IDHashGAN: Deep Hashing With Generative Adversarial Nets for Incomplete Data Retrieval

Liming Xu, Xianhua Zeng, Weisheng Li, and Ling Bai

***Abstract*—Benefiting from low storage costs and high retrieval efficiency, hash learning has been a widely adopted technology for approximating nearest neighbor in large-scale data retrieval. Deep learning to hash greatly improves image retrieval performance by integrating feature learning and hash coding into an end-to-end framework. However, subject to application scope, most existing deep hashing methods only apply to retrieval of complete data and have undesirable results when retrieving incomplete but valuable data. In this paper we propose IDHashGAN, a novel deep hashing model with generative adversarial networks to retrieve incomplete data, in which feature restoration, feature learning and hash coding are integrated into an unified end-to-end framework. The proposed model consists of four key components: (1) reconstructive and generative loss are used to generate continuous feature of incomplete data in generative network; (2) supervised manifold similarity is proposed to improve retrieval accuracy and obtain good user acceptance; (3) adversarial and classified loss are designed to distinguish authenticity and similarity in discriminative network; and (4) encoding and quantization loss are adopted to preserve similarity and control hash quality. Extensive experiments on benchmark datasets show that IDHashGAN is competitive on complete dataset and yields substantial boosts of 70% on incomplete datasets compared to state-of-the-art hashing methods.**

***Index Terms*—Generative adversarial nets, hash learning, incomplete data, supervised manifold similarity.**

I. Introduction

OVER the last decade, we have witnessed an explosive growth of multimedia with the rapid development of social media, which has attracted increasing attention to approximate nearest neighbor retrieval due to its computational efficiency and search quality. Apart from the traditional indexing methods [1], [2], another advantageous approximate nearest neighbor technology is hashing [3], which maps high-dimensional data into compact binary codes using hash functions. Thus far, a series of hashing methods have been proposed, including classical hashing models based on hand-crafted features [4]–[9] and deep hashing models based on the machine learning paradigm [10]–[18] such as deep learning and adversarial learning. The former seeks hash functions by learning hand-crafted features and encoding their quantization to hash codes successively, but resulting in sub-optimal solution. The latter uses deep networks to learn feature representation and hash code simultaneously. Moreover, some [19], [20] use generative adversarial network to improve the performance of deep hashing.

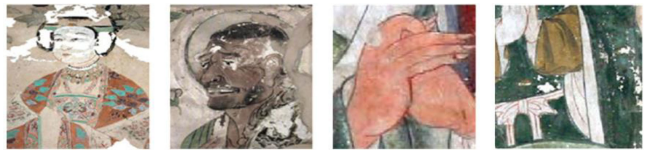

Fig. 1. The examples of damaged work of art, which are chosen from Mogao Grottoes of Dunhuang, China. HashNet [28] only achieves 44.7% MAP, and ours achieves 78.3% MAP.

Although many recent handcrafted- and deep learning-based methods have improved retrieval performance, there are still a main challenge, i.e., unacceptable results when retrieving incomplete data. Then, most of existing hashing methods are only applicable to the retrieval of complete data without taking into account the presence of great quantity of incomplete and complete data in real-world applications. To address these problem, we, in this paper, propose a novel deep hash method which is capable of handing both incomplete and complete image.

Incomplete data such as damaged work of art, blurred face images and discontinuous audio or video also play an important role in people's lives, especially in the field of cultural relic protection and criminal investigation. Incomplete data tends to be of low-resolution, defocused or motion blur and partial missing, which means that this kind of retrieval is different from that of complete data. As shown in Fig. 1, when retrieving these damaged images using state-of-the-art deep hashing method, it achieves low precision. In contrast, our proposed IDHashGAN can obtain relatively high precision compared with other comparisons.

Currently, this Mogao Grottoes dataset is not public and not commonly used, and it has never been used to verify the performance of hashing algorithms, so we add the mask to availably complete image dataset to build an incomplete dataset since there is no public incomplete image set yet. Then, the missing feature

Manuscript received January 14, 2019; revised August 29, 2019, October 11, 2019, and January 14, 2020; accepted January 12, 2021. Date of publication January 28, 2021; date of current version February 8, 2022. This work was supported in part by the National Natural Science Foundation of China under Grants 61672120 and 62076044, in part by the Natural Science Foundation of Chongqing, China under Grant cstc2019jcyj-zdxm0011, in part by the Doctoral Innovative Talents Project of Chongqing University of Posts and Telecommunications under Grant BYJS201812, and in part by the Doctoral Research Innovation Project of Chongqing under Grant CYB19173. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mohammed Daoudi. *(Corresponding author: Xianhua Zeng.)*

The authors are with the Chongqing Key Laboratory of Image Cognition, College of Computer Science and Technology, Chongqing University of Posts and Telecommunication, Chongqing 400065, China (e-mail: xulimmail@gmail.com; zengxh@cqupt.edu.cn; liws@cqupt.edu.cn; battylingb@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMM.2021.3054503.

Digital Object Identifier 10.1109/TMM.2021.3054503

of incomplete image will be restored by generative network. Finally, hash network will be utilized to learn the semantic feature of restored and original images, and generate hash codes. The main contributions of this paper can be summarized as follows:

- We propose IDHashGAN for incomplete data retrieval, in which feature restoration, feature learning and hash coding are seamlessly incorporated in an unified end-to-end framework.
- Supervised manifold similarity is proposed to obtain satisfactory retrieval performance and user acceptance with theoretical guarantee.
- The trained IDHashGAN can search both incomplete and complete images. Further, there is a non-negligible gap between the restored and real images, which augments training data to alleviate the overfitting problem.
- IDHashGAN performs well on baseline datasets and it exhibits great competitiveness compared to recent state-of-the-art deep learning-based algorithms.

The remainder of this paper is organized as follows. Related work is introduced in Section II. Deep architecture of the proposed IDHashGAN is given in Section III. Algorithm optimization is shown in Section IV. Theoretical analysis of supervised manifold and training behavior are included in Section V. Experimental results and comparisons are presented in Section VI. Finally, the conclusions are presented in Section VII.

II. Related Work

The existing hashing methods basically include unsupervised, supervised and semi-supervised methods. Unsupervised hashing [4]–[6], [21]–[23] learn the compact binary code whose distance is related to the similarity relationship of the original input data. Supervised hashing [5], [24]–[26] use samples with tag information to learn a metric function that can reflect semantic relationship among samples. Thus, the closer the samples are, the more similar they are and vice versa. However, it is difficult to obtain a large amount of tag information and always possible to obtain a partial tag. Semi-supervised hashing [7], [27] use only partial tag information to train the hash function, so it saves training time and improves the performance compared to the unsupervised hashing.

Deep learning has achieved remarkable results in the field of computer vision, and a great number of deep learning methods have been derived in the hash domain. CNNH [11] adopts a two-stage strategy to learn hash codes and image representations successively. NINH [13] incorporates the two-stage learning into an end-to-end network to learn feature representation and hash coding simultaneously. DNH [15] improves NINH by controlling quantization loss and preserving similarity with principled pairwise cross-entropy loss. DSH [18] constraints the output of deep network and makes it close to binary codes, meanwhile stabilizes the training by maintaining the gradient in {-1,+1}. Moreover, HashNet [28] points out that the similarity information is sparse, and it introduces balanced weight to address data imbalance problem. DCH [29] designs a pairwise cross-entropy loss using Cauchy distribution and achieves efficient and effective Hamming space retrieval.

The generation of realistic and continuous data has also made great progress since Generative Adversarial Networks (GANs) [30] was proposed. GANs is defined as a two-player game with the purpose of fitting generative distribution to true distribution, but it always suffers from unstable training and gradient vanishing. DCGAN [31] has made some changes to decrease the number of parameter and stabilize training but failed to cure the root. W-GANs [32] solves the problems of gradient vanishing and tiny overlap between generative and real distribution. GANs has also been applied into hash retrieval. IR-GAN [33] provides an information retrieval model with unifying GANs and results in considerable precision. DSH-GAN [19] and HashGAN [34] learn hash codes on both synthetic and real images to overcome the problem of distribution heterogeneity, but involve the original problems of GANs such as unstable training into hash learning. BGAN [35] adjusts input characteristics by restricting the input of GANs to binary codes, which yields high-quality hash codes. WeGAN [36] introduces clustering information and uncertain relationship between images and labels, and passes this relationship among images to generate hash codes.

Given that almost all of the above approaches focus on retrieving complete data with single supervised information. In this work, we firstly use GANs to restore the missing feature, then use proposed supervised manifold similarity to preserve the semantic correlation during learning the compact binary hash codes of incomplete and complete data. These two improvement constitute the proposed IDHashGAN.

III. The Proposed Method

In this section, we will introduce our method in details, including formulations, deep architecture and objective function. As shown in Fig. 2, the whole framework of IDHashGAN consists of three parts: generative, discriminative and hash network. The generative network is used to restore the missing feature of incomplete data with reconstructive and generative loss. The discriminative network, which is driven by adversarial and classified loss, is designed to distinguish whether the output image is real or restored, and judge whether those two images are similar or not. The hash network is utilized to encode raw and restored images into compact binary code with encoding and quantitative loss.

A. Notations and Problem Definition

It is necessary to introduce the notations and problem definition firstly. Bold uppercase letters, such as $\mathbf{X}$, represent matrices; lowercase letters, such as y, denote vectors. Given a training dataset $\mathbf{X} = [x_{\mathbf{1}}, x_{\mathbf{2}}, \ldots, x_{\mathbf{N}}] \in R^{d \times N}$, where $x_n \in R^d (1 \leq d \leq N)$ represented by d-dimensional feature vector is the n-th sample in $\mathbf{X}$. Our goal is to learn nonlinear hash function $h_i = F(x_i) \in \{-1, 1\}^k$, which should preserve semantic similarity with given supervised manifold similarity set $S = \{s_{ij}\}$.

Definition 1. Supervised manifold similarity: The similarity label $s_{ij} = 1$ when point x_i and point x_j are similar both on

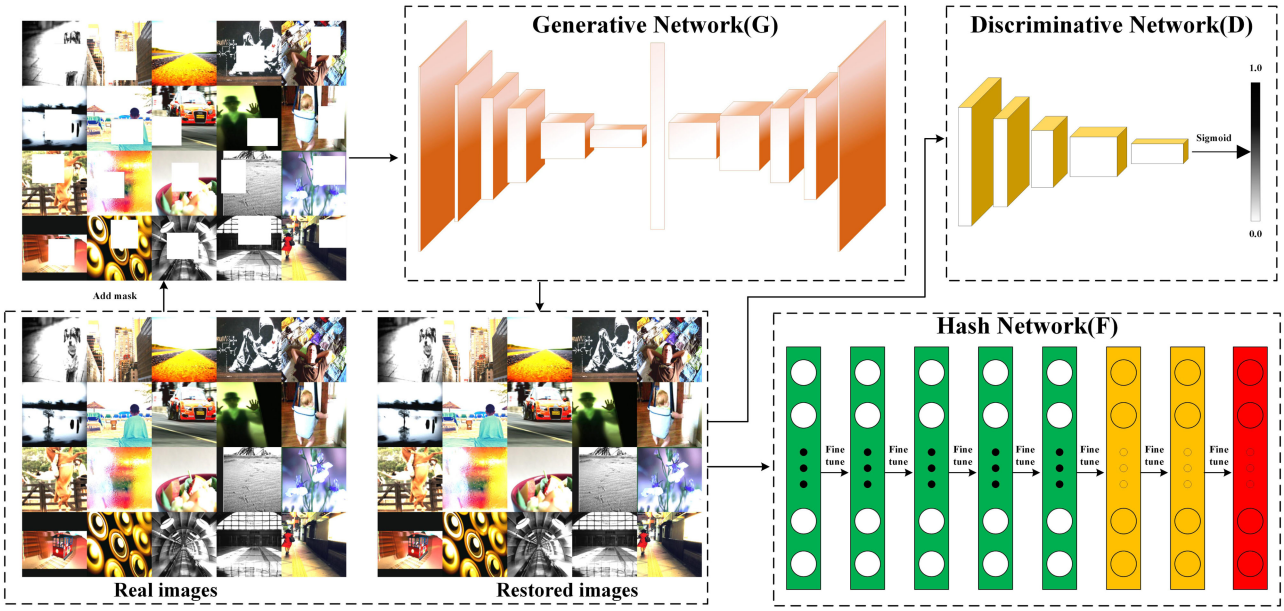

Fig. 2. The overall framework of our proposed IDHashGAN. It is worth mentioning that our goal is retrieval, meaning that the core of IDHashGAN model is to embed feature restoration into retrieval instead of retrieving after restoration. *Best viewed in color*.

pointwise (e.g. label) and pairwise (e.g. neighbor) manifold, otherwise $s_{ij} = 0$. Supervised manifold similarity is defined as:

$$\mathbf{S}_M = \mathbf{S}_{po} \odot \mathbf{S}_{pa}, \tag{1}$$

where $\mathbf{S}_{po}$ and $\mathbf{S}_{pa}$ represent pointwise (e.g. label) and pairwise (e.g. neighbor) supervised matrix, and $\odot$ denotes the element-wise multiplication.

The common way of measuring similarity between two hash codes h_i and h_j are Hamming distance $dist_H(h_i, h_j)$ and inner product $\langle h_i, h_j \rangle$, and those two can be used interchangeably for two binary hash codes as $distH(h_i, h_j) = \frac{1}{2}(K - \langle h_i, h_j \rangle)$. K is length of hash codes.

B. Generative Network

The generative network is trained to restore the missing feature and contains two kinds of loss, reconstructive and generative loss. The former aims at obtaining rough outline of incomplete images and the latter is used to capture high frequency details. Further, updating generative network with reconstructive loss can build an image dataset called reconstructive dataset that has same dimension with real image dataset, which is crucial for stabilizing training and avoiding mode collapses [37]. Following [37], the reconstructive loss is defined as

$$L_R = \frac{1}{2}||M \odot (G(\mathbf{X}, \mathbf{M}) - \mathbf{X})||_2^2, \tag{2}$$

where $G(\mathbf{X}, \mathbf{M})$ is restored image output by generative network driven by reconstructive loss. $\mathbf{X} = [x_1, x_2, \ldots, x_N]$ denotes the input image dataset and $\mathbf{M} = [m_1, m_2, \ldots, m_N]$ is mask set which is used to construct an incomplete image dataset since there is no public incomplete image dataset yet.

Eq. (2) can be written as Eq. (3) in the form of cross entropy as those two forms have the same solution in the task of binary classification.

$$\begin{aligned} L_R = & -E_{x \sim p_r(x)} \log G(x, m) \\ & - E_{x \sim p_f(x)} \log(1 - G(x, m)), \end{aligned} \tag{3}$$

where p_r and p_f denote the distribution of real and fake samples, respectively. Note that all samples generated by generative network parameterized with θ_g are regarded as fake in discriminative network D. So Eq. (3) can be estimated as:

$$L_R = E_{x \sim p_{re}(x)} \log(1 - D(G(x, m))), \tag{4}$$

where p_f is replaced with p_{re} which is called reconstructive distribution for better understanding.

Combining Eq. (4) and the generative loss in GANs, and introducing a weighted coefficient, the joint loss function of generative networks can be represented as:

$$\begin{aligned} L_G = & (1 - \alpha) E_{x \sim p_{re}(x)} \log(1 - D(G(x))) \\ & + \alpha E_{x \sim p_g(x)} \log(1 - D(G(x, m))), \end{aligned} \tag{5}$$

where $p_{re}(x)$ and $p_g(x)$ represent reconstructive distribution and prior distribution, respectively. D and G denote the discriminative and generative network, respectively. $E(\cdot)$ denotes the expectation. α is a small weight coefficient which is used to control the proportion between reconstructive and generative loss. Generative network consists of an encoder and decoder structurally, but it is not symmetrical because the input and output are different. The specific configuration is shown in Table I

C. Discriminative Network

Discriminative network is designed to distinguish the authenticity and similarity between real and restored images. The discriminative network is driven by adversarial and classification

TABLE I
THE STRUCTURE OF GENERATIVE NETWORK

Type	Configuration
Conv1	Kernel: 64×4×4, stride: 2, pad: 1, LRN, ReLU
Conv2	Kernel: 64×4×4, stride: 2, pad: 1, LRN, ReLU
Conv3	Kernel: 128×4×4, stride: 2, pad: 1, LRN, ReLU
Conv4	Kernel: 256×4×4, stride: 2, pad: 1, LRN, ReLU
Conv5	Kernel: 512×3×3, stride: 2, pad: 1, LRN, ReLU
Conv6	Kernel: 512×4×4, stride: 1, pad: 0, LRN, ReLU
Deonv1	Kernel: 512×4×4, stride: 1, pad: 0, LRN, ReLU
Deonv2	Kernel: 256×3×3, stride: 1/2, pad: 1, LRN, ReLU
Deonv3	Kernel: 128×4×4, stride: 1/2, pad: 1, LRN, ReLU
Deonv4	Kernel: 64×4×4, stride: 1/2, pad: 1, LRN, ReLU
Deonv5	Kernel: 3×4×4, stride: 1/2, pad: 1, Tanh

TABLE II
THE STRUCTURE OF DISCRIMINATIVE NETWORK

Type	Configuration
Conv1	Kernel: 64×4×4, stride: 2, pad: 1, LRN, ReLU
Conv2	Kernel: 128×4×4, stride: 2, pad: 1, LRN, ReLU
Conv3	Kernel: 256×4×4, stride: 2, pad: 1, LRN, ReLU
Conv4	Kernel: 512×3×3, stride: 2, pad: 1, LRN, ReLU
Conv5	Kernel: 512×4×4, stride: 1, pad: 0, Sigmoid

loss. The joint discriminative loss can be expressed as

$$\begin{aligned} L_D = {} & E_{x\sim p_r(x)}[\log D(x)] \\ & + E_{x\sim p_{re}(x_{re})}[1-\log D(G(x_{re}))] \\ & + \sum_{s\in S}\{s\log C(D(x),D(x_{re})) \\ & + (1-s)\log[1-C(D(x),D(x_{re}))]\}, \end{aligned} \tag{6}$$

where D and C denote discriminator and classifier, respectively. $p_r(x)$ and $p_{re}(x_{re})$ represent real and reconstructive distribution, respectively. s is the supervised manifold similarity between real and reconstructive samples.

The first two items in Eq. (6) indicate the adversarial loss which is used to distinguish authenticity, and the last two items indicate the classified loss which is designed for pairwise classification. $D(\cdot)$ is the feature vector outputted by discriminative network. We can define classifier as

$$C(D(x),D(x_{re})) = \frac{1}{1+\exp(-D^{\mathrm{T}}(x)D(x_{re}))} \tag{7}$$

The overall framework of the discriminative network is based on full connected network as shown in Table II.

The output of last convolutional layer in discriminative network is a high-dimension feature vector, while the output of this network is a fractional value which is fixed in the range of [0,1]. That fractional value represents a joint probability that the discriminative network determines an image as real, and the pair points are similar.

The "Kernel" in Tables I and II specifies the number of convolution filters and their receptive field size which is denoted as "number of kernel × length of kernel × width of kernel"; "stride" explains the convolution stride which is the interval applied to the filters to the input; "pad" indicates the number of pixels to add to each side of the input; "LRN" indicates whether Local Response Normalization [38] is applied; "pool" indicates the downsampling factor and "ReLU," "Tanh" and "Sigmoid" denote the activation function.

TABLE III
THE STRUCTURE OF HASH NETWORK

Type	Configuration
Conv1	Kernel: 64×11×11, stride: 4, pad: 0, LRN, ReLU
Pool1	maxpool: 2×2
Conv2	Kernel: 256×5×5, stride: 1, pad: 2, LRN, ReLU
Pool2	maxpool: 2×2
Conv3	Kernel: 256×3×3, stride: 1, pad: 1, ReLU
Conv4	Kernel: 256×3×3, stride: 1, pad: 1, ReLU
Conv5	Kernel: 256×3×3, stride: 1, pad: 1, ReLU
Pool3	maxpool: 2×2
Fc6	input: 512×7×7, outputs: 4096, ReLU, dropout
Fc7	input: 4096, outputs: 4096, ReLU, dropout
Fch	input: 4096, outputs: k

D. Deep Hash Network

Following previous deep hashing methods [15]–[19], [28], we construct hash network with CNN-F [38] due to its excellent performance on feature learning and object classification. The specific configuration is shown in Table III.

Two kinds of losses, encoding and quantitative loss, are incorporated into hash network. Encoding loss, also called similarity-preserving loss, is used to preserve the similarity between similar images and dissimilarity between dissimilar images. Quantitative loss is designed to control the error in continuous relaxation.

Given a set of supervised manifold similarity labels $S=\{s_{ij}\}_{i,j=1}^{N}$, the Maximum Likelihood Estimation of the hash codes $\mathbf{H}=[h_1,h_2,\ldots,h_{2N}]$ of $2N$ training data which contain real and restored samples can be represented as:

$$\begin{aligned} \log P(\mathbf{H}|\mathbf{S}) &\propto \log P(\mathbf{S}|\mathbf{H})\log P(\mathbf{H}) \\ &= \sum_{s_{ij}} \log P(s_{ij}|h_i,h_j)P(\mathbf{H}), \end{aligned} \tag{8}$$

where $P(\mathbf{S}|\mathbf{H})$ is likelihood function, $P(\mathbf{H})$ is prior distribution and $P(s_{ij}|h_i,h_j)$ is posteriori probability of supervised manifold similarity s_{ij} for given hash code h_i and h_j.

HashNet [28] points out there is data imbalanced problem, i.e., the number of similar pairs is much smaller than that of dissimilar pairs. So a weight w_{ij} for training pairs is introduced into Eq. (8) which can be rewritten as

$$\begin{aligned} \log P(\mathbf{H}|\mathbf{S}) &\propto \log P(\mathbf{S}|\mathbf{H})\log P(\mathbf{H}) \\ &= \sum_{s_{ij}} w_{ij}\log P(s_{ij}|h_i,h_j)P(\mathbf{H}) \\ \text{s.t. } & h_i\in\{-1,\ +1\}^k,\ h_j\in\{-1,\ +1\}^k \end{aligned} \tag{9}$$

The weight w_{ij} in Eq. (9) can be defined as

$$w_{ij}=\begin{cases} |S|/|S_1|\,, & s_{ij}=1 \\ |S|/|S_0|\,, & s_{ij}=0 \end{cases}, \tag{10}$$

where S_1 and S_0 denote the number of similar and dissimilar pairs in training data. For each pair,

$$P(s_{ij}|h_i,h_j) = \begin{cases} a\left(\langle h_i,h_j\rangle\right), & s_{ij}=1 \\ 1-a\left(\langle h_i,h_j\rangle\right), & s_{ij}=0 \end{cases}$$
$$= \left[a\left(\langle h_i,h_j\rangle\right)\right]^{s_{ij}}\left[1-a\left(\langle h_i,h_j\rangle\right)\right]^{1-s_{ij}}, \quad (11)$$

where $a(h)=1/(1+e^{-h})$ is a sigmoid activation function and $\langle\cdot\rangle$ denotes the inner product. Then we can conclude that

$$\log\prod P(s_{ij}|h_i,h_j) = \log(1+e^{\langle h_i,h_j\rangle}) - \sum_{s_{ij}\in S} s_{ij}\langle h_i,h_j\rangle \quad (12)$$

The Eq. (12) indicates a posteriori probability used to infer hash codes and hash functions. Combining Eq. (12) and the definition of Hamming distance, we can obtain that the smaller the Hamming distance $dist_H(h_i,h_j)$ is, the larger the inner product $\langle h_i,h_j\rangle$ and posteriori probability $P(1|h_i,h_j)$ will be, implying that image x_i and x_j should be similar and vice versa. Hence, Eq. (12) can be a reasonable explanation to hash learning. For notion brevity, we define encoding loss L_{F1} as:

$$L_{F1} = \sum_{s_{ij}\in S} w_{ij}\log(1+e^{\langle h_i,h_j\rangle}) - w_{ij}s_{ij}\langle h_i,h_j\rangle \quad (13)$$

As we all known, it is challenging to discrete optimization of Eq. (9) with the binary constrains $h_{i,j}\in\{-1,\ +1\}^k$ and continuous relaxation $h_{i,j}\in[-1,\ +1]^k$ applied to the binary constraints for ease of optimization, but resulting in sub-optimal problem. The bimodal Gaussian prior is adopted to control the quantization error of continuous relaxation and close the gap between Hamming distance and cosine distance for learning high-quality hash codes. For discrete hash codes $h_i\in\mathbf{H}=[h_1,h_2,\ldots,h_{2N}]$, the bimodal Gaussian prior is defined as:

$$p(h_i) = \frac{1}{2\varepsilon}\exp\left(-\frac{|||h_i|-1||_1}{\varepsilon}\right) \quad (14)$$

We can deduce the quantization error according to Taylor expansion from Eq. (14) as:

$$L_{F2} = \sum_{i=1}^{2N} |||h_i|-1||_1 \quad (15)$$

Replacing the absolute function $|x|$ with function $logcosh(x)$ according to smooth surrogate [15], then combining Eq. (13) and Eq. (15), we can denote the joint hash loss function as:

$$L_F = \sum_{s_{ij}\in S} w_{ij}\log(1+e^{\langle h_i,h_j\rangle}) - w_{ij}s_{ij}\langle h_i,h_j\rangle + \beta\sum_{i=1}^{2N}\log\cosh(h_i-1) \quad (16)$$

E. Objective Function

We merge the above three cost functions together, i.e., joint generative loss L_G, joint discriminative loss L_D and joint hashing loss L_F, with hyper-parameter λ to construct the whole optimization of this three-player game, which can be denoted as:

$$\min_{G,F}\max_{D}\ \lambda[L_G+L_D]+L_F \quad (17)$$

Algorithm 1: The Learning Algorithm of IDHashGAN.

Input:
Training set X; Random mask M; Length of code K.
Output
Restored Image Set; Hash code B; Network D, G and F with the parameters θ_d, θ_g and θ_f, respectively.
Initialization
mini-batch size n, total iteration: I, pre-train iteration: $I_{pre-train}$.

```
for iter < I do
  Sample a batch of images X from real image set;
  Add random mask to each image in the mini-batch;
  if iter < I_pre-train then
    Update the generative network G by descending along its stochastic gradient using Eq. (18);
    Save the reconstructive dataset restored by generative network;
  else
    Sample from the reconstructive dataset in mini-batch;
    Update the discriminative network D by ascending along its stochastic gradient using Eq. (19);
    Update the generative network G by descending along its stochastic gradient using Eq. (21);
    Update the hash network F by descending along its stochastic gradient using Eq. (22);
  end if
end for
```

Both weight coefficient α in Eq. (5) and λ in Eq. (17) are designed to ensure high precision retrieval instead of high quality recovery because our goal is retrieval rather than restoration.

IV. Algorithm Optimization

It can be proved that the objective function Eq. (17) is not convex, so we adopt the mini-batch gradient descent method and alternating learning strategy to learn parameters, i.e., update one parameter each time with other parameters fixed and optimize the model iteratively until loss converges or preset maximum iterations is reached. The learning procedure of our algorithm is summarized in Algorithm 1.

A. Pre-Training Generative Network

The generative network driven by reconstructive loss can generate the rough outline of incomplete images but lack of high frequent details. More importantly, it provides the same dimension reconstructive dataset as real dataset, which plays the key role in avoiding mode collapse and stabilizing training.

Make θ_d and θ_f fixed, then update θ_g by descending its stochastic gradient:

$$\nabla_{\theta_g}\frac{1}{n}\sum_{i=1}^{n}\alpha\log(1-D(G(x^{(i)},m))) \quad (18)$$

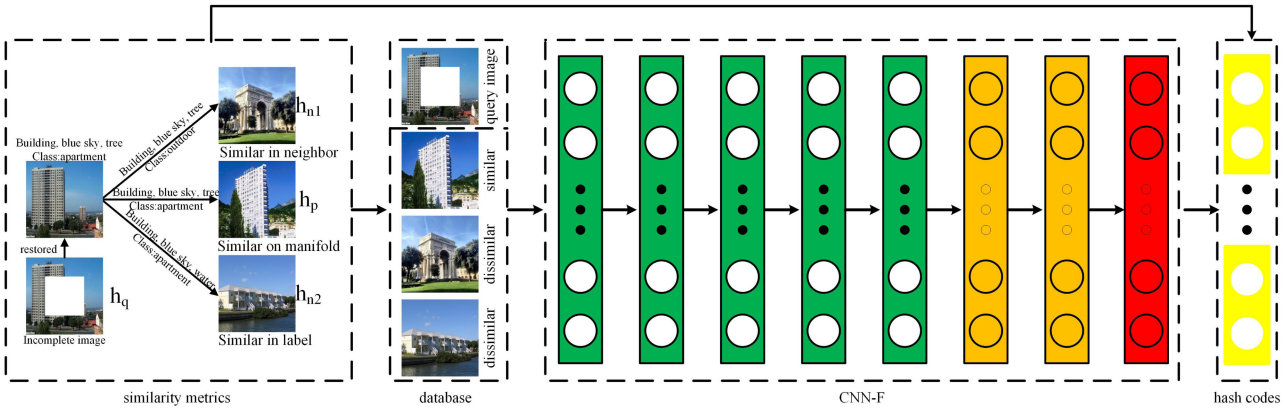

Fig. 3. Supervised manifold similarity in our proposed IDHashGAN. *Best viewed in color.*

B. Updating Discriminative Network

Make other parameters fixed and update the parameters θ_d. The deep neural network parameter θ_d is updated by ADAM (Adaptive Moment Estimation, ADAM) [39] with back-propagation mechanism. The derivation of Eq. (3) with respect to θ_d can be represented as:

$$\nabla_{\theta_d}\frac{\lambda}{n}\sum_{i=1}^{n}\Bigg\{\log D(x^{(i)})+\log(1-D(G(x_{re}^{(i)})))+\sum_{s_{ii}\in S}s_{ii}\log C(D(x^{(i)}),D(x_{re}^{(i)}))+(1-s_{ii})\log[1-C(D(x^{(i)}),D(x_{re}^{(i)}))]\Bigg\} \tag{19}$$

where $x^{(i)}$ and $x_{re}^{(i)}$ denote the i-th original and reconstructed image in mini-batch original and reconstructed image dataset, respectively. s_{ii} is supervised manifold similarity between $x^{(i)}$ and $x_{re}^{(i)}$.

C. Updating Generative Network

We also train the neural network parameter θ_g using ADAM algorithm. Step 5–6 in Algorithm 1 deal with incomplete images and produce a reconstructive dataset. Then the generative network samples from the reconstructive dataset instead of low-dimension noise. So we can conclude that $G(x_{re})=G(x,m)$ when the reconstructive loss converges [37]. Then Eq. (5) can be simplified as:

$$L_G=E_{x\sim p_{re}(x)}[\log(1-D(G(x,m)))] \tag{20}$$

When θ_d and θ_f are fixed, we update θ_g by descending generator's gradient:

$$\nabla_{\theta_g}\frac{\lambda}{n}\sum_{i=1}^{n}\log(1-D(G(x_{re}^{(i)}))) \tag{21}$$

D. Updating Hash Network

We do not choose ADAM but SGD (Stochastic Gradient Descent, SGD) [40] to update hash network because this network is not trained from scratch but copied from the pre-trained CNN-F which needs a smaller learning rate. When θ_d and θ_g are fixed, the θ_f will be updated by descending gradient:

$$\nabla_{\theta_f}\frac{1}{n}\sum_{i,j=1}^{n}\{w_{ij}\log(1+e^{\langle h_i,h_j\rangle})-w_{ij}s_{ij}\langle h_i,h_j\rangle+\beta\log\cosh(h_i-1)\} \tag{22}$$

V. Detailed Theoretical Analysis

In this section, we will provide the detailed description and derivation on supervised manifold similarity and training behavior. As shown in Fig. 3, supervised manifold similarity can preserve semantic correlation among instances.

A. Supervised Manifold Similarity

According to triplet negative log likelihoods [41] which can be utilized to measure relationship among three instances, i.e., query X_q, positive X_p and negative instance X_n. Then, Eq. (8) can be represented equally as:

$$p(S|Hq,Hp,Hn)=\prod p((h_q,h_p,h_{n1},h_{n2})|Hq,Hp,Hn), \tag{23}$$

with

$$p((h_q,h_p,h_{n1},h_{n2})|Hq,Hp,Hn)=a(\langle h_q,h_p\rangle-\langle h_q,h_{n1}\rangle-\langle h_q,h_{n2}\rangle-r) \tag{24}$$

where S represents the similarity metric(e.g. label, neighbor or supervised manifold similarity). a denotes the activation function as defined before. r is a threshold margin of nearest neighbor which is used to enforce distance between positive and negative instances. H_q, H_p and H_n denote the hash code of query, positive and negative instance outputted by hash network, respectively. Note that both h_{n1} and h_{n2} are the hash code of negative instances.

As shown in Fig. 3, the query image h_q is similar with h_p, but it is dissimilar with h_{n1} or h_{n2} in database. Therefore, the triplet loss with the proposed supervised manifold similarity can

be formulated as:

$$\begin{aligned} J1 =& \log p(X|Hq, Hp, Hn) \\ =& \sum \log p((h_q, h_p, h_{n1}, h_{n2})|Hq, Hp, Hn) \\ =& \sum \log(1+\exp(\langle h_q, h_p\rangle - \langle h_q, h_{n1}\rangle - \langle h_q, h_{n2}\rangle - r)) \\ & - (\langle h_q, h_p\rangle - \langle h_q, h_{n1}\rangle - \langle h_q, h_{n2}\rangle - r) \end{aligned} \tag{25}$$

Analogously, the triplet loss with label similarity and neighbor similarity can be formulated as Eq. (26) and Eq. (27), respectively.

$$\begin{aligned} J2 =& \log p(X|Hq, Hp, Hn) \\ =& \sum \log p((h_q, h_p, h_{n1}, h_{n2})|Hq, Hp, Hn) \\ =& \sum \log(1+\exp(\langle h_q, h_p\rangle + \langle h_q, h_{n1}\rangle - \langle h_q, h_{n2}\rangle)) \\ & - (\langle h_q, h_p\rangle + \langle h_q, h_{n1}\rangle - \langle h_q, h_{n2}\rangle) \end{aligned} \tag{26}$$

Eq. (26) does not have the item of r since r is unrelated to the label similarity.

$$\begin{aligned} J3 =& \log p(X|Hq, Hp, Hn) \\ =& \sum \log p((h_q, h_p, h_{n1}, h_{n2})|Hq, Hp, Hn) \\ =& \sum \log(1+\exp(\langle h_q, h_p\rangle + \langle h_q, h_{n2}\rangle - \langle h_q, h_{n1}\rangle - r)) \\ & - (\langle h_q, h_p\rangle + \langle h_q, h_{n2}\rangle - \langle h_q, h_{n1}\rangle - r) \end{aligned} \tag{27}$$

Comparing Eq. (25) with Eq. (26) or Eq. (27), we can clearly conclude that optimizing the loss $J1$ will reduce the Hamming distance between the query instance h_q and the positive instance h_p while increasing the Hamming distance between h_q and negative instance h_{n1}/h_{n2}. So the supervised manifold similarity will preserve the semantic correlation among instances during the optimization.

B. The Analysis on Training Behavior

GANs [30] has provided the whole solution about non-convex function with multiple parameters, but it fails to illustrate derivative of specific parameters. Moreover, all the losses defined above, i.e., joint generative loss, joint discriminative loss, similarity preserving and quantization loss, are pairwise loss. We now show that all the losses can be optimized efficiently through the standard back-propagation mechanism in an end-to-end framework.

As mentioned earlier, the cost function of discriminative network can be represented as Eq. (28) by combining adversarial and classified loss.

$$\begin{aligned} L_D =& E_{x\sim p_r(x)}[\log D(x)] \\ & + E_{x\sim p_{re}(x_{re})}[1 - \log D(G(x_{re}))] \\ & + \sum_{s\in S}\{s\log C(D(x), D(x_{re})) \\ & + (1-s)\log[1 - C(D(x), D(x_{re}))]\}, \end{aligned} \tag{28}$$

For notation brevity, we define L_{D1} and L_{D2} as adversarial and classified loss, respectively. Then, the gradient of parameters in discriminative network can be denoted as:

$$\frac{\partial L_D}{\partial W_k^l} = \left(\frac{\partial L_{D1}}{\partial \hat{d}_{ik}^l} + \frac{\partial L_{D2}}{\partial \hat{d}_{ik}^l}\right)\frac{\partial \hat{d}_{ik}^l}{\partial W_k^l} = \delta_{ik}^l d_i^{l-1}, \tag{29}$$

where $d_{ik}^{l\,\wedge} = W_k^l x_{ik}^{l-1} + b_i^{l-1}$ is the output of the k-th neural unit in the l-th layer before activation $a(\cdot)$. Since our discriminative network is built up on full convolution network, so $d_{ik}^{l\,\wedge} = W_k^l x_{ik}^{l-1}$ where W_k^l represents the weight of the k-th neural unit in the l-th layer of discriminative network. The derivation of parameters of L_{D1} and L_{D2} can be expressed as Eq. (30) and Eq. (31) (on the top of next page), respectively.

$$\frac{\partial L_{D1}}{\partial \hat{d}_{ik}^l} = \frac{1}{n}\sum_{i,j=1}^{n}\left[\frac{1}{d_i \ln 2}\nabla a^l(\hat{d}_{ik}^l) - \lambda\frac{1}{d_j \ln 2}\nabla a^l(\hat{d}_{jk}^l)\right], \tag{30}$$

where d_i and d_j represent feature maps of real and restored images in discriminative network, respectively. n is the batch-size and the term $\nabla a^l(\cdot)$ is the derivative of the l-th activation function. $q = \exp(-d_i{}^{\mathrm{T}} d_j)\nabla a^l(d_{ik}^{l\,\wedge})\nabla a^l(d_{jk}^{l\,\wedge})$ and $p = 1 + \exp(-d_i^{\mathrm{T}} d_j)$ are intermediate variables.

$$\begin{aligned} \frac{\partial L_{D2}}{\partial \hat{d}_{ik}^l} =& \frac{1}{n}\sum_{i,j=1}^{n}\sum_{s_{ij}\in S}\left[s_{ij}\frac{1}{C(d_i, d_j)\ln 2}\right. \\ & \left. + (1-s_{ij})\frac{1}{[1-C(d_i, d_j)]\ln 2}\right] p^{-2} q \end{aligned} \tag{31}$$

$$\begin{aligned} \delta_{ik}^l =& \frac{1}{n}\sum_{i,j=1}^{n}\sum_{s_{ij}\in S}\left[(1-s_{ij})\frac{1}{1-C(d_i, d_j)}\right. \\ & \left. + s_{ij}\frac{1}{C(d_i, d_j)}\right] p^{-2} q \end{aligned} \tag{32}$$

Combining Eq. (30) and Eq. (31), we can conclude Eq. (32) which indicates the pairwise residual term that measures how much the k-th neural unit in the l-th layer is responsible for the error of pair points in the network output.

Combining reconstructive and generative loss, the joint generative loss can be represented as:

$$\begin{aligned} L_G =& (1-\alpha)E_{x\sim p_{re}(x)}\log(1 - D(G(x))) \\ & + \alpha E_{x\sim p_g(x)}[\log(1 - D(G(x, m)))] \end{aligned} \tag{33}$$

We can conclude $G(x_{re}) = G(x, m)$ after reconstructive sampling [37] and then simplify Eq. (33) as:

$$L_G = E_{x\sim p_{re}(x)}[\log(1 - D(G(x)))] \tag{34}$$

Analogously, we derive Eq. (34) with respect to parameters W and can conclude:

$$\frac{\partial L_G}{\partial W_k^l} = \frac{\partial L_G}{\partial \hat{g}_{ik}^l}\frac{\partial \hat{g}_{ik}^l}{\partial W_k^l} = \delta_{ik}^l g_i^{l-1}, \tag{35}$$

where δ_{ik}^l has the definition as:

$$\delta_{ik}^l = \frac{1}{n}\sum_{i=1}^{n}\left[\frac{1}{(1-g_i)\ln 2}\nabla a(\hat{g}_{ik}^l)\right] \tag{36}$$

In the process of updating generative network, we only need to descend the gradient in Eq. (35), which is pointwise loss and can be updated by standard back-propagation mechanism.

Combining similarity preserving and quantization loss, the joint loss of hash network can be represented as:

$$L_F = \sum_{s_{ij}\in S} w_{ij}\log(1+\exp(\langle h_i, h_j\rangle)) - \sum_{s_{ij}\in S} w_{ij}s_{ij}\langle h_i, h_j\rangle + \beta\sum_{i=1}^{2N}\log\cosh(h_i-1) \tag{37}$$

For notion brevity, we also define L_{F1} and L_{F2} as similarity preserving loss and quantization loss, respectively. Then the gradient of parameters in hash network can be denoted as:

$$\frac{\partial L_F}{\partial W_k^l} = \sum_{s_{ij}\in S}\left(\frac{\partial L_{F1}}{\partial \hat{h}_{ik}^l} + \beta\frac{\partial L_{F2}}{\partial \hat{h}_{ik}^l}\right)\frac{\partial \hat{h}_{ik}^l}{\partial W_k^l} = \delta_{ik}^l h_i^{l-1}, \tag{38}$$

where δ_{ik}^l is defined by Eq. (39).

$$\delta_{ik}^l = \sum_{i,j=1}^{N}\sum_{s_{ij}\in S}\left\{w_{ij}\left[\frac{1}{1+\exp(\langle h_i^l, h_j^l\rangle)} - s_{ij}\right]h_{\mathrm{j}k}^l\nabla a^l(\hat{h}_{ik}^l) + \beta\tanh(|h_{ik}^l-1|)\mathrm{sgn}(h_{ik}^l)\nabla a^l(\hat{h}_{ik}^l)\right\} \tag{39}$$

Clearly, h_i^l and h_j^l are outputs of real and restored images in the l-th layer of hash network, respectively. For a hidden unit k in the l-th layer, we compute the residual δ_{ik}^l based on a weighted average of the errors of all the units in the l-th layer that involve x_{ik}^l as an input, which is just consistent with standard BP procedure,

$$\delta_{ik}^{l-1} = \left(\sum_{u=1}^{u_l}\delta_{iu}^l W_{uk}^l\right)\nabla a^l(\hat{x}_{ik}^l), \tag{40}$$

where u_l denotes the number of hidden units in the l-th layer and x_{ik}^l is the output of the l-th layer before activation, e.g., d_{ik}^l, g_{ik}^l and h_{ik}^l described above. It means that the residuals of Eq. (35) and Eq. (38) in all layers can be simply updated by back-propagation.

It is evident that the residual term of Eq. (32) (on the top of next page) is more complex than that of Eq. (35) or Eq. (38) because it involves two different datasets. d_i and d_j represent feature maps of real and restored images in discriminative network. In our unified framework, we can regard d_i and d_j as the feature maps of augmented dataset which includes real and restored image set. So the term $\nabla a^l(d_{ik}^{l\,\wedge})\nabla a^l(d_{jk}^{l\,\wedge})$ in Eq. (32) can be replaced with $[\nabla a^l(d_{ik}^{l\,\wedge})]^2 (1\le i\le 2n)$, meaning that standard BP mechanism can be achieved by this change.

The training time with different hard-wares and different training ways are shown in Table IV, and these results are consistent with our theoretical analysis.

The figures of second and third column indicate the time required to complete an iteration, the figures of fourth and fifth column indicate the whole training time of corresponding network. That is, 5.376 seconds is required to finish an iteration when using CPU and 13.6 minutes is required to finish the whole training of generative network when using GPU.

TABLE IV
THE TRAINING TIME OF DIFFERENT HARD WARES AND TRAINING WAYS ON INCOMPLETE CIFAR-10

Network	CPU(s/iter)	GPU(s/iter)	separated	unified
Generative Net	5.376	0.386	13.6min	13.6min
Discriminative Net	4.647	0.231	38.3min	18.6min
Hash Net	4.441	0.226	19.8min	19.9min

From Table IV, it is observed that GPU finishes a iteration in decimal second, while the CPU takes 16-20 times to complete the same iteration. Besides, when GPU is applied, training IDHashGAN in an unified way is more efficient and faster than using separated way. Note that whatever training way we adopt to train the model, generative and hash network always achieve standard BP mechanism with point loss.

VI. EXPERIMENT AND DISCUSSIONS

To verify the effectiveness of IDHashGAN, we conduct extensive evaluations on three widely-used benchmark datasets, i.e., CIFAR-10, NUS-WIDE, MIRFlickr. Then compare our method with several state-of-the-art hashing algorithms and analyze the results. Datasets and implementation are available at https://github.com/LimingXuM3.

A. Datasets and Experimental Setting

CIFAR-10 is a public dataset containing 60 000 color images in 10 classes, each has 6000 images with the size of 32 × 32. Following DCH [29], we select 100 images from each class randomly as query set, and the rest is training sample.

NUS-WIDE is a multi-label dataset containing 269 648 web images with 81 concept labels collected from Flickr. Following DSH-GAN [19], we employ the sub-set of images relates to 21 most frequent labels, where each label is associated with 5000 images. Then we randomly select 2100 images (100 per class) as test sample and the rest is trained as training sample.

MIRFlickr is a cross-modal dataset containing 25 000 images collected from Flickr, each image is associated with several textual tags. We select 20 015 points in our experiment and follow the partition ratio between training, query and database as DPSH [17] conducts.

We conduct the experiments on Intel(R) Xeon(R) Gold 6148 CPU @2.6 GHz with 20 cores and eight Tesla V100-SXM2 GPUs. The batchsize is set to 64, and then pre-training and total training iteration are fixed to 9×10^4 and 1.2×10^5, respectively. The generative and discriminative network will be updated alternately by ADAM optimization with β_1=0.9 and β_2=0.99.

We copy fine-tune convolutional layers $conv1$-$conv5$, pooling layers $pool1$-$pool3$ and fully-connected layers $fc6$-$fc7$ from the pre-trained CNN-F, then add and train hash layer fch with back-propagation. Since the fch layer is trained from scratch, we set its learning rate to be 0.5 times than that of the lower layers. α, β and λ are hyper-parameters which are used to balance the weight between different losses and will be set automatically according to the results of cross-validation from 10^{-5} to 10^2 with a multiplicative step-size 10 on different datasets.

TABLE V
MEAN AVERAGE PRECISION (MAP) OF HAMMING RANKING ON INCOMPLETE AND COMPLETE DATASETS

Methods	Incomplete/Complete CIFAR-10		Incomplete/Complete NUS-WIDE		Incomplete/Complete MIRFlickr	
	16bits	64bits	16bits	64bits	16bits	64bits
LSH [3]	- / 0.124	- / 0.124	- / 0.328	- / 0.501	- / 0.504	- / 0.550
SH [4]	- / 0.192	- / 0.215	- / 0.406	- / 0.410	- / 0.532	- / 0.527
KSH [5]	- / 0.524	- / 0.569	- / 0.551	- / 0.635	- / 0.694	- / 0.711
ITQ [6]	- / 0.354	- / 0.449	- / 0.509	- / 0.561	- / 0.553	- / 0.574
CNNH [11]	- / 0.486	- / 0.491	- / 0.570	- / 0.600	- / 0.733	- / 0.740
NINH [13]	0.334 / 0.550	0.367 / 0.586	0.398 / 0.697	0.425 / 0.715	0.438 / 0.746	0.467 / 0.773
DNH [15]	- / 0.718	- / 0.735	- / 0.774	- / 0.791	- / 0.819	- / 0.842
DPSH [17]	0.401 / 0.720	0.428 / 0.776	0.448 / 0.818	0.469 / 0.852	0.450 / 0.833	0.481 / 0.855
HashNet [28]	0.417 / 0.743	0.435 / 0.787	0.462 / 0.829	0.487 / 0.856	0.473 / 0.842	0.490 / 0.861
DCH [29]	0.477 / 0.790	0.481 / 0.793	0.482 / 0.830	0.513 / 0.855	0.489 / 0.845	0.522 / **0.870**
DSH-GAN [19]	0.214 / 0.758	0.235 / **0.803**	0.349 / **0.848**	0.362 / **0.863**	0.368 / 0.844	0.396 / 0.865
IDHashGAN (ours)	**0.732** / **0.792**	**0.745** / 0.793	**0.802** / 0.833	**0.816** / 0.859	**0.813** / **0.845**	**0.827** / 0.868

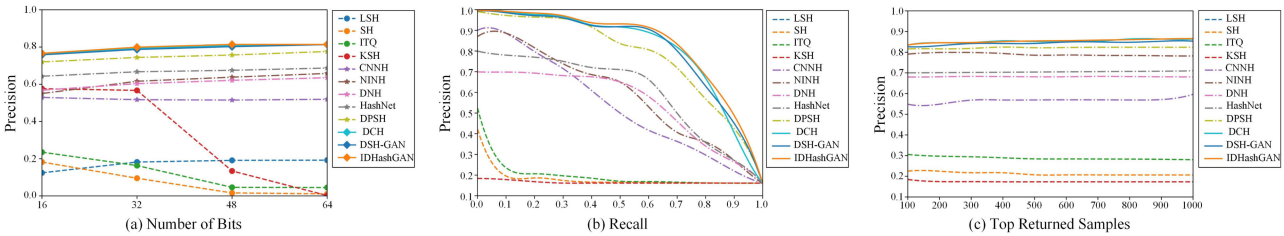

Fig. 4. Comparisons with state-of-art hashing methods on CIFAR-10 dataset. (a) Precision within Hamming radius 2 using hash lookup. (b) Precision-Recall curves with 48 bits. (c) TopN-precision curves with 48 bits. *Best viewed in color.*

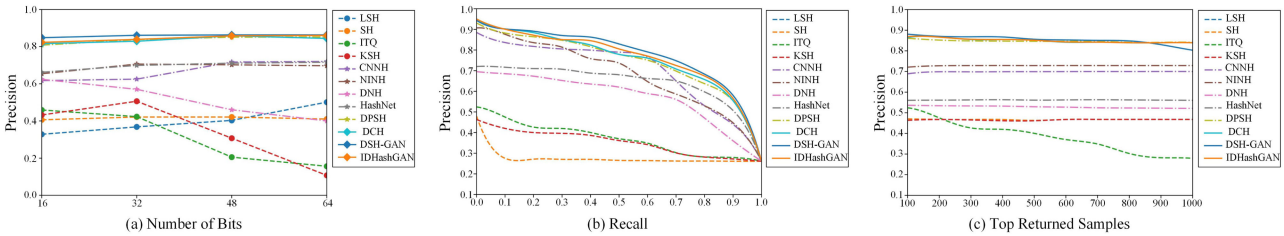

Fig. 5. Comparisons with state-of-art hashing methods on NUS-WIDE dataset. (a) Precision within Hamming radius 2 using hash lookup. (b) Precision-Recall curves with 48 bits. (c) TopN-precision curves with 48 bits. *Best viewed in color.*

B. Evaluation Protocol and Baseline Methods

In order to evaluate the retrieval performance of the proposed IDHashGAN, we adopt four widely used evaluations, including Mean Average Precision (MAP), Precision-Recall curves, Precision curves within Hamming distance 2, and Precision curves with respect to the numbers of top returned samples. Following DCH [29], DSH-GAN [19] and DPSH [17], we adopt MAP@1000 for CIFAR-10, MAP@2100 for NUS-WIDE and MAP@2000 for MIRflickr.

We generally conduct two typical tasks: complete and incomplete image hash retrieval. Firstly, We compare our method with state-of-the-art hashing methods, namely LSH [3], SH [4], KSH [5], ITQ [6], CNNH [11], NINH [13], DNH [15], DPSH [17], HashNet [28], DCH [29], DSH-GAN [19] and proposed IDHashGAN in complete dataset. Note that the first four hashing methods mentioned above use hand-crafted features and the rest are deep hashing methods. For the hand-crafted feature-based hashing methods, each image in benchmark datasets is represented by a 512-dimensional GIST vector and an officially available 500-dimensional bag-of-words vector. For the deep hashing methods, we resize all images to be 224×224 pixels and then directly exploit the raw image pixels as input. Then we compare the proposed IDHashGAN with five representative deep hashing methods, i.e., NINH, DPSH, HashNet, DSH-GAN and DCH, in incomplete image retrieval task since they have achieved better performance on complete image datasets.

C. Results on Complete and Incomplete Datasets

The comparisons on complete and incomplete datasets are listed in Table V, Figs. 4–5. All the results show that deep learning-based hashing methods outperform than the hashing with hand-crafted feature, and GANs-based hash has achieved better results. Specifically, comparing with the best baseline using traditional hand-crafted features, i.e. KSH, ours achieves absolute increments of 45.26%, 43.23% and 21.92% in average MAP for different bits on three complete datasets, respectively. Comparing with the state-of-the-art deep learning-based hashing method HashNet, ours achieves increments of 3.68%, 0.42% and 0.58%. Comparing with GANs-based hashing method, DSH-GAN, our model achieves increments of 1.60% and 0.12% on

TABLE VI
THE MAP OF HAMMING RANKING ON INCOMPLETE CIFAR-10

Manifold Similarity	16bits	32bits	48bits	64bits
$l = 1, r = 0$	0.737	0.744	0.751	0.754
$l = 1, r = 1$	0.742	0.749	0.753	0.754
$l = 1, r = 3$	0.792	**0.796**	**0.797**	**0.793**
$l = 1, r = 5$	**0.796**	0.794	0.793	0.788
$l = 0, r = 1$	0.521	0.543	0.547	0.552
$l = 0, r = 3$	0.565	0.569	0.575	0.575
$l = 0, r = 5$	0.566	0.568	0.573	0.574

complete CIFAR-10 and MIRFlickr respectively, while it decreases by 0.94% on complete NUS-WIDE.

The experiments on incomplete datasets show that deep learning-based methods can also retrieve incomplete data but yield unacceptable results. NINH, HashNet and DPSH learn partial feature which is insufficient for hash coding. DSH-GAN refers to incomplete training samples and adopts GANs to generate images and label, so the generated samples have no reference value, which results in low accuracy. The MAP on incomplete dataset shows that ours effectively captures semantic relationships and significantly improves average retrieval accuracy by 73.37%, 70.50% and 70.29% than second place of compared methods on incomplete datasets.

D. Comparison of Similarity Metrics

We integrate pointwise and pairwise supervised information to construct supervised manifold similarity, i.e., $s_{ij} = 1$ when x_i and x_j are similar both on pointwise and pairwise manifold space, otherwise $s_{ij} = 0$. The experiments about taking different supervised information on incomplete CIFAR-10 demonstrate the effectiveness of supervised manifold similarity. The results are shown in Table VI.

$l = 1$ in Table VI denotes that pointwise supervised information is taken into account and r denotes the radius of nearest neighbor. Table VI shows that our proposed supervised manifold similarity makes a higher MAP in incomplete datasets. Furthermore, it also achieves favourable acceptance, as shown in next part.

E. User Acceptance on Retrieved Results

NUS-WIDE is multi-label dataset, which always provides a high MAP result from setting $s_{ij} = 1$ as long as there is a same label. Label information belongs to coarse-grained supervised information. Many hashing methods, including DSH-GAN, achieve high MAP in this multi-label dataset but may retrieve unwanted retrieved results.

As shown in Fig. 6, the query image is 'red' car but retrieved 'white' car when using pointwise supervised information only. This result belongs to the same class, but do not meet users' needs to the maximum extent. Fig. 6 shows that IDHashGAN achieves better acceptance than baseline methods which take single supervised information on retrieved images. Although our proposed IDHashGAN performs few decrement on multi-label dataset, it achieves satisfactory acceptance and acceptable precision.

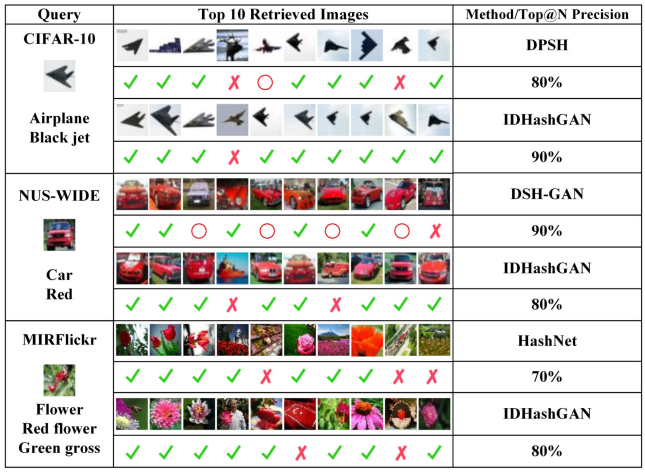

Fig. 6. The examples of top ten retrieved images and precision@10 on three baseline datasets. Red circle represents retrieved images have same class as query image but do not satisfy users' needs.

Fig. 7. The examples of restoration on incomplete MIRFlickr dataset. The left of masked images are input images and the right are restored images. The mask size is 1/4 or 1/16 times than that of real images, i.e., fractional time is 1/4 or 1/16. *Best viewed in color.*

F. Visualization of Restoration

We adopt alternating learning strategy to optimize objective function, which will yield three models, i.e., generative, discriminative and hash model. Then we can use the generative model to restore incomplete images. We add the random mask to the real images, then restore the masked images with yielded generative model. Random means random size and random location, but we set the size of mask to fractional multiple of original images to discuss the influence of different degree of missing on retrieval results. Now, we show some restoration results on MIRflickr as presented in Fig. 7, which contains large size images and has good visualization on restoration.

As shown in Fig. 7, IDHashGAN can obtain satisfactory restoration in vision on landscape. Undesirable performance may be produced from this generative model which is employed for restoring incomplete face images, since MIRflicker is a

TABLE VII
MEAN AVERAGE PRECISION (MAP) RESULTS OF IDHASHGAN AND ITS VARIANTS ON INCOMPLETE AND COMPLETE DATASETS

Methods	Incomplete/Complete CIFAR-10		Incomplete/Complete NUS-WIDE		Incomplete/Complete MIRFlickr	
	16bits	64bits	16bits	64bits	16bits	64bits
Variants-C	0.662 / 0.686	0.687 / 0.705	0.680 / 0.713	0.701 / 0.728	0.714 / 0.732	0.723 / 0.746
Variants-B	0.685 / 0.712	0.693 / 0.725	0.764 / 0.808	0.768 / 0.821	0.798 / 0.834	0.801 / 0.848
Variants-Pa	0.417 / 0.743	0.436 / 0.787	0.462 / 0.828	0.487 / 0.856	0.473 / 0.842	0.490 / 0.861
Variants-Po	0.344 / 0.616	0.381 / 0.636	0.386 / 0.648	0.402 / 0.688	0.419 / 0.672	0.439 / 0.698
IDHashGAN	**0.732 / 0.792**	**0.745 / 0.793**	**0.802 / 0.833**	**0.816 / 0.859**	**0.813 / 0.845**	**0.827 / 0.868**

mixture dataset of limited size which contains the images of various classes ranging from landscape to animals.

G. Discussion

The proposed IDHashGAN can retrieve incomplete data and improves substantially compared to the recent state-of-the-art hashing methods owing to two important perspectives: (1) IDHashGAN embeds feature restoration into hashing, which is the first attempt about incomplete data retrieval; and (2) IDHashGAN adopts supervised manifold similarity to preserve semantic correlation among instances, which address the problem of single supervised information in DPSH, HashNet, DCH and DSH-GAN.

We investigate four IDHashGAN variants: (1) IDHashGAN-C is the IDHashGAN variant without classifier in discriminative network; (2) IDHashGAN-B is the IDHashGAN variant without using quantization loss in hash network; (3) IDHashGAN-Pa is the IDHashGAN variant without pairwise supervised information; (4) IDHashGAN-Po is the IDHashGAN variant without pointwise supervised information.

From Table VII, we can see that IDHashGAN achieves an average MAP increment of 7.18%, 5.61% and 2.56% over IDHashGAN-B after introducing quantization error. IDHashGAN-B takes continuous relaxation strategy to address discrete optimization and results in sub-optimization. IDHashGAN-Po or IDHashGAN-Pa takes single supervised information into account, which ignores a few labeled samples or relation between nearest neighbor samples. Naturally, IDHashGAN-Pa incurs an average MAP decrement of 73.40%, 73.57% and 70.33% compared to IDHashGAN. A similar trend is that IDHashGAN-Po incurs average MAP decrement of 104.16%. 105.38% and 91.21% compared to the proposed IDHashGAN. Classifier in discriminative network plays the key role in distinguishing similarity between real and restored images. Table VII shows that IDHashGAN outperforms IDHashGAN-C by 15.51%, 17.17% and 14.13% in average MAP on three incomplete datasets.

One of the principles about feature restoration is that the less missing the better inpainting and vice versa, which is also applicable for incomplete data retrieval. If the size of the mask is small enough, it can achieve the retrieval accuracy of the complete data. The Table IV shows the retrieve results of incomplete data when the fractional multiple is 1/16. The fractional multiple refers to the proportion of the mask in the complete image. That is, 0 means no mask and 1/4 means the size of mask is a quarter of the full image. It is clearly that there is a non-negligible difference when retrieving incomplete and complete data.

TABLE VIII
THE MAP OF HAMMING RANKING ON INCOMPLETE NUS-WIDE WITH FRACTIONAL MULTIPLE MASK

Multiple	16bits	32bits	48bits	64bits
1/4	0.660	0.677	0.681	0.683
1/16	0.802	0.804	0.811	0.816
1/64	0.819	0.821	0.826	0.829
1/256	0.822	0.824	0.831	0.836
1/1024	0.827	0.831	0.839	0.840
0	0.833	0.849	0.857	0.859

The specific configurations listed in Table I and Table II are just the most basic configuration for generative and discriminative network when fractional multiple is 1/16, and the architectures of generative and discriminative network will change when multiple changes, i.e., the smaller the multiple is, the fewer layers the generative network and discriminative network has.

VII. CONCLUSION

This paper has presented a deep hashing method with generative adversarial nets to retrieve incomplete and complete data by introducing feature restoration into hashing methods. Then, the supervised manifold similarity is proposed to preserve the semantic correlation among instances and obtain the good user acceptance, which is demonstrated by detailed theoretical analysis. In addition, this paper has proved that the proposed IDHashGAN can be trained with standard back-propagation mechanism in an end-to-end framework. Finally, extensive experiments demonstrate that our method can yield satisfactory retrieval performance in incomplete and complete retrieval task compared to other state-of-the-art hashing methods, and the trained generative model can also restore the incomplete data. In the real world, there exists much incomplete data, including incomplete text, audio, video and irregular missing images, so we will introduce different generative methods to expand retrieval scope of incomplete data in future work.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their help.

REFERENCES

[1] A. Grigorova, F. Natale, C. Dagli, and T. Huang, "Content-based image retrieval by feature adaptation and relevance feedback," *IEEE Trans. Multimedia*, vol. 9, no. 6, pp. 1183–1192, Oct. 2007.
[2] E. Tiakas, D. Rafailidis, A. Dimou, and P. Daras, "MSIDX: Multi-sort indexing for efficient content-based image search and retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1415–1430, Oct. 2013.

[3] J. Wang, T. Zhang, J. Song, N. Sebe, and H. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.
[4] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1753–1760.
[5] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 2074–2081.
[6] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach tolearning binary codes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 7, 2013, pp. 817–824.
[7] J. Wang, S. Kumar, and S. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.
[8] Y. Lv, W. Ng, Z. Zeng, D. Yeung, and P. Chan, "Asymmetric cyclical hashing for large scale image retrieval," *IEEE Trans. Multimedia*, vol. 17, no. 8, pp. 1225–1235, Aug. 2015.
[9] Y. Jiang, J. Wang, X. Xue, and S. Chang, "Query-adaptive image search with hash codes," *IEEE Trans. Multimedia*, vol. 15, no. 2, pp. 442–453, Feb. 2013.
[10] S. Ercoli, M. Bertini, and A. Bimbo, "Compact hash codes for efficient visual descriptors retrieval in large scale databases," *IEEE Trans. Multimedia*, vol. 19, no. 11, pp. 2521–2532, Nov. 2017.
[11] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for imageretrieval via image representation learning," in *Proc. AAAI Conf. Artif. Intell.*, 2014, pp. 2156–2162.
[12] V. 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.*, 2015, pp. 2475–2483.
[13] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3270–3278.
[14] K. Lin, H. Yang, J. Hsiao, and C. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 27–35.
[15] H. Zhu, M. Long, J. Wang, and Y. Cao, "Deep hashing network for effcient similarity retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2415–2421.
[16] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang, "Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification," *IEEE Trans. Image Process.*, vol. 24, no. 12, pp. 4766–4779, Dec. 2015.
[17] W. Li, S. Wang, and W. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. Int. Joint Conf. Artif. Intell.*, 2017, pp. 1711–1717.
[18] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2064–2072.
[19] Z. Qiu, Y. Pan, T. Yao, and T. Mei, "Deep semantic hashing with generative adversarial networks," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 225–234.
[20] K. Ghasedi, F. Zheng, Y. Yang, D. Cheng, and H. Huang, "Unsupervised deep generative adversarial hashing network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 3664–3673.
[21] M. Norouzi, D. Fleet, and R. Salakhutdinov, "Hamming distance metric learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1061–1069.
[22] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1042–1050.
[23] W. Liu, J. Wang, S. Kumar, and S. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn.*, 2011, pp. 1–8.
[24] T. Ge, K. He, and J. Sun, "Graph cuts for supervised binary coding," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 250–264.
[25] F. Shen, C. Shen, W. Liu, and H. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 37–45.
[26] W. Kang, W. Li, and Z. Zhou, "Column sampling based discrete supervised hashing," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 1230–1236.
[27] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proc. Int. Conf. Comput. Vis.*, 2009, pp. 2130–2137.
[28] Z. Cao, M. Long, J. Wang, and P. Yu, "HashNet: Deep learning to hash by continuation," in *Proc. Int. Conf. Comput. Vis.*, 2017, pp. 1129–1137.
[29] Y. Cao, M. Long, B. Liu, and J. Wang, "Deep cauchy hashing for hamming space retrieval," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 5609–5618.
[30] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.
[31] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. Int. Conf. Learn. Representation.*, 2016, pp. 1–14.
[32] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[33] J. Wang *et al.*, "IRGAN: A minimax game for unifying generative and discriminative information retrieval models," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2017, pp. 515–524.
[34] Y. Cao, B. Liu, M. Long, and J. Wang, "HashGAN: Deep learning to hash with pair conditional wasserstein GAN," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 1287–1296.
[35] J. Song *et al.*, "Binary generative adversarial networks for image retrieval," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 394–401.
[36] Y. Wang *et al.*, "WeGAN: Deep image hashing with weighted generative adversarial networks," *IEEE Trans. Multimedia*, vol. 22, no. 6, pp. 1458–1469, Jun. 2020.
[37] L. Xu, X. Zeng, W. Li, and Z. Huang, "Multi-granularity generative adversarial nets with reconstructive sampling for image inpainting," *Neurocomputing*, vol. 40, no. 2, pp. 220–234, 2020.
[38] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[39] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representation.*, 2015, pp. 1–15.
[40] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proc. Int. Conf. Comput. Statist.*, 2010, pp. 148–157.
[41] X. Wang, Y. Shi, and K. Kitani, "Deep supervised hashing with triplet labels," in *Proc. Asian Conf. Comput. Vis.*, 2016, pp. 70–84.

Liming Xu received the B.S. and M.S. degrees in computer science and technology from China West Normal University, Nanchong, China, in 2014 and 2017, respectively. He is currently working toward the Ph.D. degree with the Department of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include machine learning, adversarial learning, image processing, and learning to hash.

Xianhua Zeng received the Ph.D. degree in computer software and theory from Beijing Jiaotong University, Beijing, China, in 2009. He is currently a Professor with the Chongqing Key Laboratory of Computational Intelligence, College of Computer Science and Technology, Chongqing University of Posts and Telecommunications, Chongqing, China. From 2013 to 2014, he was a Visiting Scholar with the University of Technology Sydney, Ultimo, NSW, Australia. His main research interests include medical image processing, machine learning, and data mining.

Weisheng Li graduated from the School of Electronics and Mechanical Engineering, Xidian University, Xi'an, China, in 1997, the M.S. degree from the School of Electronics and Mechanical Engineering, Xidian University, in 2000, and the Ph.D. degree from the School of Computer Science and Technology, Xidian University, in 2004. He is currently a Professor with the Chongqing University of Posts and Telecommunications, Chongqing, China. His research interests include intelligent information processing and pattern recognition.

Ling Bai received the B.S. and M.S. degrees in information and communication engineering from the Chongqing University of Posts and Telecommunications, Chongqing China, in 2014 and 2017, respectively. She is currently working toward the Ph.D. degree with the Department of Computer Science and Technology, Chongqing University of Posts and Telecommunications. Her research interests include machine learning, image processing, and pattern recognition.