Pattern Recognition 112 (2021) 107785

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Unsupervised deep hashing with node representation for image retrieval

Yangtao Wang[a], Jingkuan Song[b], Ke Zhou[a], Yu Liu[a,*]

[a] Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Key Laboratory of Information Storage System, Ministry of Education of China

[b] University of Electronic Science and Technology of China, Chengdu, China

ARTICLE INFO

Article history:
Received 11 March 2020
Revised 25 August 2020
Accepted 2 December 2020
Available online 13 December 2020

Keywords:
Deep hashing
GCN
Node representation
Image retrieval

ABSTRACT

Supervised graph convolution network (GCN) based hashing algorithms have achieved good results by recognizing images according to the relationships between objects, but they are hard to be applied to label-free scenarios. Besides, most existing unsupervised deep hashing algorithms neglect the relationships between different samples and thus fail to achieve high precision. To address this problem, we propose NRDH, an unsupervised **D**eep **H**ashing method with **N**ode **R**epresentation for image retrieval, which adopts unsupervised GCN to integrate the relationships between samples into image visual features. NRDH consists of node representation learning stage and hash function learning stage. In the first stage, we treat each image as a node of a graph and design GCN-based AutoEncoder, which can integrate the relationships between samples into node representation. In the second stage, we use above node representations to guide the network and help learn the hash function to fast achieve an end-to-end hash model to generate semantic hash codes. Extensive experiments on CIFAR-10, MS-COCO and FLICKR25K show NRDH can achieve higher performance and outperform the state-of-the-art unsupervised deep hashing methods.

1. Introduction

With the increasing amount of data, the hashing technique [1,27,28] has become one of the indispensable means in the data (image) retrieval domain. Supervised deep hashing methods using convolution neural network (CNN) [2] can make the extracted features more accurate than conventional shallow ones, thus improving the precision of hash codes. Nevertheless, the availability of hashing means has not been enhanced, because elaborately collecting such labels (using for supervision) is labor-intensive and no training data with such labels can cover the diversity of all classifications in the real world. In other words, new data or new categories without annotations are constantly emerging in real scenarios, resulting in that supervised strategies fail to achieve the desired performance. Zero-shot learning [42–44] may alleviate this problem to a certain extend, but it usually requires other supplementary information by means of knowledge transfer.

Unsupervised deep learning no longer relies on annotations information and completes the learning process through the visual features of the data itself, which owns higher practicability than supervised ones [26,29,30]. AutoEncoder (AE) [22] using deep network directly generates latent features by encoding the input data, and then decodes the encoding results (minimizing reconstruction error) to fit with the input itself. DeepBit [3] introduces the theory of image rotation invariance, then adopts a deep network to fit both the original image and the rotated one to learn the hash function. These methods are ingenious, but they suffer from over-fitting because the distribution is limited by each independent training data, so the practicability of these schemes has not been significantly improved in terms of precision.

Another label-free hashing strategy is to firstly generate pseudo labels by unsupervised means and then train hash function by supervised learning with above labels. DSTH [4] performs the minimum graph cut operation on the graph structure composed of data features by means of graph embedding [6,7], so as to allocate relative position to data according to the distances between features to generate pseudo labels. DHPL [5] integrates likelihood maximization, mutual information and quantization error minimization into the pseudo labels generating process, which ensures that the distribution of these pseudo labels shares the same latent distribution between data. However, directly applying these pseudo labels to fitting the features generated from the model may lead to large quantization loss as well as convergence difficulty of the model. To

* Corresponding author.
E-mail address: liu_yu@hust.edu.cn (Y. Liu).

https://doi.org/10.1016/j.patcog.2020.107785
0031-3203/

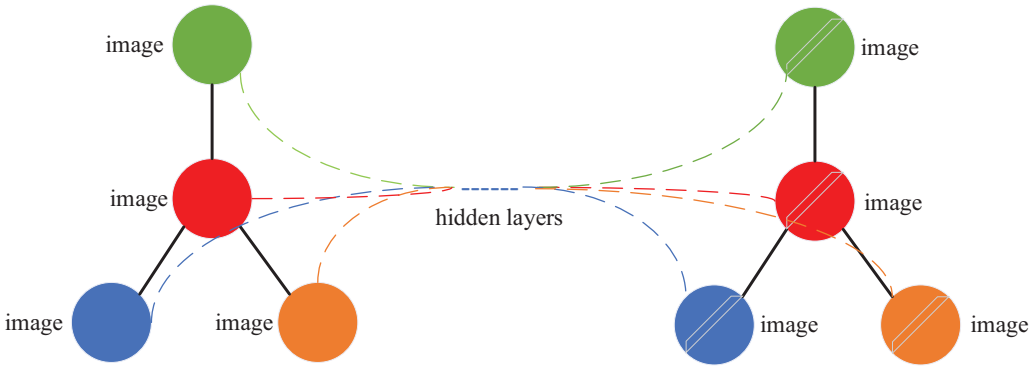


Fig. 1. Feature extraction in GCN propagation. We treat each image as a node of the graph and GCN will accumulate and propagate the relationships between different images into the output of node-level feature via multiple hidden layers.

address this problem, we hope to utilize a deep learning way to generate pseudo features that also contain relationships between samples, and use these features (which replace above relationships based pseudo labels [34]) to guide the hash function to generate binary codes.

Fortunately, graph convolution network (GCN) [8,38] has been applied as an effective means that expresses and integrates relationships into features. GCN, which operates on graph-structure data, has attracted increasing attention because of its fine capacity of exploiting relationships between nodes. Formally, GCN takes both relationships and features as input, then forward propagates the relationships in the network and extracts each node's feature according to those features of the node itself and other related neighbor nodes. As shown in Fig. 1, GCN produces node-level features by multiple hidden layers according to the diverse relationships between different nodes (images) on the graph, making that the relationships are accumulated and propagated into the output of each node.

In view of above thoughts, we introduce GCN and propose NRDH, an unsupervised **D**eep **H**ashing method with **N**ode **R**epresentation for image retrieval. NRDH consists of node representation learning stage and hash function learning stage. In the first stage, NRDH regrads each image as a node of the graph and sends both the nodes and correlation matrix in batch to our designed GCN-based AE to learn the latent node representation (feature) for each image. We will begin to train our hash function after the GCN-based AE network converges. Then NRDH utilizes the generated node representations to guide the hash function learning in the next stage. We conduct extensive experiments on three datasets including CIFAR-10, MS-COCO and FLICKR25K. Experimental results show NRDH achieves the state-of-the-art retrieval results compared with existing unsupervised deep hashing methods.

The main contribution of this paper are summarized as follows.

(1) We introduce GCN and propose an unsupervised deep hashing model NRDH, which treats each image as a node of a graph and utilize GCN to learn the similarity between images.
(2) We design a GCN-based AE network, which effectively learns the latent node representation of each image in the unsupervised way. Then NRDH utilizes these node representations to guide the hash function learning and generate semantic hash codes.
(3) NRDH is easy to implement and its image retrieval performance outperforms the state-of-the-art unsupervised deep hashing methods.

The rest of this paper is organized as follows. Section 2 talks about existing unsupervised hashing methods and GCN based works. We formulate and introduce our NRDH in detail in Section 3. Section 4 presents the ablation study and experimental results. At last, we conclude this paper in Section 5.

2. Related works

In this section, we will discuss both shallow and deep unsupervised hashing methods as well as several recent GCN based researches.

2.1. Unsupervised hashing

Unsupervised hashing methods usually adopt learning frameworks without supervised information or do not directly use label annotations to train the hash function. Locality Sensitive Hashing (LSH) [9] completes the encoding process according to the data itself, but this data-independent method can hardly explore the correlation between data. Based on Laplacian Eigenmaps (LE) [10], Spectral Hashing (SH) [11] executes spectral decomposition after constructing the graph to generate hash codes, which alleviates data-independent problem, but this scheme is limited under the premise of uniform data distribution. Self-taught Hashing (STH) [12] enhances its practicability to more scenarios by fitting the pseudo labels generated by SH. With deep learning developing rapidly, AE has become the mainstream method in unsupervised hashing. DeepBit [3] utilizes image rotation invariance to enhance the robust of hash in the AE process. Based on ITQ, DBD-MQ [14] utilizes AE to obtain the rotation of zero-centered data so as to improve the quantization accuracy. In order to obtain more accurate hash codes, DSTH [4] improves both the feature extraction part and pseudo labels fitting part of STH. DHPL [5] emphasizes the latent shared distribution in getting the pseudo labels process, which improves the representativeness of pseudo labels. Distillhash [16], one of the best unsupervised hashing algorithms, constantly refines data pairs, so that the most representative data are used to participate in the hash function learning. It can be seen that the key work of unsupervised hashing without classifications information is to mine the relationships between data, especially through graph structure. We learn from these work and expect to first generate the image features that contain the relationships between samples and then utilize these features to guide our model to obtain the hash function.

2.2. Graph convolution network

The basic idea of graph convolution network (GCN) [8,36,37] is updating one node's feature based on those features of the node itself and other related neighbor nodes according to the correlation matrix. By learning the structural similarities between training data points, GCN can integrate the relationships into data fea-

tures [38]. Formally, GCN takes the correlation matrix A as well as feature matrix X as input, and produces the node-level output. The forward propagation process in GCN is described as:

$$H^{l+1} = a^l(AH^lW^l), \tag{1}$$

where H^l, W^l and $a^l(\cdot)$ respectively denote the input, weight and non-linear activation function (like Sigmoid or ReLU) of the lth graph convolution layer.

As a deep learning technique that effectively learns and extracts relationships, GCN has been applied to relational feature extraction, node classification prediction and information retrieval tasks [20,21,33,35]. Cross-model works regrade each feature of text or image as a node representation, and complete the learning process according to the mutual relationships. Jing et al. [19] proposes to utilize GCN for text modeling and another neural network for image modeling, which achieves significant improvement with a pairwise similarity training loss function. GCH [17], another cross-modal research, learns modality-unified binary codes via an affinity graph, then adopts GCN to map hash codes by the relationships between nodes. Besides, ML-GCN [18] can achieve remarkable performance for multi-label classification tasks. It treats each object in the image as a node, and constructs a graph among these nodes, finally uses GCN to learn the probability of different objects appearing in an image, which explores the label correlation dependency and promotes the precision of image recognition. LAH [45] also utilizes GCN to explore the correlation between objects and achieve remarkable performance on image retrieval. These works are enlightening to us, but they lack exploration under unsupervised conditions.

3. Proposed methodology

Based on previous studies, we propose NRDH, an unsupervised deep hashing method with node representation for image retrieval, which integrates the complex relationships between samples into image representations. The main contribution of this paper is that we introduce GCN into unsupervised deep hashing to promote the image retrieval performance.

NRDH consists of node representation learning stage and hash function learning stage. In node representation learning stage, we take the image data (training set) as input, then use pre-trained model (i.e., ResNet 101 [23]) to extract initialized features from these data. Note that each initialized feature denotes a node in the following graph building and GCN training process. Next, different from previous works, we successively use a batch (e.g., 10) nodes to construct a graph according to the Cosine distance [31,32] between these features as well as binarize these Cosine values to calculate the correlation matrix (each element is set to be 1 if its Cosine value exceeds a predefined threshold ρ between 0 and 1). Both the correlation matrix and its corresponding features will be sent to our designed GCN-based AE network in batch to generate node representation for each image. We repeatedly conduct this training process until the AE network converges. In the next hash function learning stage, we use both the image data and their corresponding node representations (extracted from above GCN) as input to train a CNN. Noth that these node representations will guide the model to exactly extract features that express semantic relationships based on different distances between diverse categories. Fig. 2 shows the learning framework of our proposed method, and we introduce the details below.

3.1. Node representation learning stage

In this stage, we design a GCN-based AE network to learn the node representation (which will guide the hash function learning in the second stage) for each image (vector x) in the training set (containing N images). We adopt a deep model (i.e., ResNet 101) to extract initialized features for training data from the last pooling layer of the model. Given that, according to the pre-trained deep model, we acquire N z-dimensional feature vectors $\mathcal{FV} = \{v_1, v_2, \ldots, v_N\}$, where $\forall v \in \mathbb{R}^z$. Then we choose a batch (i.e., M) feature vectors (regraded as nodes) to construct a graph $\mathcal{G} = (\mathcal{FV}_\mathcal{M}, \mathcal{SE}_\mathcal{M})$, where $\mathcal{FV}_\mathcal{M}$ represents M nodes and $\mathcal{SE}_\mathcal{M} = \{e_{i,j}\}_{i,j=1}^{M}$ $(i, j \in \mathbb{Z}^+)$ represents the edges. M is a parameter to balance the relationships between different nodes (images) in the GCN propagation process. We test the change of performance with different values of M in Table 3. Note that each batch images will generate their corresponding graph. Each edge $e_{i,j}$ is associated with a weight $u_{i,j}$ (Cosine distance) defined as (note that we tried Euclidean distance but it resulted in a bad effect in Fig. 3(a) and (b)):

$$u_{i,j} = \frac{\vec{v_i} \cdot \vec{v_j}}{\|v_i\| \cdot \|v_j\|}. \tag{2}$$

However, similar to ML-GCN [18], if we directly adopt the simple correlation matrix, we will also face the over-fitting problem and over-smoothing problem, thereby declining the performance of NRDH. In the following, we set two key parameters ρ and p to respectively binarize and re-weight this correlation matrix.

Over-fitting problem The correlation matrix plays a crucial propagating impact in GCN training process. However, this simple correlation ($u_{i,j}$) may result in the over-fitting problem and damages the generalization ability of our model. Therefore, we propose to binarize the correlation and generate a relatively sparse correlation matrix to weaken the impact of edges. Specifically, we use the threshold $\rho \in [0, 1]$ to binarize the correlation matrix, and this process is described as:

$$U_{i,j} = \begin{cases} 1 & u_{i,j} > \rho, \\ 0 & \text{otherwise}. \end{cases} \tag{3}$$

where U is the binary correlation matrix.

Over-smoothing problem In the propagation process of GCN, the feature of each node is actually the weighted sum of its own feature and the adjacent nodes' features. In fact, we find that the outputs (i.e., node representations) of the last encoder layer tend to be the same sometimes, which may bring an over-smooth problem and make the node representations become indistinguishable. Therefore, to avoid this drawback, we reallocate the weight of each element in correlation matrix. The re-weighted scheme is formulated as:

$$U'_{i,j} = \begin{cases} p & i = j, \\ (1-p)/\sum_{j=1}^{M} U_{i,j} & \text{otherwise}. \end{cases} \tag{4}$$

where U' is the re-weighted correlation matrix, and $p \in [0, 1]$ or $1 - p$ denotes the weights allocated to the node itself or other cor-

Table 1
Parameters of GCN for node representation learning.

AutoEncoder		Parameter		Activation function
Location	Layer	Input	Output	
1	Graph convolution layer	2048	64	ReLU
2	Graph convolution layer	64	64	ReLU
3	Graph convolution layer	64	g	–

Table 2
Parameters of fc layers for hash function learning.

fc layers		Parameter		Activation function
Location	Layer	Input	Output	
1	Linear	ResNet_output	256	LeakyReLU + BN
2	Linear	256	64	LeakyReLU + BN
3	Linear	64	g	BN + Tanh

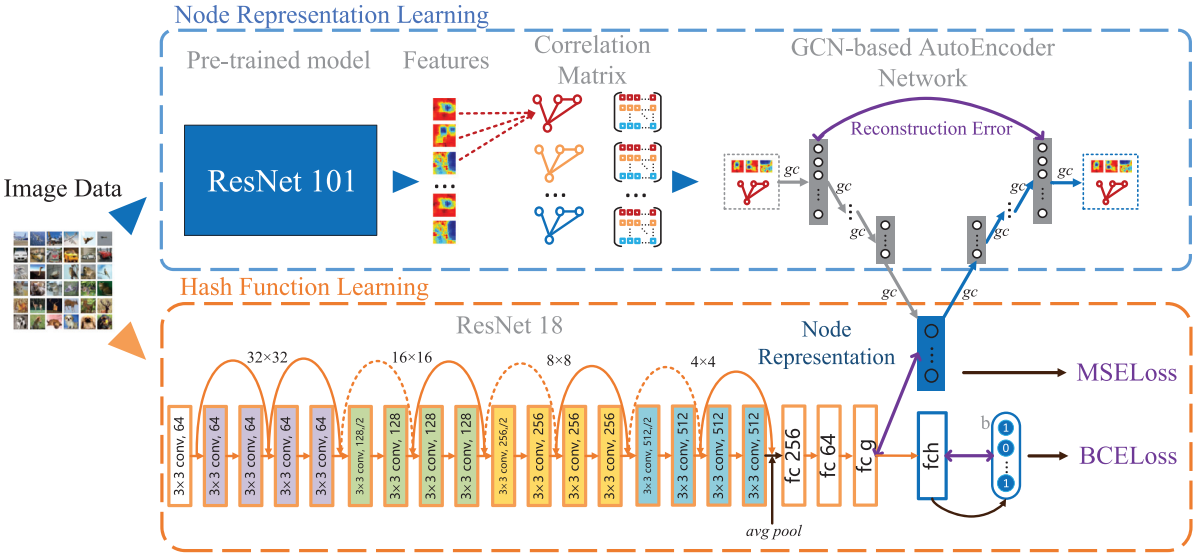
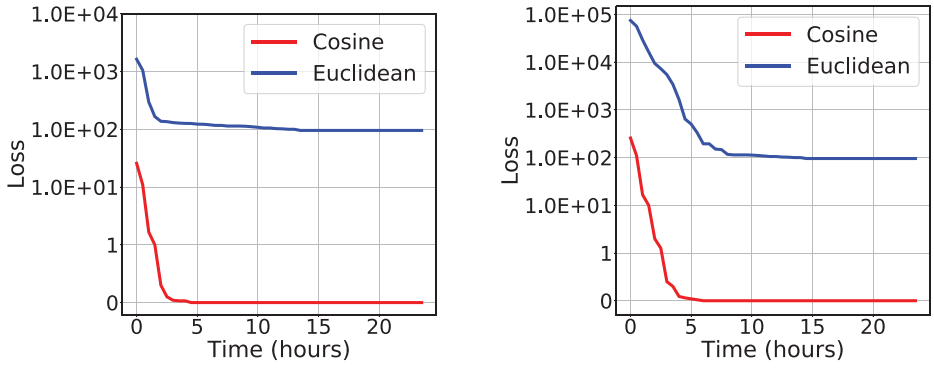


Fig. 2. NRDH framework.



(a) CIFAR-10 (b) MS-COCO

Fig. 3. GCN convergence on CIFAR-10 and MS-COCO using different distance metrics to construct the correlation matrix.

related nodes. In this way, we can update the node representations by choosing a fixed weight p. For example, a node representation will rely more on itself if using a larger p. Otherwise, its representation will be determined by other neighbor nodes.

The correlation matrix will greatly influence the performance of NRDH in the GCN propagation. Therefore, in order to find the optimal correlation matrix, we first vary the values of ρ (see Table 4) and p (see Table 5) and then observe the change of performance on different datasets in our experiments.

GCN-based AE training After obtaining the initialized feature of each image (node) and the correlation matrix of each batch images (nodes), we begin to train our GCN-based AE network. In each batch, the input consists of M nodes' initialized features $X = \{v_1, v_2, \ldots, v_M\}$ and their corresponding correlation matrix $U^{'}$. The reason why we design AE network is that it is an excellent unsupervised method to map input to latent representation space. In our encoder network, we intend to integrate the complex relationships between samples of each batch into the node representation learning. We reconstruct the input in the decoder network and further update all parameters of the whole network with reconstruction errors by each iteration. Given the input (X and $U^{'}$) and K hidden layers, each GCN layer can be written as:

$$X^{l+1} = f^l(U^{'}X^lW^l), l \in [1, K] \tag{5}$$

where X^l denotes the latent features of M nodes in the l-layer, W^l and denotes the weights of the l-layer respectively, and $f^l(\cdot)$ denotes the non-linear operation which is a ReLU function. Thus we can integrate and model the complex relationships into node representation learning by multiple GCN layers. When $l = K$, we can acquire X^{K+1} as the result of the encoder network (each node is mapped into a pre-defined g-dimensional real-valued vector). Next, we generate the output $\widehat{X}$ as the result of reconstruction space by reversing the above processing. As shown in the blue frame of Fig. 2, the loss function $\mathcal{L}_r$ for this reconstruction process is shown as follows:

$$\mathcal{L}_r = avg\|X - \widehat{X}\|_2^2. \tag{6}$$

where $avg\|\cdot\|_2^2$ denotes the average reconstruction loss for training samples. Besides, in order to prevent over-fitting, we design regularizer term $\mathcal{L}_w$ as follows:

$$\mathcal{L}_w = \frac{1}{2}\sum_{l=1}^{K}\|W^{(l)}\|_F^2 + \|\widehat{W^{(l)}}\|_F^2. \tag{7}$$

Based on above design, the terminal loss function $\mathcal{L}_1$ is :

$$\mathcal{L}_1 = \alpha\mathcal{L}_r + \beta\mathcal{L}_w, \tag{8}$$

where α and β are hyper-parameters that satisfy $\alpha + \beta = 1$.

When $\mathcal{L}_1$ reaches convergence after multiple epochs, the GCN will generate a g-dimensional real-valued vector ($g(x)$) for each im-

age contained in the training set, where g is a pre-defined dimension (i.e., 16, 32, 48, 64, 96, 128 in our experiments) of the GCN's output. These node representations that contain semantic relationships between samples will guide the hash function learning in the next stage.

3.2. Hash function learning stage

The primary work of this stage is to learn the hash function based on the feature ($g(x)$) of each image (x) from the first stage, where $x \in [x_1, x_2, \ldots, x_N]$. There exist two main reasons that we have to use pseudo (features) labels to fit the hash function for samples. One is that we need to weaken the effect of samples distribution (this principle is the same as STH). The other is that both the graph (i.e., correlation matrix) building and GCN training processes affect the efficiency of generating hash codes, and we hope our hash function model applied to on-line process is end-to-end.

We employ CNN to train the image data with node representations from the first stage. As shown in the orange frame of Fig. 2, we use two loss functions to train this network, where fc is the fully-connected layer (used for fitting node representations) and fch is the activation output layer (used for fitting and generating binary values). Formally, we set a function $f : \mathbb{R}^I \to \mathbb{R}^O$, where I is the input set which has N images, O is the output set and x represents an input vector (image):

$$\begin{aligned} f^{(1)}(x) &= \sigma(W^{(1)}x + b^{(1)}), \\ f^{(q)}(x) &= \sigma(W^{(q)}f^{(q-1)}(x) + b^{(q)}). \end{aligned} \tag{9}$$

where $q = 2, 3, \ldots, Q$ (Q is number of CNN layers) and $\sigma(*)$ is Leakly ReLU with BatchNorm (BN) function. The Leakly ReLU $\Psi(*)$ is calculated as follows:

$$\Psi(x) = \begin{cases} x & x > 0, \\ \lambda x & x \le 0. \end{cases} \tag{10}$$

where $\lambda \in [0, 1)$. The BN $\Gamma(*)$ is calculated as follows:

$$\Gamma^{(q)}(x) = \frac{x^{(q)} - E^{(q)}(x)}{\sqrt{V^{(q)}(x)}}, \tag{11}$$

where

$$\begin{aligned} E^{(q)}(x) &= \frac{1}{n}\sum_{i=1}^{n} x_i^{(q)}, \\ V^{(q)}(x) &= \frac{1}{n}\sum_{i=1}^{n} (x_i^{(q)} - E^{(q)}(x))^2. \end{aligned} \tag{12}$$

In the last (Qth) layer, we respectively denote the output of fc and the activation output of fch as:

$$\begin{aligned} fc(x) &= f^{(Q)}(x), \\ fch(x) &= Tanh(fc(x)). \end{aligned} \tag{13}$$

where $Tanh(\cdot)$ is the activation function calculated as:

$$Tanh(x) = \frac{exp(x) - exp(-x)}{exp(x) + exp(-x)}. \tag{14}$$

We use MSELoss[1] to fit $fc(x)$ with node representations, and denote this loss function as:

$$\mathcal{L}_{fc} = avg\sum_{i}^{N} \| fc(x_i) - g(x_i) \|_2^2. \tag{15}$$

where x_i denotes the ith image and $g(x_i)$ denotes the node representation of x_i. At the same time, we binarize the output $fch(x_i)$:

$$b_i = 0.5 \times (sign(fch(x_i)) + 1). \tag{16}$$

Given that the pth value in $fch(x_i)$ is $fch(x_i^p)$, we use Binary Cross Entropy Loss (BCELoss,[2] a common function to fit multiple labels) to train the hash function, and denote this loss function as (we tried other multi-label classification loss functions but did not obtain a better result than BCELoss in Fig. 5):

$$\mathcal{L}_{fch} = -\sum_{p=1}^{g} b_i^p log(fch(x_i^p)) + (1 - b_i^p) log(1 - fch(x_i^p)). \tag{17}$$

Based on above design, the terminal loss function $\mathcal{L}_2$ is calculated as:

$$\mathcal{L}_2 = \mu\mathcal{L}_{fc} + \omega\mathcal{L}_{fch}, \tag{18}$$

where μ and ω are hyper-parameters that satisfy $\mu + \omega = 1$. Note that different combinations between μ and ω reflect which one loss function contributions more to the final generated hash codes. We record the change of performance in Table 6 by varying the values of μ and ω. We use $\mathcal{L}_2$ to train and update the hash function model until the network converges. At last, for the ith image, the pth bit binary value of $fch(x_i^p)$ is:

$$fch(x_i^p) = \begin{cases} 1 & fch(x_i^p) \ge 0, \\ 0 & \text{otherwise}. \end{cases} \tag{19}$$

4. Experiments

In this part, we first describe the experimental settings and implementation details (including parameters settings, networks settings, datasets description and evaluation metrics), then present the ablation study about the components that influence our model, next show the experimental results of NRDH compared with existing unsupervised deep hashing algorithms and finally illustrate the visual retrieval results. We choose the pre-trained model (ResNet 101) to extract initialized features from the last pooling layer for training images. In the correlation matrix construction, we respectively set $\rho = 0.4$ (Eq. (3)) and $p = 0.8$ (Eq. (4)) to binarize and re-weight the correlation matrix. As for the hyper-parameters settings, we set $\alpha = 0.999$ and $\beta = 0.001$ (common in regularizer term setting) to train the GCN loss function $\mathcal{L}_1$ (Eq. (8)) to generate the g-dimensional node representation. Meanwhile, in the hash function learning stage, we set $\mu = 0.8$ and $\omega = 0.2$ (Table 6) to learn the hash function $\mathcal{L}_2$ (Eq. (18)). In fact, we have tried several networks to complete this training, and found the ResNet 18 [23] could bring the best performance and higher covergence efficiency by avoiding vanishing gradient. Note that we reset the fc layers and retrain the whole network. The specific parameters of GCN-based AE and fc layers of ResNet are respectively listed in Tables 1 and 2, where g denotes the hash length. We adopt the commonly-used evaluation metrics including mean Average Precision (mAP), Precision@Recall (PR) and F_1-score to verify the retrieval performance of NRDH on the following three datasets. All the experiments are implemented on a Linux server with 8 Tesla P40 GPUs.

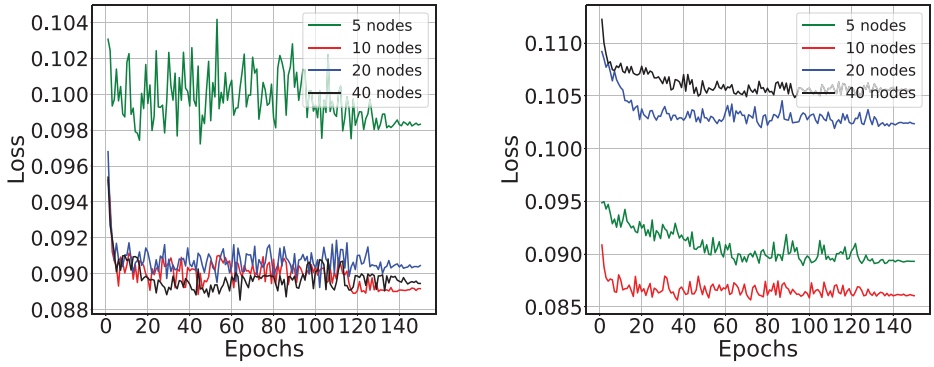
4.1. Datasets

CIFAR-10 consists of 60,000 images which are labeled with 10 classes, with 6000 samples in each class. There are 5000 training images and 1,000 test images in each class.

MS-COCO is a popular dataset for image recognition, segmentation and captioning. The current release contains 118,287 training

[1] https://pytorch.org/docs/master/nn.html?highlight=loss#torch.nn.MSELoss

[2] https://pytorch.org/docs/master/nn.html?highlight=loss#torch.nn.BCELoss



(a) CIFAR-10 (b) MS-COCO

Fig. 4. GCN convergence on CIFAR-10 and MS-COCO using different number of nodes in a batch.

Table 3
PR and F_1-score varying number of nodes at 48 bits.

Dataset	Number of nodes	Metric	Recall						F_1-score
			10%	20%	40%	60%	80%	100%	
CIFAR-10	5	Precision	0.483	0.392	0.314	0.221	0.184	0.157	0.298
	10		**0.541**	**0.451**	**0.359**	**0.253**	**0.221**	**0.170**	**0.303**
	20		0.538	0.447	0.353	0.246	0.217	0.167	0.299
	40		0.540	0.449	0.357	0.246	0.220	0.170	0.301
MS-COCO	5	Precision	0.579	0.517	0.427	0.353	0.259	0.171	0.333
	10		**0.581**	**0.591**	**0.430**	**0.354**	**0.261**	**0.171**	**0.334**
	20		0.578	0.516	0.424	0.351	0.258	0.170	0.332
	40		0.578	0.517	0.425	0.350	0.258	0.170	0.332

Table 4
mAP results varying ρ at 48 bits.

Dataset	ρ									Metric
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	
CIFAR-10	0.332	0.336	0.339	**0.341**	0.339	0.337	0.335	0.333	0.331	mAP
MS-COCO	0.398	0.401	0.405	**0.408**	0.406	0.404	0.404	0.402	0.401	

Table 5
mAP results varying p at 48 bits.

Dataset	p									Metric
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	
CIFAR-10	0.321	0.325	0.330	0.334	0.337	0.339	0.340	**0.341**	0.340	mAP
MS-COCO	0.394	0.398	0.400	0.403	0.405	0.407	0.407	**0.408**	0.406	

Table 6
mAP results varying (μ, ω) at 48 bits.

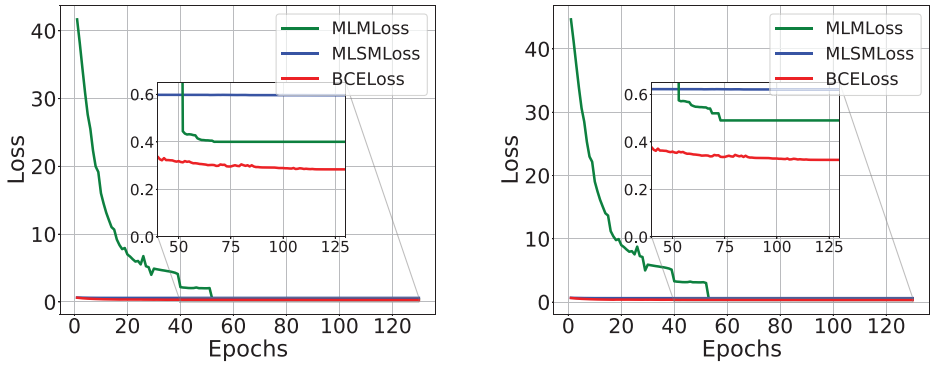
Dataset	(μ, ω)								Metric
	(0.1, 0.9)	(0.2, 0.8)	(0.3, 0.7)	(0.4, 0.6)	(0.5, 0.5)	(0.6, 0.4)	(0.7, 0.3)	(0.8, 0.2)	(0.9, 0.1)	
CIFAR-10	0.317	0.324	0.331	0.335	0.336	0.337	0.339	**0.341**	0.336	mAP
MS-COCO	0.382	0.390	0.395	0.399	0.401	0.402	0.406	**0.408**	0.401	

images, 40,504 validation images and 40,775 test images, where each image is averagely labeled by 2.9 object labels from the 80 semantic concepts. Since the ground-truth labels of the test set are not available, we evaluate the performance on the validation set.

FLICKR25K is a collection of 25,000 multi-label images belonging to 24 unique provided labels, and each image is annotated by 4.7 labels on average. We randomly select 2000 images as the test set. The remaining images are used as the retrieval images, where we randomly select 10,000 images as the training set.

4.2. Ablation study

In this section, we conduct ablation study to explore the influence of using different components on our model. We compare the experimental results on (single-object) CIFAR-10 and (multi-label) MS-COCO from the following four aspects: (1) distance metric ways of constructing the correlation matrix (in node representation learning stage), (2) number of nodes sent to the GCN in a batch (in node representation learning stage), (3) different



(a) CIFAR-10 (b) MS-COCO

Fig. 5. Hash function convergence on CIFAR-10 and MS-COCO with different loss functions.

loss functions to train the hash function (in hash function learning stage), (4) hyper-parameters ρ (Eq. (3)), p (Eq. (4)), μ and ω (Eq. (18)) settings and (5) GCN component influence on the model.

Distance metrics In this part, we respectively demonstrate the effects of using Cosine distance and Euclidean distance (mentioned in Section 3.1) on the model convergence in node representation learning stage of NRDH. As shown in Fig. 3(a) and (b), Euclidean distance brings a large loss error which completely damages the convergence of our model, while using Cosine distance can converge to a much smaller value with a faster speed on these two datasets. To ensure the convergence and reduce loss error of NRDH, we choose Cosine distance to construct the correlation matrix.

Number of nodes In each batch, we will send M (mentioned in Section 3.1) nodes to the GCN, and then complete the training process via multiple epochs. On the one hand, in order to speed up the training and save the efficiency, we will not allocate too many nodes in each batch, because a large graph will severely affect the convergence efficiency of our model. Therefore, we try to design a relatively small graph to propagate the relationships between nodes. On the other hand, we hope this small graph can bring high-quality hash codes and achieve better performance. We test the convergence trend in GCN training using $M = 5, 10, 20, 40$ nodes in a batch. This process aims to find the optimal number of nodes that can well propagate the relationships and promote the performance of NRDH. As shown in Fig. 4(a), using 10, 20, or 40 nodes to construct the graph brings a similar convergence value (nearly 0.089) except for a relatively larger loss error on 5 nodes, but $M = 10$ produces a slight advantage than other settings on CIFAR-10. Furthermore, Fig. 4(b) presents the training results on the multi-label MS-COCO dataset. Obviously, $M = 10$ brings an overwhelming superiority in terms of convergence than other three settings. In addition, Table 3 lists the PR and F_1-score with different number of nodes at 48-bits on these two datasets. Although there are not large differences between different number of nodes in terms of precision, $M = 10$ produces the best results on the whole. Therefore, in order to reduce the loss error and enhance the precision of retrieval effect, we choose $M = 10$ to conduct the following experiments. As a result, $M = 10$ can well balance the relationships between different images on both single-label and multi-label datasets.

Different loss functions In this part, we train the hash model by comparing BCELoss with MultiLabelMarginLoss (MLMLoss) and MultiLabelSoftMarginLoss (MLSMLoss), all of which are common multi-label classification loss functions[3] (mentioned in Section 3.2) provided by PyTorch. Fig. 5(a) and (b) present the convergence trend of the hash function. We find BCELoss brings a lower loss error with faster convergence speed. Therefore, in this hash function learning stage, we choose BCELoss to fit binary values.

Hyper-parameters settings In this part, we evaluate the effect on our model using different hyper-parameters including ρ (Eq. (3)), p (Eq. (4)), μ and ω (Eq. (18)). As shown in Table 4, we vary ρ from 0.1 to 0.9 to binarize the correlation matrix and find NRDH can achieve the highest mAP on both CIFAR-10 and MS-COCO when $\rho = 0.4$. This may result from $\rho = 0.4$ is a better threshold to restrict the relationships between nodes. As shown in Table 5, we vary p from 0.1 to 0.9 to generate the weighted correlation matrix and find NRDH can achieve the highest mAP on both CIFAR-10 and MS-COCO when $p = 0.8$. As a result, $p = 0.8$ can well balance the relationships between a node and its neighbor nodes. By varying ρ and p, we aim to find the optimal correlation matrix in the GCN propagation. Besides, we also conduct experiments to vary the values of (μ, ω) pairs to train the hash function and list the mAP results at 48 bits in Table 6. As we see, we change μ from 0.1 to 0.9, and find $\mu = 0.8, \omega = 0.2$ can bring the best result. This may result from that $\mu = 0.8, \omega = 0.2$ can balance both the node representation fitting and binary values mapping. As we know, the binarization process higly relies on the fitted features. Therefore, $\mu = 0.8, \omega = 0.2$ can well fit the node representations and help generate high-quality hash codes.

GCN component on different models In this part, we evaluate the effect of using different feature extraction models on our model, including VGG [39] and ResNet 101 [23] with and without GCN component on CIFAR-10 and MS-COCO in Table 7. As we see, the combination of ResNet 101 and GCN can achieve the highest mAP at 48 bits on these two datasets. Note that using VGG with GCN can also obtain a good result, but neither VGG nor ResNet 101 will produce a satisfied effect without GCN. This illustrates GCN plays a crucial role in promoting the performance of the model. In addition, we believe a more powerful network will help learn better image representations which may further improve the precision of hash codes.

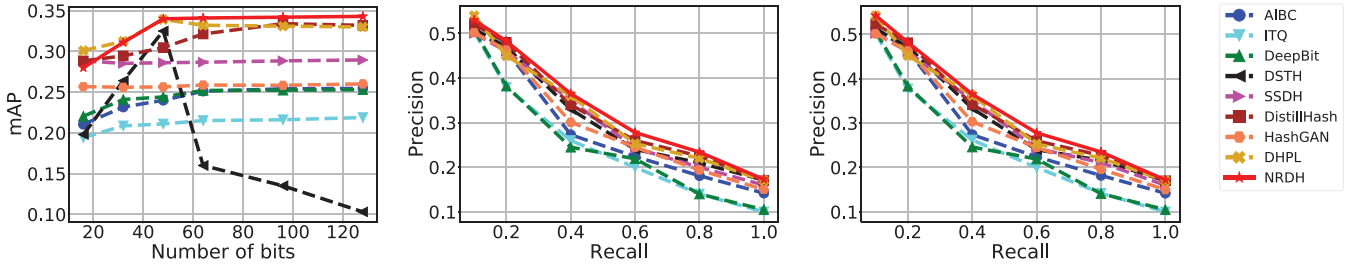
4.3. Comparisons with the state-of-the-art methods

From above ablation study, we find NRDH can bring a better result on multi-object MS-COCO than single-label CIFAR-10. This is because NRDH can integrate rich semantic information into node representation, so we add another multi-object FLICKR25K to verify our observation in this section. We compare NRDH with the state-of-the-art unsupervised image hashing algorithms published in recent years, including AIBC [24], ITQ [13], DeepBit [3],

[3] https://pytorch.org/docs/master/search.html?q=loss&check_keywords=yes&area=default

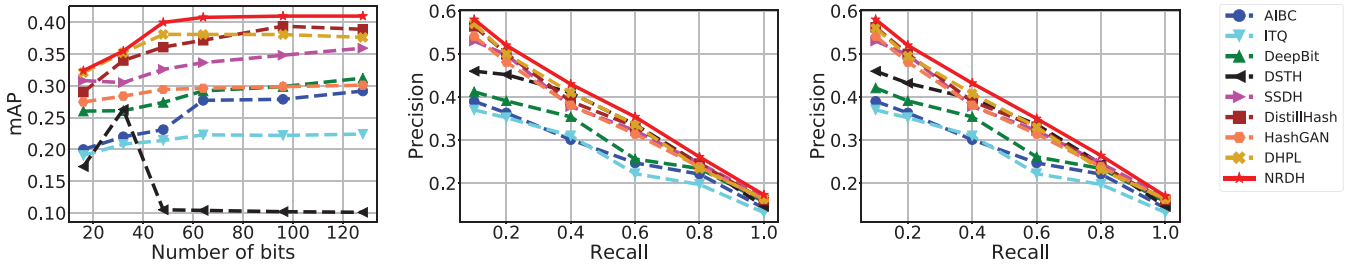
Table 7
mAP results with or without GCN component on different models at 48 bits.

Dataset	Method				Metric
	VGG		ResNet 101		
	with GCN	without GCN	with GCN	without GCN	
CIFAR-10	0.340	0.277	**0.341**	0.281	mAP
MS-COCO	0.407	0.342	**0.408**	0.346	



(a)mAP on CIFAR-10 (b) 48-bits PR curve on CIFAR-10 (c) 64-bits PR curve on CIFAR-10

Fig. 6. mAP with different code lengths and (48, 64)-bits PR curve on CIFAR-10.



(a) mAP on MS-COCO (b) 48-bits PR curve on MS-COCO (c) 64-bits PR curve on MS-COCO

Fig. 7. mAP with different code lengths and (48, 64)-bits PR curve on MS-COCO.

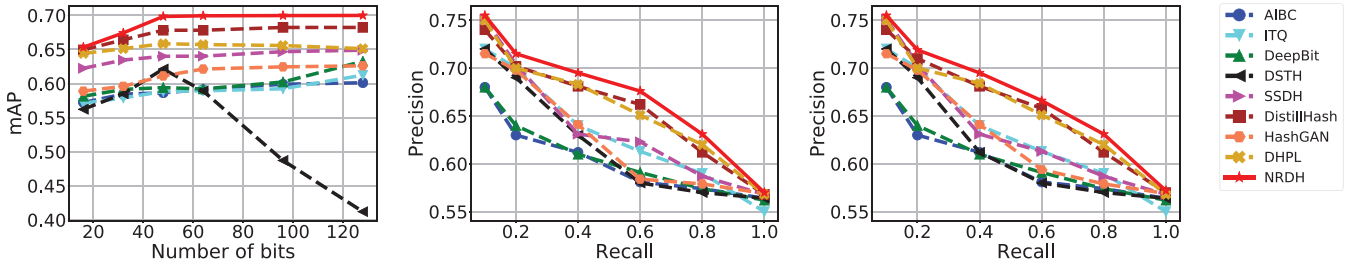
DSTH [4,40,41], SSDH [15], DistillHash [16], HashGAN [25], and DHPL [5]. On all datasets, we show the mAP results at different code lengths (i.e., 16, 32, 48, 64, 96, 128) and draw PR curve with Hamming radius 2 at 48 and 64 bits.

As shown in Fig. 6 which lists the comparison results of mAP and PR curve on CIFAR-10, NRDH achieves the best performance at 48 bits (without significant growth after 48 bits) and outperforms other candidates after 48 bits in terms of mAP. In Fig. 6(b), the precision of NRDH is dominant when its recall rate reaches 0.6 and 0.8, which is respectively higher than the runner up by 1.8% and 1.3% at 48 bits. Fig. 6(c) presents a similar trend of PR curve on CIFAR-10 at 64 bits. From Fig. 7(a) which shows the mAP results on MS-COCO, NRDH produces higher mAP and exceeds the runner up by nearly 0.5% to 2.4% on all cases. Besides, its precision is dominant at all recall rate cases at 48 bits (Fig. 7(b)) and 64 bits (Fig. 7(c)) with an average 2% performance improvement. Furthermore, we evaluate the performance of NRDH on FLICKR25K in Fig. 8. Similarly, NRDH achieves the higher mAP in all cases, and respectively outperforms the runner up by 1% to 3% in Fig. 8(a). In addition, our method also obtains the higher precision than all candidates in all recall rate cases at 48 bits (Fig. 8(b)) as well as 64 bits (Fig. 8(c)), and averagely exceeds the runner up by 1% to 3%.

From above experimental analysis, we find that NRDH is partially dominant on CIFAR-10, completely dominant on MS-COCO, and absolutely dominant on FLICKR25K. This phenomenon lies in that these three datasets own diverse characteristics (objects). As we know, CIFAR-10 is a single-object dataset, which can not give full play to the advantage of NRDH, resulting its relatively poor performance. NRDH can integrate complex semantic relationships into node representation, so it can well learn similarity hash codes on MS-COCO which consists of multi-label images. Furthermore, NRDH can achieve remarkable performance on more complex FLICKR25K (with more objects of each image on average). It is the ability of NRDH to integrate relationships into image representation that plays a crucial impact on datasets with rich semantic information. All the results in Figs. 6–8 demonstrate the superiority of our method.

4.4. Visual retrieval results

In this section, we visualize the top 10 retrieved images for given random query data on CIFAR-10 at 48 bits of NRDH. As shown in Fig. 9, we respectively list the returned results for “automobile”, “bird”, “horse” and “ship”. Our approach can achieve sat-



(a) mAP on FLICKR25K (b) 48-bits PR curve on FLICKR25K (c) 64-bits PR curve on FLICKR25K

Fig. 8. mAP with different code lengths and (48, 64)-bits PR curve on FLICKR25K.

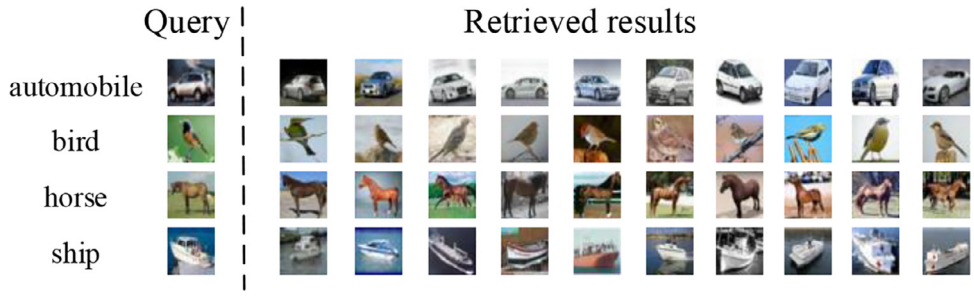


Fig. 9. Top 10 visual retrieval results on CIFAR-10.

isfying results that qualitatively illustrate NRDH is able to extract semantic binary attributes.

5. Conclusion

In this paper, we propose NRDH, an unsupervised deep hashing method with node representation for image retrieval, which consists of node representation learning stage and hash function learning stage. NRDH integrates the relationships between samples into node representations via GCN-based AE in the first stage. These node representations can guide the model training and help learn the hash function to generate semantic hash codes in the second stage. We conduct extensive ablation study to explore the influence of using different parameters settings and components on our model. In addition, we compare NRDH with the state-of-the-art unsupervised hashing methods and the experimental results verify the superiority of our method. Finally, the visual retrieval results illustrate NRDH effectively capture the similarity between samples.

Our proposed approach is effective to learn the relationships between samples, but the performance improvement is limited. NRDH didn't take the attention mechanism into consideration. In the future, we will improve our scheme from three aspects. (1) In the node representation learning stage, we can adopt more powerful model with attention mechanism to extract the initialized features. (2) In the GCN-based AE network, we can add some constraints and add the attention mechanism on the loss function $\mathcal{L}_1$ to obtain more accurate node representation. (3) Once we obtain the accurate node representations, in the hash function learning stage, we can design and train a simple CNN instead of ResNet-18 to further speed up this learning process.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

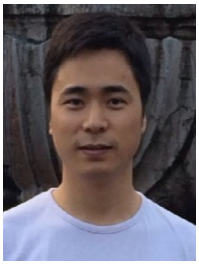
This work is supported by the Innovation Group Project of the National Natural Science Foundation of China no. 61821003 and the National Natural Science Foundation of China no. 61902135. Thanks for Jay Chou, a celebrated Chinese singer whose songs have been accompanying the author.

References

[1] J. Song, H. Jégou, C. Snoek, Q. Tian, N. Sebe, Guest editorial: large-scale multimedia data retrieval, classification, and understanding, IEEE Trans. Multimed. 19 (9) (2017) 1965–1967.
[2] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: NIPS, 2012, pp. 1106–1114.
[3] K. Lin, J. Lu, C.-S. Chen, J. Zhou, Learning compact binary descriptors with unsupervised deep neural networks, in: CVPR, 2016, pp. 1183–1192.
[4] Y. Liu, J. Song, K. Zhou, L. Yan, L. Liu, F. Zou, L. Shao, Deep self-taught hashing for image retrieval, IEEE Trans. Cybern. 49 (6) (2019) 2229–2241.
[5] H. Zhang, L. Liu, Y. Long, L. Shao, Unsupervised deep hashing with pseudo labels for scalable image retrieval, IEEE Trans. Image Process. 27 (4) (2018) 1626–1638.
[6] B. Perozzi, R. Al-Rfou, S. Skiena, Deepwalk: online learning of social representations, in: SIGKDD, 2014, pp. 701–710.
[7] D. Wang, P. Cui, W. Zhu, Structural deep network embedding, in: SIGKDD, 2016, pp. 1225–1234.
[8] T.N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, ICLR, 2017.
[9] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: SoCG, 2004, pp. 253–262.
[10] M. Belkin, P. Niyogi, Laplacian eigenmaps for dimensionality reduction and data representation, Neural Comput. 15 (6) (2003) 1373–1396.
[11] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: NIPS, 2008, pp. 1753–1760.
[12] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: SIGIR, 2010, pp. 18–25.
[13] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 35 (12) (2013) 2916–2929.
[14] Y. Duan, J. Lu, Z. Wang, J. Feng, J. Zhou, Learning deep binary descriptor with multi-quantization, in: CVPR, 2017, pp. 4866–4875.
[15] E. Yang, C. Deng, T. Liu, W. Liu, D. Tao, Semantic structure-based unsupervised deep hashing, in: IJCAI, 2018, pp. 1064–1070.

[16] E. Yang, T. Liu, C. Deng, W. Liu, D. Tao, Distillhash: unsupervised deep hashing by distilling data pairs, in: CVPR, 2019, pp. 2946–2955.
[17] R. Xu, C. Li, J. Yan, C. Deng, X. Liu, Graph convolutional network hashing for cross-modal retrieval, in: IJCAI, 2019, pp. 982–988.
[18] Z.-M. Chen, X.-S. Wei, P. Wang, Y. Guo, Multi-label image recognition with graph convolutional networks, in: CVPR, 2019, pp. 5177–5186.
[19] J. Yu, Y. Lu, Z. Qin, W. Zhang, Y. Liu, J. Tan, L. Guo, Modeling text with graph convolutional network for cross-modal information retrieval, in: PCM, 2018, pp. 223–234.
[20] L. Yao, C. Mao, Y. Luo, Graph convolutional networks for text classification, in: AAAI, 2019, pp. 7370–7377.
[21] F. Hu, Y. Zhu, S. Wu, L. Wang, T. Tan, Hierarchical graph convolutional networks for semi-supervised node classification, in: IJCAI, 2019, pp. 4532–4539.
[22] A. Krizhevsky, G.E. Hinton, Using very deep autoencoders for content-based image retrieval, ESANN, 2011.
[23] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: CVPR, 2016, pp. 770–778.
[24] F. Shen, W. Liu, S. Zhang, Y. Yang, H.T. Shen, Learning binary codes for maximum inner product search, in: ICCV, 2015, pp. 4148–4156.
[25] K.G. Dizaji, F. Zheng, N. Sadoughi, Y. Yang, C. Deng, H. Huang, Unsupervised deep generative adversarial hashing network, in: CVPR, 2018, pp. 3664–3673.
[26] D.C.G.a. Pedronette, F.M.F. Gonçalves, I.R. Guilherme, Unsupervised manifold learning through reciprocal kNN graph and connected components for image retrieval tasks, Pattern Recognit. 75 (2018) 161–174.
[27] Y. Luo, Y. Yang, F. Shen, Z. Huang, P. Zhou, H.T. Shen, Robust discrete code modeling for supervised hashing, Pattern Recognit. 75 (2018) 128–135.
[28] J. Song, L. Gao, L. Liu, X. Zhu, N. Sebe, Quantization-based hashing: a general framework for scalable image and video retrieval, Pattern Recognit. 75 (2018) 175–187.
[29] A. Sanakoyeu, M.A. Bautista, B. Ommer, Deep unsupervised learning of visual similarities, Pattern Recognit. 78 (2018) 331–343.
[30] X.-S. Wei, C.-L. Zhang, J. Wu, C. Shen, Z.-H. Zhou, Unsupervised object discovery and co-localization by deep descriptor transformation, Pattern Recognit. 88 (2019) 113–126.
[31] V. Novotný, Implementation notes for the soft cosine measure, in: CIKM, 2018, pp. 1639–1642.
[32] Y. Liu, Y. Wang, J. Song, C. Guo, K. Zhou, Z. Xiao, Deep self-taught graph embedding hashing with pseudo labels for image retrieval, in: ICME, 2020, pp. 1–6.
[33] B. Topcu, H. Erdogan, Fixed-length asymmetric binary hashing for fingerprint verification through GMM-SVM based representations, Pattern Recognit. 88 (2019) 409–420.
[34] J. Liang, R. He, Z. Sun, T. Tan, Exploring uncertainty in pseudo-label guided unsupervised domain adaptation, Pattern Recognit. 96 (2019).
[35] C.-Y. Zhang, Q. Zhao, C.L.P. Chen, W. Liu, Deep compression of probabilistic graphical networks, Pattern Recognit. 96 (2019).
[36] F. Manessi, A. Rozza, M. Manzo, Dynamic graph convolutional networks, Pattern Recognit. 97 (2020).
[37] Z. Luo, L. Liu, J. Yin, Y. Li, Z. Wu, Deep learning of graphs with ngram convolutional neural networks, IEEE Trans. Knowl. Data Eng. 29 (10) (2017) 2125–2139.
[38] R. Levie, F. Monti, X. Bresson, M.M. Bronstein, Cayleynets: graph convolutional neural networks with complex rational spectral filters, IEEE Trans. Signal Process. 67 (1) (2019) 97–109.
[39] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, ICLR, 2015.
[40] Y. Liu, Y. Wang, K. Zhou, Y. Yang, Y. Liu, J. Song, Z. Xiao, A framework for image dark data assessment, in: APWeb-WAIM, 2019, pp. 3–18.
[41] Y. Liu, Y. Wang, K. Zhou, Y. Yang, Y. Liu, Semantic-aware data quality assessment for image big data, Future Gener. Comput. Syst. 102 (2020) 53–65.
[42] Y. Yang, Y. Luo, W. Chen, F. Shen, J. Shao, H.T. Shen, Zero-shot hashing via transferring supervised knowledge, in: ACM MM, 2016, pp. 1286–1295.
[43] H. Zhang, Y. Long, L. Shao, Zero-shot hashing with orthogonal projection for image retrieval, Pattern Recognit. Lett. 117 (2019) 201–209.
[44] Y. Guo, G. Ding, J. Han, Y. Gao, Sitnet: discrete similarity transfer network for zero-shot hashing, in: IJCAI, 2017, pp. 1767–1773.
[45] Y. Xie, Y. Liu, Y. Wang, L. Gao, P. Wang, K. Zhou, Label-attended hashing for multi-label image retrieval, in: IJCAI, 2020, pp. 955–962.

Yangtao Wang received a B.S. degree from the Faculty of Mathematics and Statistics, Hubei University, Wuhan, China, in 2015. He is currently a Ph.D. candidate in Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology (HUST). His current research interests include differential privacy, machine learning, graph neural network, etc. He has published papers in international conferences and journals, including SIGMOD, ADMA, APWeb-WAIM, DATE, FGCS, WWWJ, etc.



Jingkuan Song is currently a professor of University of Electronic Science and Technology of China. He was a Postdoctoral Research Scientist in Columbia University. He joined a University of Trento as a Research Fellow (2014–2016). He obtained his Ph.D. degree in Information Technology from The University of Queensland (UQ), Australia, in 2014. He received his BS degree in Computer Science from University of Electronic Science and Technology of China. His research interest includes large-scale multimedia retrieval, image/video segmentation and image/video annotation using hashing, graph learning and deep learning techniques. Jingkuan serves as a lead guest editor for TMM special issue on 'Large-scale Multimedia Data Retrieval, Classification, and Understanding'. He is also a reviewer for IEEE Transaction on Multimedia, Transactions on Cybernetics, Transactions on Knowledge and Data Engineering, etc. He is a member of IEEE, and a member of ACM.

Ke Zhou is currently a professor of Huazhong University of Science and Technology (HUST). He received the B.S., M.S., and Ph.D. degrees in computer science and technology from HUST in 1996, 1999, and 2003 respectively. His main research interests include network/cloud storage, data security and service, parallel I/O. He has published more than 50 papers in international journals and conferences, including MSST, ACM MM, INFOCOM, ICS, ICPP, ICCD, SIGMOD, etc. He is a senior member of CCF and a member of IEEE.

Yu Liu received a B.S. degree and an M.S. degree from the School of Computer Science and Technology, Wuhan Institute of Technology, Wuhan, China, in 2008 and 2012 respectively, and a Ph.D. degree in computer science from Huazhong University of Science and Technology (HUST) in 2017. Currently, he is a postdoctoral researcher in the Dept. of Software Engineering, HUST, China. His current research interests include machine learning, large-scale multimedia search, big data, and storage, etc. He has published papers in international conferences and journals, including ACM MM, SIGMOD, APWeb-WAIM, IEEE Transactions on Cybernetics, FGCS, WWWJ, MTAP etc.