false-positive of *aeroplane* because they have the similar shape feature, and *aeroplane* might have same color with *bird*. Our method utilises maximum likelihood in iterative optimisation to capture the distribution of data, which must be the reason why it can outperform other methods.

3) Performance of Pre-Trained Model: In conventional hashing methods, pre-trained VGG model is usually exploited to extract data features, which is used as the input for image retrieval, it is efficient and effective, but the existing pre-trained VGG model was trained on the ImageNet, which contains only 1,000 categories of images, and sometimes the retrieval datasets have different classes from the ImageNet, *e.g.* CIFAR-10 have a class called *bird*, but the ImageNet only have a class namely *indigo bird*, which is a special case of *bird*. MIRFLICKR-25K and NUS-WIDE also have some classes of scene that the ImageNet does not have. Therefore, If we only use the extracted features from $fc7$ layer with the pre-trained VGG model, that will lead to the degradation problem. In our method, based on the pre-trained VGG model, we fix $conv1-conv5$ and release $fc6-fc7$, which is utilised

TABLE II

IMAGE RETRIEVAL RESULTS (mAP) ON THREE POPULAR DATASETS WHEN USING THE PRE-TRAINED VGG MODEL

Code Length	CIFAR-10	MIRFLICKR-25K	NUS-WIDE
16 bits	0.3427	0.6667	0.5403
32 bits	0.3487	0.6794	0.5492
48 bits	0.3735	0.6774	0.5209
64 bits	0.3710	0.6744	0.5260

to fine-tune the network and solve the degradation problem. But if we fixed the f_c6– f_c7, the same as the pre-trained VGG model, which will result in lower mAPs on all the three datasets, which are listed in table (II).

V. CONCLUSION

In this paper, we have proposed a novel unsupervised deep hashing method with pseudo labels to address the existing problem in scalable image retrieval. We integrate maximum likelihood, maximum correlation, and maximum variance together to find pseudo labels, and use these pseudo labels as supervising information to train a deep network to solve the unsupervised hashing problem. Since our method combines all the advantages of likelihood, correlation, quantization and deep features, it achieves better performance compared to other state-of-the-art approaches, and the effectiveness of proposed framework is shown on three popular image retrieval datasets.

REFERENCES

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

[2] Y. Long, L. Liu, F. Shen, L. Shao, and X. Li, "Zero-shot learning using synthesised unseen visual data with diffusion regularisation," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2017.2762295.

[3] J. Yu, D. Tao, M. Wang, and Y. Rui, "Learning to rank using user clicks and visual features for image retrieval," *IEEE Trans. Cybern.*, vol. 45, no. 4, pp. 767–779, Apr. 2015.

[4] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, "A fast optimization method for general binary code learning," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5610–5621, Dec. 2016.

[5] Y. Luo, Y. Yang, F. Shen, Z. Huang, P. Zhou, and H. T. Shen, "Robust discrete code modeling for supervised hashing," *Pattern Recognit.*, vol. 75, pp. 128–135, Mar. 2018.

[6] L. Liu, M. Yu, and L. Shao, "Learning short binary codes for large-scale image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1289–1299, Mar. 2017.

[7] X. Liu, L. Huang, C. Deng, B. Lang, and D. Tao, "Query-adaptive hash code ranking for large-scale multi-view visual search," *IEEE Trans. Image Process.*, vol. 25, no. 10, pp. 4514–4524, Oct. 2016.

[8] X. Liu, J. He, and S.-F. Chang, "Hash bit selection for nearest neighbor search," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5367–5380, Nov. 2017.

[9] X. Shi, F. Xing, K. D. Xu, Y. Xie, H. Su, and L. Yang, "Supervised graph hashing for histopathology image retrieval and classification," *Med. Image Anal.*, vol. 42, pp. 117–128, Dec. 2017.

[10] Y. Long and L. Shao, "Describing unseen classes by exemplars: Zero-shot learning using grouped simile ensemble," in *Proc. IEEE Winter Conf. Appl. Compt. Vis.*, Mar. 2017, pp. 907–915.

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

[12] X. Liu, Z. Li, C. Deng, and D. Tao, "Distributed adaptive binary quantization for fast nearest neighbor search," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5324–5336, Nov. 2017.

[13] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proc. Annu. ACM Symp. Theory Comput.*, 2002, pp. 380–388.

[14] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li, "Query-adaptive reciprocal hash tables for nearest neighbor search," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 907–919, Feb. 2016.

[15] Z. Lin, G. Ding, J. Han, and J. Wang, "Cross-view retrieval via probability-based semantics-preserving hashing," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4342–4355, Dec. 2017.

[16] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, "Structure sensitive hashing with adaptive product quantization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2252–2264, Oct. 2016.

[17] A. Joly and O. Buisson, "Random maximum margin hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2011, pp. 873–880.

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

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

[20] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.

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

[22] 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.

[23] J. He, W. Liu, and S.-F. Chang, "Scalable similarity search with optimized kernel hashing," in *Proc. Int. Conf. Knowl. Discovery Data Mining*, 2010, pp. 1129–1138.

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

[25] L. van der Maaten and G. Hinton, "Visualizing data using t-SNE," *J. Mach. Learn. Res.*, vol. 9, pp. 2579–2605, Nov. 2008.

[26] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published, doi: 10.1109/TPAMI.2017.2666812.

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

[28] L. Liu, L. Shao, F. Shen, and M. Yu, "Discretely coding semantic rank orders for supervised image hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 1425–1434.

[29] M. A. Carreira-Perpinan and R. Raziperchikolaei, "Hashing with binary autoencoders," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 557–566.

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

[31] D. Zhang, D. Meng, and J. Han, "Co-saliency detection via a self-paced multiple-instance learning framework," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 5, pp. 865–878, May 2017.

[32] J. Han, H. Chen, N. Liu, C. Yan, and X. Li, "CNNs-based RGB-D saliency detection via cross-view transfer and multiview fusion," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2017.2761775.

[33] D. Zhang, J. Han, L. Jiang, S. Ye, and X. Chang, "Revealing event saliency in unconstrained video collection," *IEEE Trans. Image Process.*, vol. 26, no. 4, pp. 1746–1758, Apr. 2017.

[34] X. Yao, J. Han, D. Zhang, and F. Nie, "Revisiting co-saliency detection: A novel approach based on two-stage multi-view spectral rotation co-clustering," *IEEE Trans. Image Process.*, vol. 26, no. 7, pp. 3196–3209, Jul. 2017.

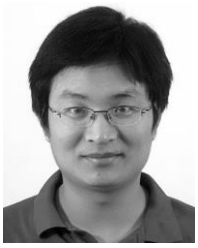
[35] L. Liu, F. Shen, Y. Shen, X. Liu, and L. Shao, "Deep sketch hashing: Fast free-hand sketch-based image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 2298–2307.

[36] Y. Shen, L. Liu, L. Shao, and J. Song, "Deep binaries: Encoding semantic-rich cues for efficient textual-visual cross retrieval," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 4097–4106.

[37] Y. Shen, L. Liu, and L. Shao, "Unsupervised deep generative hashing," in *Proc. Brit. Mach. Vis. Conf.*, 2017, pp. 1–11.

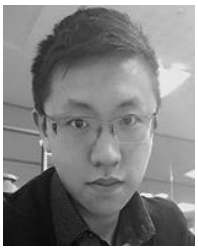
[38] Y. Long, L. Liu, L. Shao, F. Shen, G. Ding, and J. Han, "From zero-shot learning to conventional supervised classification: Unseen visual data synthesis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1627–1636.

[39] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 2475–2483.
[40] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2016, pp. 1183–1192.
[41] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 219–234.
[42] 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 Recognit.*, Jun. 2013, pp. 2938–2945.
[43] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2012.
[44] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2957–2964.
[45] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for scalable image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2010, pp. 3424–3431.
[46] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1042–1050.
[47] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 353–360.
[48] X. Shi, F. Xing, J. Cai, Z. Zhang, Y. Xie, and L. Yang, "Kernel-based supervised discrete hashing for image retrieval," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 419–433.
[49] X. Shi, F. Xing, K. Xu, M. Sapkota, and L. Yang, "Asymmetric discrete graph hashing," in *Proc. AAAI Conf. Artif. Intell.*, 2017, pp. 2541–2547.
[50] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reason.*, vol. 50, no. 7, pp. 969–978, 2009.
[51] D. P. Kingma and J. L. Ba, "ADAM: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–15.
[52] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf
[53] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "Nus-wide: A real-world Web image database from National University of Singapore," in *Proc. Int. Conf. Image Video Retr.*, 2009, Art. no. 48.
[54] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *Proc. Int. Conf. Multimedia Inf. Retr.*, 2008, pp. 39–43.
[55] Z. Jin, C. Li, Y. Lin, and D. Cai, "Density sensitive hashing," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1362–1371, Aug. 2014.
[56] X. Zhu, L. Zhang, and Z. Huang, "A sparse embedding and least variance encoding approach to hashing," *IEEE Trans. Image Process.*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014.
[57] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. Int. Joint Conf. Artif. Intell.*, 2016, pp. 1711–1717.
[58] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–29.
[59] Y. Yuan, X. Lu, and X. Li, "Learning hash functions using sparse reconstruction," in *Proc. Int. Conf. Internet Multimedia Comput. Serv.*, 2014, pp. 14–18.
[60] X. Lu, X. Zheng, and X. Li, "Latent semantic minimal hashing for image retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 355–368, Jan. 2017.

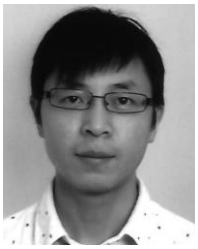


Haofeng Zhang received the B.Eng. and the Ph.D. degrees from the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China, in 2003 and 2007, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include computer vision and robotics.

Li Liu received the B.Eng. degree in electronic information engineering from Xi'an Jiaotong University, Xi'an, China, in 2011, and the Ph.D. degree from the Department of Electronic and Electrical Engineering, The University of Sheffield, Sheffield, U.K., in 2014. His research interests include computer vision, machine learning, multimedia, and data mining.



Yang Long received the B.Eng. degree from Northeast Forestry University, China, in 2012, and the M.Sc. and the Ph.D. degrees from the Department of Electronic and Electrical Engineering, The University of Sheffield, U.K., in 2013 and 2017, respectively. He currently holds a post-doctoral position at the Open Lab, School of Computing, Newcastle University, U.K. His research interests include computer vision, machine learning, and multimedia.



Ling Shao (M'09–SM'10) received the B.Eng. degree in electronic and information engineering from the University of Science and Technology of China, and the M.Sc. degree in medical image analysis and the Ph.D. degree in computer vision with the Robotics Research Group, University of Oxford. He was a Senior Scientist with Philips Research, The Netherlands, from 2005 to 2009. He was a Senior Lecturer with the Department of Electronic and Electrical Engineering, The University of Sheffield, from 2009 to 2014. He was a Professor with Northumbria University, from 2014 to 2016. He is currently the Chief Scientist of JD Artificial Intelligence Research (JDAIR), Beijing, China, and a Professor of computer vision and machine learning with the University of East Anglia, Norwich, U.K. His research interests include computer vision, deep learning/machine learning, multimedia, and image/video processing. He is a fellow of the British Computer Society and the Institution of Engineering and Technology. He is an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEM FOR VIDEO TECHNOLOGY, and several other journals.