Unsupervised Semantic-Preserving Adversarial Hashing for Image Search

Cheng Deng, *Member, IEEE*, Erkun Yang, Tongliang Liu, Jie Li, Wei Liu, *Senior Member, IEEE*, and Dacheng Tao, *Fellow, IEEE*

***Abstract*— Hashing plays a pivotal role in nearest-neighbor searching for large-scale image retrieval. Recently, deep learning-based hashing methods have achieved promising performance. However, most of these deep methods involve discriminative models, which require large-scale, labeled training datasets, thus hindering their real-world applications. In this paper, we propose a novel strategy to exploit the semantic similarity of the training data and design an efficient generative adversarial framework to learn binary hash codes in an unsupervised manner. Specifically, our model consists of three different neural networks: an encoder network to learn hash codes from images, a generative network to generate images from hash codes, and a discriminative network to distinguish between pairs of hash codes and images. By adversarially training these networks, we successfully learn mutually coherent encoder and generative networks, and can output efficient hash codes from the encoder network. We also propose a novel strategy, which utilizes both feature and neighbor similarities, to construct a semantic similarity matrix, then use this matrix to guide the hash code learning process. Integrating the supervision of this semantic similarity matrix into the adversarial learning framework can efficiently preserve the semantic information of training data in Hamming space. The experimental results on three widely used benchmarks show that our method not only significantly outperforms several state-of-the-art unsupervised hashing methods, but also achieves comparable performance with popular supervised hashing methods.**

***Index Terms*— Hashing, image search, adversarial learning, deep learning.**

I. Introduction

MASSIVE image data repositories with high dimensions are widespread in search engines and social networks, which has lead to renewed interest in efficient search and indexing algorithms [1]–[7]. Since exact nearest neighbor search is typically time-consuming and unsuitable for large-scale datasets, approximate nearest neighbor (ANN) search [8], [9], which attempts to find the nearest neighbor with a high probability, has been actively studied and successfully applied to many machine learning and data mining problems, such as retrieval [10]–[15], classification [16], and object recognition [17]. Nevertheless, effective and efficient ANN searching remains a challenge in contexts with large volumes and high dimensions.

Representative ANN solutions include tree-based [18], [19] and hash-based methods [8], [20]–[28]. In real-world applications, visual data are usually represented by high-dimensional features, e.g., SIFT-based bag-of-words features [29] and deep features. Traditional tree-based methods are known to suffer in high-dimensional feature space, and their performance has been theoretically shown to degrade to match that of the linear scan in many cases [30]. Therefore more attention is being paid to hash-based methods [20], [21], [23], [31]–[36], which have relative low storage cost and a fast query speed.

Traditional hashing methods are usually built using hand-crafted features and try to learn hash functions that can map the original high-dimensional representations into low-dimensional binary codes, such that the similarity to the original space can be well preserved. However, these methods, which rely on shallow learning architectures and hand-crafted features, have limited representation capacity and cannot simultaneously optimize feature learning and hash code learning, thus compromising search performance in practice.

Recently, deep learning has revolutionized computer vision, machine learning, and many other related areas [37]–[40]. Deep learning-based hashing methods [41], [42] have also been proposed for efficient ANN search, showing that deep networks can help in the more effective learning of hash codes. Usually, these methods are supervised by massive numbers of ranking pairwise [41] or triplet labels [43]. Pairwise or triplet loss functions are thus adopted to map similar points to similar hash codes in Hamming space and guide the hash function learning process. However, since the manual labeling of data is often time-consuming, laborious, and expensive, heavy reliance on the availability of copious annotated data may hinder the application of these methods in practice.

To make better use of the more widely available, unlabeled data, unsupervised deep hashing methods have also been proposed. Most recent unsupervised deep hashing methods use deep autoencoders to learn hash codes [44], [45]. These methods first map inputs to hash codes and then try to

Manuscript received May 31, 2018; revised January 17, 2019; accepted February 23, 2019. Date of publication March 13, 2019; date of current version June 20, 2019. This work was supported in part by the National Natural Science Foundation of China under Grant 61572388 and Grant 61703327, in part by the Key R&D Program-The Key Industry Innovation Chain of Shaanxi under Grant 2017ZDCXL-GY-05-04-02, Grant 2017ZDCXL-GY-05-02, and Grant 2018ZDXM-GY-176, in part by the National Key R&D Program of China under Grant 2017YFE0104100, and in part by the Australian Research Council Projects under Grant DP-180103424, Grant DE-1901014738, Grant FL-170100117, and Grant IH-180100002. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Riccardo Leonardi. *(Corresponding author: Erkun Yang.)*

C. Deng, E. Yang, and J. Li are with the School of Electronic Engineering, Xidian University, Xi'an 710071, China (e-mail: chdeng.xd@gmail.com; ekyang@stu.xidian.edu.cn; leejie@mail.xidian.edu.cn).

T. Liu and D. Tao are with the UBTECH Sydney Artificial Intelligence Centre, School of Computer Science, Faculty of Engineering and Information Technologies, The University of Sydney, Sydney, NSW 2008, Australia (e-mail: tongliang.liu@sydney.edu.au; dacheng.tao@sydney.edu.au).

W. Liu is with Tencent AI Lab, Shenzhen 518057, China (e-mail: wl2223@columbia.edu).

Digital Object Identifier 10.1109/TIP.2019.2903661

reconstruct the original inputs from the learned hash codes at pixel level. However, since natural images usually contain a lot of variability in trivial factors like location, color, and pose. Pixel-wise reconstruction may degrade the learned hash codes by focusing on these trivial variations. Other recent deep hashing methods learn hash codes by maximizing their representation capacity [46] or enforcing the similarity between rotated images and their corresponding original images [47]. However, neither of these methods can capture and preserve the semantic similarities between different instances, so may lose the opportunity to learn more accurate hash codes.

In this paper, we aim to learn compact hash codes, which can be used to believably reconstruct inputs without pixel-wise reconstruction and simultaneously preserve high-level semantic information without the use of labels. To achieve these two purposes, we first design two networks that aim to generate synthetic images from hash codes and generate hash codes from images respectively. To enable credible reconstruction, we need to match these two generative processes. Inspired by recent GAN variants [48], we design a discriminative network and propose an adversarial learning framework, in which the discriminator network is trained to distinguish pairs of hash codes and images from the first two generative networks. These two networks are trained to fool the discriminator network. By adopting a minimax game mechanism, we can successfully match these two generative processes and learn hash codes that believably reconstruct the inputs. For the second objective, we take advantage of recent, advanced deep architectures that have proven effective for extracting features with rich semantic information. Based on these deep features, we propose a novel strategy to construct a semantic similarity matrix. The hash code learning process is also designed to preserve this semantic similarity. In doing so, we can successfully capture and preserve the semantic information without labels.

By incorporating the semantic similarity-preserving objective with the adversarial hashing learning framework, we propose a method called Unsupervised ADversarial Hashing (UADH). Specifically, as illustrated in Fig. 1, UADH comprises three neural networks: (1) an encoder network to generate hash codes from real images; (2) a generative network to generate synthetic images from hash codes; and (3) a discriminative network that aims to distinguish between pairs of hash codes and images from the encoder and the generative networks, respectively. Furthermore, in order to capture and preserve the semantic information from the training dataset, we propose a novel strategy that leverages both feature and neighbor similarities to construct a similarity matrix.

Our main contributions can be summarized as follows:

- Our method can successfully align the joint distributions from the designed decoder and encoder networks and thereby learn a more representative latent space for hash codes.
- We design a novel strategy to learn semantic similarity directly from deep features, which enables our method to adopt loss functions that are usually reserved for supervised learning.
- Experimental results on three popular benchmarks demonstrate that our method not only outperforms other state-of-the-art unsupervised hashing methods, but also obtains comparable performance with representative supervised hashing methods.

The rest of this paper is organized as follows. We review the relevant literature in Section II. We present our novel unsupervised adversarial hashing for image search in Section III. Section V outlines the experiments, while by the concluding remarks are provided in Section VI.

II. Related Work

In this section, we briefly review some representative hashing approaches and provide a brief introduction to some related works in generative adversarial network (GAN) areas.

A. Hashing for Similarity Search

Recently, a variety of hashing methods have been proposed, most of which fall into two categories: data-independent hashing, also known as locality-sensitive hashing (LSH), and data-dependent hashing, also called learn-to-hash (L2H). Data-independent hashing usually learns a hash function family independently of the training dataset. One of the most popular data-independent hashing methods is LSH [8], which embeds similar data into similar binary codes with high probabilities. Many extensions have been developed following the basic LSH concept [16], [49], all of which are fundamentally built on random projection and remain unaware of the data distribution. Although these methods are guaranteed to have high collision probability for similar data items, in practice they usually need long hash bits and multiple hash tables to achieve high precisions. Therefore, the huge storage requirements of these methods may restrict their applications.

Compared to traditional data-independent hashing methods, data-dependent hashing, which learns hash functions from training data, can usually achieve comparable or even better accuracy with shorter codes. Consequently, this approach has attracted more attention. Data-dependent hashing can be further divided into two main categories: namely, supervised and unsupervised methods. For supervised hashing, hash functions are usually learned to preserve similarities extracted from both original data and labels in Hamming space. Representative supervised hashing methods include minimal loss hashing (MLH) [50], kernel-based supervised hashing (KSH) [23], adaptive hashing (AH) [51], and fast supervised hashing (FashHash) [52]. MLH aims to learn similarity-preserving hash functions that can map high-dimensional data into binary codes, the formulation of which is based on structure prediction with latent variables and a hinge-like loss function. KSH utilizes inner products to approximate the Hamming distance, and learns hash functions to project data points to compact codes, the Hamming distances of which are minimized for similar pairs and maximized for dissimilar pairs. AH tries to learn hash functions in an online manner and designs an online learning algorithm to update the hash functions iteratively using streaming data. FashHash reveals that nonlinear hash functions can perform better than linear ones

due to their powerful generalization. Specifically, FashHash adopts boosted decision trees to fit the learned binary codes. An efficient GraphCut-based block search method is also proposed to solve large-scale inference.

In this paper, we mainly focus on unsupervised hashing. Representative unsupervised hashing methods include iterative quantization (ITQ) [53], spectral hashing (SH) [22], discrete graph hashing (DGH) [54], spherical Hashing (SpH) [55], anchor graph hashing (AGH) [56], and stochastic generative hashing (SGH) [57]. ITQ tries to map original data to a low-dimensional space by means of principal component analysis (PCA) and then learn an orthogonal rotation matrix that maps the data to binary codes with minimum quantization error. SH interprets the hash code learning by means of spectral graph partitioning and then relaxes this graph partitioning problem using an efficient spectral method. DGH uses a discrete optimization framework to solve the graph hashing problem and explicitly deals with the discrete constraints, meaning that it can directly output binary codes. The objective function of DGH is similar to that of SH. SpH aims to preserve the balanced and independent properties of binary codes by exploiting a hypersphere-based hashing function, which can map spatially coherent data points to similar binary codes. AGH tries to learn compact codes by discovering the inherent neighborhood structure of the training data and constructs anchor graphs to approximate this data structure, which can also accelerate the computation of the graph's Laplacian eigenvectors. SGH adopts a generative approach to learn hash functions via the minimum description length principle. The learned hash codes can maximally compress the input data and can also be used to regenerate the inputs. A stochastic distributional gradient-based learning algorithm is also developed to avoid the optimization difficulty caused by binary constraints. However, all these methods use hand-crafted features and cannot simultaneously optimize features and hash code learning, resulting in degraded performance.

In addition to these shallow approaches, some unsupervised deep hashing methods have also been proposed [44], [45], [47]. Due to the lack of semantic labels or class information, most unsupervised deep learning-based hashing methods involve point-wise constraints. As a pioneering work, semantic hashing [45] uses Restricted Boltzmann Machines (RBMs) as an auto-encoder network to generate hash codes, and adopts a reconstruction loss to optimize the learning procedure. Deep Hashing (DH) [46] develops a neural network that seeks multiple hierarchical non-linear transformations in order to map data points to binary codes. Quantization loss occurs between the original real-valued features and the learned binary codes; this is typically a point-wise constraint. DeepBit [47] considers original images and their corresponding rotated images as similar pairs and tries to learn hash codes to preserve this similarity. Since the rotated images are generated from the original images, the pair-wise loss function is actually also a point-wise constraint. Note that since all these methods involve point-wise constraints, they cannot capture and preserve the semantic similarities between different data points, which is critical for nearest neighbor search. Thus, opportunities to learn more effective hash codes may be lost.

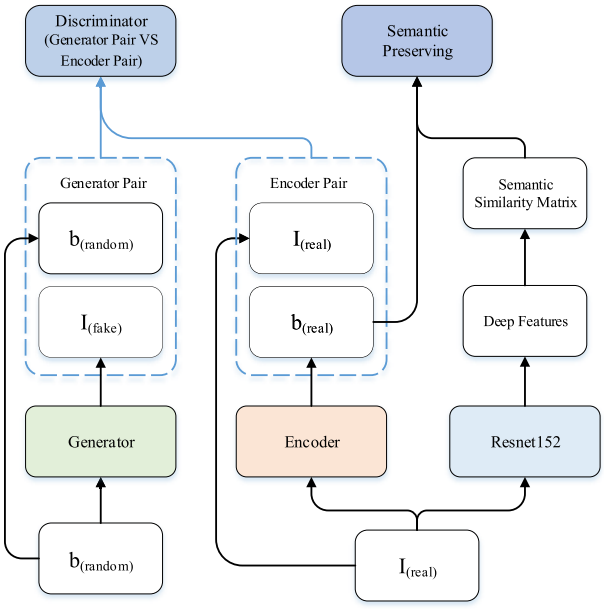


Fig. 1. The architecture of the proposed UADH consists of four main components. (1) The semantic similarity matrix learning part, which learn a semantic similarity matrix by exploiting the semantic information from deep features. (2) A encoder, which takes real images as inputs and outputs hash codes. (3) A generator, which generates nearly real images images from random hash codes. (4) A discriminator, which takes pair of images and hash codes as inputs. The architecture is trained by jointly optimizing a GAN loss and a semantic preserving loss.

B. Generative Adversarial Networks

Generative adversarial networks (GANs), first proposed by Goodfellow *et al.* [58], have been widely used in computer vision to generate realistic images [59], [60]. There are two adversaries in a GAN framework, a generator and a discriminator, which are trained using a two-player minimax game mechanism. The discriminator is trained to distinguish between real images and synthetic images, while the generator is trained to fool the discriminator. Despite their large capacity for image generation, GANs do not contain an encoder network to map images to a latent space, which is necessary for hash code learning. Recently, an approach called BGAN [61] adopted GAN for hashing. However, in addition to adversarial loss, BGAN also requires a pixel-wise reconstruction loss to enable the hash encoder learning process. Since pixel-wise loss usually results in less effective hash codes, the performance of BGAN may be degraded. In this paper, inspired by recent GAN variants [48], [48], we explicitly design an encoder network and optimize it by training a discriminator to distinguish between pairs of hash codes and images. We constrain the inputs of the encoder to be approximate binary values in order to incorporate hash code learning into the framework.

III. Unsupervised Adversarial Hashing

Suppose there are N images without labels represented as $\mathcal{I} = \{I_i | i = 1, \ldots, N\}$. Our goal is to learn a mapping function

$\mathcal{H}: \mathcal{I} \rightarrow \{0,1\}^K$, such that an input image I_i can be encoded to a K-bit binary code. sign$(\cdot)$ indicates an element-wise sign function, defined as:

$$\text{sign}(x) = \begin{cases} 1, & x \geq 0, \\ -1, & x < 0. \end{cases} \tag{1}$$

Fig. 1 shows our proposed framework. Our method includes three networks. The first is an encoder network, which outputs hash code given an image as input. The second is a generative network, which generates synthetic images with random binary codes as input. The last is a discriminative network, which aims to distinguish hash code and image pairs from the encoder and generative network. The encoder network is also optimized to preserve semantic information, represented by a semantic similarity matrix. In the reminder of this section, we first elaborate the network architecture and describe the overall loss function in detail, then introduce the construction of the semantic similarity matrix.

A. Architecture

1) Encoder Network: The encoder network maps an input image to a compact binary code. We design the encoder network based on the VGG-F architecture [62] and replace the last fully-connected layer with a new fully-connected layer that acts as a hash layer. The number of hidden units for this layer is set as the hash bit length. We denote the pairs of input image and output approximate binary code for this network as encoder pairs.

2) Generative Network: The generative network aims to generate synthetic images from latent Hamming space. We adopt a generative network similar to the generator in DCGAN [63], which has four fractionally-strided convolutions, details of which can be found in [63]. The input of this network is constrained to be approximate binary code. Pairs of input approximate binary code and output images for this network are denoted as generator pairs.

3) Discriminative Network: The objective of the discriminative network is to distinguish generator pairs from encoder pairs. Since the inputs of this network contain two different modalities, *i.e.*, images and hash codes, we first fuse information from these two modalities and then distinguish between the different samples. To do so, our discrimination network is split into three parts: an image feature fusion part, a binary feature fusion part, and a joint discrimination part. The image feature fusion part contains five convolutional layers and is used to map images to latent representations. The binary feature fusion part consists of two fully-connected layers and is used to map binary codes to latent representations. These two types of representations are then concatenated to obtain a joint representation. The discrimination part contains three fully-connected layers and is used to distinguish whether the joint representation is from the encoder network or the generative network. More detailed configuration of this network is showed in Table I, where *f.* indicates filter size, *st.* indicates stride size, *bn* means that we apply batch normalization after the activation function, and *lrelu* means we use the *lrelu* function as the activation function for the corresponding layer.

TABLE I
THE ARCHITECTURE OF THE DISCRIMINATIVE NETWORK

Subnetwork	Layer	Configuration
Image feature fusion	conv_I1	f. $32 \times 5 \times 5$; st. 1×1; lrelu
	conv_I2	f. $64 \times 4 \times 4$; st. 2×2; lrelu; bn
	conv_I3	f. $32 \times 5 \times 5$; st. 1×1; lrelu; bn
	conv_I4	f. $32 \times 5 \times 5$; st. 1×1; lrelu; bn
	conv_I5	f. $32 \times 5 \times 5$; st. 1×1; lrelu; bn
Binary feature fusion	fc_B1	256; lrelu; bn
	fc_B2	512; lrelu; bn
Joint discrimination	fc_D1	1024; lrelu; bn
	fc_D1	512; lrelu; bn
	fc_D1	1; sigmoid

B. Loss Functions

The proposed approach aims to learn an efficient encoder network that can map input images into Hamming space. To learn compact yet discriminative hash codes, we enforce two important criteria. First, we construct a semantic similarity matrix based on deep features and encourage the learned hash codes to preserve this semantic similarity. Second, we want to reconstruct input images from their corresponding hash codes. However, instead of pixel-wise reconstruction we expect the learned hash codes to be invariant to trivial variations such as color, location and pose. To achieve this purpose, we design two loss functions, namely *semantic-preserving loss* and *adversarial inference loss*.

In the following subsections, we first introduce these two individual losses and then describe the construction of the semantic similarity matrix in detail.

1) Semantic-Preserving Loss: Assume that a semantic similarity matrix $\boldsymbol{S} \in \{1,-1\}^{N \times N}$ is already given, the construction of which will be described in Section III-B.3. Note that $S_{ij} = 1$ means that images I_i and I_j are semantically similar, while $S_{ij} = -1$ indicates that images I_i and I_j are semantically dissimilar. In order to preserve the semantic similarity between different images, we try to map semantically similar data points into hash codes with minimal Hamming distance 0, and semantically dissimilar data points into hash codes with maximal Hamming distance, *i.e.*, the number of hash bits K. However, in practice, directly optimizing the Hamming distances is nontrivial because of the complex mathematical formula. So, indicated in [23], we adopt the inner products of hash codes as a surrogate for Hamming distances. The objective function can be formulated as follows:

$$\begin{aligned} &\min_{\boldsymbol{B}} L_s = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(\frac{1}{K} \boldsymbol{b}_i^T \boldsymbol{b}_j - S_{ij} \right)^2, \\ &s.t.\ \boldsymbol{B} \in \{-1,+1\}^{m \times k} \\ &\quad \boldsymbol{b}_i = h(\boldsymbol{x}_i), \quad \forall i \in \{1,2,\ldots,N\}. \end{aligned} \tag{2}$$

where $\boldsymbol{b}_i$ is the binary code for image $\boldsymbol{I}_i$ and $h(\cdot)$ is the hash function.

To incorporate this objective function with the neural networks, we can set $h(\boldsymbol{x}_i) = \text{sign}(E(\boldsymbol{x}; \Theta))$, where $E(\boldsymbol{x}; \Theta) \in \mathbb{R}^k$ denotes the output of the encoder network for $\boldsymbol{x}$ and Θ denotes the parameters for the network. However, it should

be note that binary values make standard back-propagation of the encoder network infeasible, which is known as the *ill-posed gradient* problem. In this paper, for the convenience of optimization, we relax the binary code item in the range of $(-1, 1)$ and use $\tanh(\cdot)$ to approximate the $\text{sign}(\cdot)$ function. Accordingly, the objective function can be rewritten as:

$$\min_{\Theta} \; J(\Theta) = \sum_{i=1}^{n} \left(\frac{1}{k} h(\boldsymbol{x}_i)^T h(\boldsymbol{x}_i) - S_{ij} \right)^2, \\ s.t. \; h(\boldsymbol{x}_i) = \tanh(E(\boldsymbol{x}_i; \Theta)), \quad \forall i \in \{1, 2, \ldots, n\}. \quad (3)$$

2) Adversarial Inference Loss: As noted above, we want to recover the input images based on the output of the encoder network. However, instead of using pixel-wise reconstruction, we only want to believably reconstruct the inputs and make the learned hash codes robust to trivial variations. To achieve this goal, we first design an encoder network that tries to project the original images into binary codes, a generative network that aims to believably reconstruct the inputs, and a discriminative network to guide the learning process of the encoder and generative networks. We then jointly train these three networks using the following adversarial inference loss:

$$L_a = \frac{1}{N} \sum_{i=1}^{N} \Big[\log \Big(D\big(\boldsymbol{I_i}, E(\boldsymbol{I_i})\big) \Big) \Big] \\ + \frac{1}{N} \sum_{i=1}^{N} \Big[\log \Big(1 - D\big(G(\boldsymbol{b}'_i), \boldsymbol{b}\prime_i\big) \Big) \Big], \quad (4)$$

where $\boldsymbol{b}'_i$ is a random approximate binary code, $E(\cdot)$ means the output of the encoder network, and $G(\cdot)$ is the output of the generative network.

By adversarially training these three networks using the adversarial inference loss, we can finally match the distributions for the encoder pair $(\boldsymbol{I_i}, E(\boldsymbol{I_i}))$ and the generator pair $(G(\boldsymbol{b}'_i), \boldsymbol{b}'_i)$ together. The generation process of the encoder network and the generative network is reversed. Therefore, we can learn hash codes and credibly reconstruct input images from them by using these encoder and generative networks.

3) Semantic Similarity Matrix: Recent studies [64], [65] have shown that features extracted from pre-trained deep architectures contain rich semantic information. To better utilize such high-level information, we extract deep features from a pre-trained deep architecture. Based on these features, we then construct a semantic similarity matrix to explicitly capture the semantic relationships across different images.

More specifically, we assume that images with distances obviously smaller than the others are semantically similar and data points with distances obviously larger than the others are semantically dissimilar. Accordingly, we first extract deep features and then conduct k-nearest neighbors (K-NN) searching based on the cosine distance of different image pairs. For each image, we set the nearest K_1 images as its neighbors and obtain the initial similarity matrix S_f:

$$(S_f)_{ij} = \begin{cases} 1, & \text{if } x_j \text{ is } K_1\text{-NN of } x_i, \\ -1, & \text{otherwise.} \end{cases} \quad (5)$$

However, since the dataset used to pre-train the deep model is usually different from our target dataset and the distributions for these two datasets are also usually different, the initial similarity matrix S_f may be inaccurate. Therefore, to further refine this matrix, we add more rigorous constraints when setting similar neighbors. Note that if two images are semantically similar, they usually share more common neighbors than dissimilar images. Accordingly, we try to explicitly extract this semantic relationship. Specifically, for each image pair, we count the number of their common neighbors, which are defined in Eq. 5, and consider pairs with an obviously higher number of common neighbors to be semantically similar and pairs with obviously fewer common neighbors to be dissimilar. For each image, based on the number of common neighbors, we sort the remaining images in descending order and set the images that lie in the top K_2 as its neighbors. Thus, we can obtain another similarity matrix S_n by:

$$(S_n)_{ij} = \begin{cases} 1, & \text{if } x_j \text{ is } K_2\text{-NN of } x_i, \\ -1, & \text{otherwise.} \end{cases} \quad (6)$$

Now, we have obtained two kinds of similarity matrix S_f, which are obtained based on the similarity of deep features, and S_n, which are obtained based on the number of common neighbors. Here, to take full advantage of these two kinds of similarity matrix, we assume that semantically similar points should not only have similar deep features but also share more common neighbors. Accordingly, we can generate the final semantic similarity matrix by combining S_f and S_n:

$$(S)_{ij} = \begin{cases} 1, & \text{if } (S_f)_{ij} = 1 \text{ and } (S_n)_{ij} = 1, \\ -1, & \text{otherwise.} \end{cases} \quad (7)$$

4) Out of Sample Extension: For any query image, we can obtain its hash code by directly forward-propagating through the encoder network, as follows:

$$\boldsymbol{b}_i = \text{sign}(E(\boldsymbol{I}_i)). \quad (8)$$

IV. Optimization

The overall objective of UADH is to integrate the semantic-preserving loss in Eq. (2) with the adversarial inference loss in Eq. (4). Since our approach adopts a GAN-like adversarial framework, we train the whole architecture using a minimax game mechanism. The whole learning algorithm is summarized in Algorithm 1. In particular, the encoder and the generative network are jointly optimized according to the following loss:

$$L_1 = L_s + \alpha L_a, \quad (9)$$

where α is a parameter to balance these two parts.

We use the mini-batch SGD method to learn the neural network parameters. Specifically, we first sample a mini-batch of the training data points and then update the parameters based on this mini-batch data. For the sake of simplicity, we define $\boldsymbol{z}_i = E(\boldsymbol{x}_i; \Theta)$, and $\boldsymbol{v}_i = h(\boldsymbol{x}_i)$. We then calculate

the gradient of L_s with regard to z_i as:

$$\frac{\partial L_s}{\partial z_i} = \frac{2}{kN^2}\sum_{i=1}^{N}(1-\boldsymbol{v}_i^2)\odot(\sum_{j=1}^{m}\left|S_{ij}\right|(\frac{1}{K}\boldsymbol{v}_i\top\boldsymbol{v}_j - S_{ij})\boldsymbol{v}_j + (\frac{1}{K}\boldsymbol{v}_i\top\boldsymbol{v}_i - 1)\boldsymbol{v}_i). \quad (10)$$

The gradients for loss function L_s *w.r.t.* other parameters can be readily computed using the chain rule. To calculate the gradients of L_a *w.r.t* its outputs, we first define $\boldsymbol{d}_i = D(\boldsymbol{I}_i, E(\boldsymbol{I}_i))$ and $\boldsymbol{g}_i = D(G(\boldsymbol{b}_i'), \boldsymbol{b}_i')$, then obtain the gradients by:

$$\frac{\partial L_a}{\partial \boldsymbol{d}_i} = \frac{\alpha}{N\boldsymbol{d}_i}, \quad \frac{\partial L_a}{\partial \boldsymbol{g}_i} = \frac{\alpha}{N(\boldsymbol{g}_i - 1)}. \quad (11)$$

Again, the gradients for loss function L_a *w.r.t* other parameters can be easily computed using the chain rule. Combining Eq. (10) and Eq. (11), we can easily obtain the gradients of L_1 *w.r.t* all parameters of the encoder and generative networks, then update these parameters using SGD.

The discriminative network is optimized with the following loss:

$$L_2 = -L_a. \quad (12)$$

We can easily obtain the gradients of L_2 *w.r.t.* $\boldsymbol{d}_i$ and $\boldsymbol{g}_i$ by:

$$\frac{\partial L_a}{\partial \boldsymbol{d}_i} = -\frac{1}{N\boldsymbol{d}_i}, \quad \frac{\partial L_a}{\partial \boldsymbol{g}_i} = -\frac{1}{N(\boldsymbol{g}_i - 1)}. \quad (13)$$

The gradients for loss function L_2 *w.r.t.* other parameters of the discriminative network can also be readily computed using the chain rule. We then update these parameters using SGD.

By minimizing Eq. (9), the encoder network can successfully map input images to a Hamming space where the semantic similarity can be preserved. Moreover, by minimizing Eq. (13), the discriminative network is trained to distinguish whether the pairs of hash codes and images are from the encoder network or from the generative network. By alternately conducting these two training processes, we can ultimately learn a mutually coherent encoder and generative network.

Algorithm 1 UADH

Training Stage

Requirements: Training images $\mathbf{X}$, code length K, and mini-batch size N.
Procedure:
1. Initialize parameters for the encoder and generative networks and discriminative network.
repeat
3.1 Randomly sample N points from $\mathcal{I}$ to construct a mini-batch.
3.2 Sample N random vectors from $\mathcal{N}(0, \boldsymbol{I})$, and get the approximate binary codes using the sigmoid function.
3.3 Update parameters of the encoder and generative network according to (9).
3.4 Update parameters of the discriminative network according to (13).
until *convergence*;

Testing Stage

Input: Image query $\mathbf{q}_i$, parameters for the encoder network.
Output: Hash codes for the query.
Procedure:
1. Calculate the output of the encoder network by directly forward-propagating the input images through the encoder network.
2. Calculate the hash codes by using Eq. (8).

V. Experimental Results

In this section, we provide extensive evaluations of the proposed method and demonstrate its performance based on three widely used benchmark datasets, **CIFAR10**, **FLICKR25K**, and **NUSWIDE**. In the following part, we first introduce the datasets, then present and discuss the experimental results.

A. Datasets

CIFAR10 contains 10 object categories, each consisting of 6000 images, resulting in a total of 60,000 images. The dataset is divided into a retrieval set and a test set containing 50,000 and 10,000 images respectively. We randomly select 5,000 images from the retrieval set as a training set.

FLICKR25K is a image dataset collected from the Flickr website, which contains 25,000 images. Each image is associated with one or more labels from the 24 provided labels. 2,000 images are randomly selected as a test set, While, the rest images are used as a retrieval set, from which we randomly select 5,000 images as a training set.

NUSWIDE contains 269,648 images, each of which are annotated with multiple labels referring to 81 concepts. The subset containing the 10 most popular concepts is used here. We randomly select 5,000 images as a test set; the remaining images are used as a retrieval set, and 10,500 images are randomly selected from the retrieval set for use as a training set.

For our method and other deep hashing methods, raw images are accepted as the inputs. For other shallow architecture-based methods, we extract 4096-dimensional deep features from the fc7-layer of the VGG-F network [62]. Note that the construction of the semantic similarity matrix is an offline process, we can thus utilize a more powerful network to learn this matrix. Accordingly, in this paper, we employ the resnet152 architecture [66] to learn the semantic similarity matrix. To further demonstrate the superiority of the proposed method, we also extract 2048-dimensional feature vectors from

the pool5-layer using the resnet152 architecture for traditional architecture-based methods.

B. Implementation Details

We implement the UADH model via TensorFlow [67]. For the encoder network, we adopt the VGG-F architecture [65]. Parameters for all layers except for the hash layer are fine-tuned from the pre-trained model. All other parameters in our model are learned from scratch. We employ the SGD method to optimize our model, with 0.9 momentum, min-batch size 24, and learning rate 10^{-3}. Parameters α, K_1, and K_2 are selected using a cross-validation set. In our experiments we set α as 10^{-4}. K_1 and K_2 are set as 500 and 300 for CIFAR10, and 2,000 and 1,000 for FLICKR25K and NUSWIDE.

We adopt four evaluation criteria to evaluate the performance, which are mean average precision (MAP), topN-precision, recall@K, and precision-recall. The first three are based on Hamming ranking results, which returns retrieved data according to their Hamming distance with the query data. While, the last criterion is based on hash lookup, which constructs a lookup table and returns points falling within a given Hamming radius centered at the query.

MAP is one popular criteria to measure the retrieval performance. For a query data, a list of R-ranked points are first returned. Then the average precision (AP) is given as

$$AP = \frac{1}{N}\sum_{r=1}^{R} P(r)\delta(r), \tag{14}$$

where N is the relevant data point number for the query, and $P(r)$ is the precision value for the first r returned data points. Note that $\delta(r) = 1$ if the r-th returned data points is relevant to the given query; otherwise, $\delta(r) = 0$. MAP denotes the average APs for queries. R represents the number of retrieval points. Two points are regarded as relevant if they share at least one common label. TopN-precision gives the precision for different returned data point numbers. Recall@K counts the percentage of true neighbors among all the ground truth. Precision-recall shows recall and the corresponding precision values, and is usually obtained by varying the Hamming radius and evaluating the precision and recall values.

C. Baseline Methods

The proposed method is compared with five state-of-the-art unsupervised shallow architecture-based hashing methods (ITQ, SH, DSH, SpH, and SGH) and two recently proposed deep unsupervised hashing methods (DeepBit and BGAN).

- ITQ [53] first performs PCA to map original data to a low-dimensional space then learns an orthogonal rotation matrix to minimize the quantization error.
- SH [22] treats the problem of finding the best code for a given dataset as a graph partitioning problem, and obtains a spectral method by relaxing this partitioning problem.
- DSH [68] extends LSH [69] by exploring the geometric structure of the data and avoids randomly selecting projections by using projections that best agree with the data distribution.

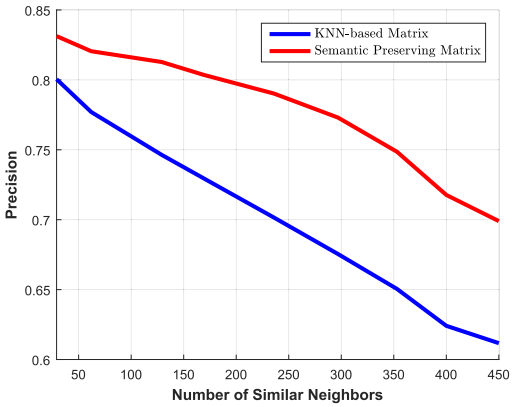


Fig. 2. Nearest neighbor precision for the KNN-based matrix and our proposed semantic similarity matrix.

- SpH [55] uses a hypersphere to formulate a spherical hash function, and also proposes an iterative optimization algorithm to learn the hash codes.
- SGH [57] adopts a variational autoencoder framework and learns hash functions using the minimum description length principle, such that the learned hash codes can efficiently compress the dataset.
- DeepBit [47] constructs a similarity matrix by rotating the original images and considering the rotated image and the corresponding original image to be similar. Hash codes are learned to preserve this similarity.
- BGAH [61] restricts the input noise variable of GAN to be binary, and tries to generate images that preserve neighbor structure and content information.

D. Effect of Semantic Similarity Matrix

We evaluate the precision of nearest neighbors for the matrix based on KNN and our proposed semantic similarity matrix for the CIFAR10 dataset. Results are shown in Fig. 2, which clearly demonstrates the superiority of our proposed semantic similarity matrix.

E. Results

We first give the MAP results for our method and all baseline methods across different hash bit lengths on all three datasets as a global evaluation. We then fix the code length to 32 and draw precision-recall curves, topN-precision curves, and recall@K curves of the proposed UADH and all baseline methods to present a comprehensive comparison.

Table II reports the MAP values for all methods with code length varying from 16 to 128 and all shallow methods evaluated based on VGG-F features. From Table II, we can see that the proposed method consistently obtains the best results across different hash bit lengths and different datasets. Specifically, for CIFAR10, the MAP of UADH obtains a relative improvement over the best baseline methods of 17.98%, 21.87%, 18.90%, and 18.30% for 16, 32, 64, and 128 bit length respectively. To further verify the effectiveness of our UADH, we extract CNN features from the resnet152 architecture, which is pre-trained on the ImageNet dataset. All shallow

TABLE II

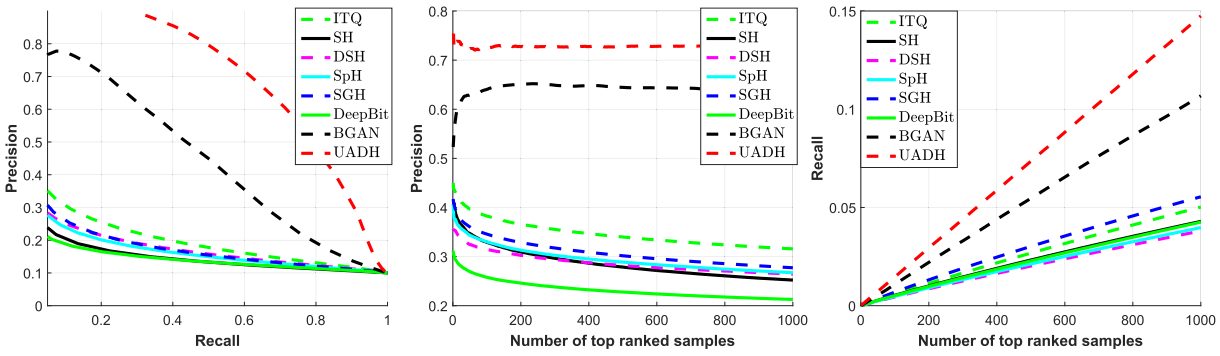
COMPARISON WITH BASELINES BASED ON VGG-F FEATURES IN TERMS OF MAP. THE BEST ACCURACY IS SHOWN IN BOLDFACE

method	CIFAR10				FLICKR25K				NUSWIDE			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
SH	0.1621	0.1575	0.1549	0.1548	0.6091	0.6105	0.6033	0.6014	0.4350	0.4129	0.4062	0.4100
SpH	0.1664	0.1745	0.1931	0.1992	0.6119	0.6315	0.6381	0.6451	0.4458	0.4537	0.4926	0.5000
ITQ	0.1999	0.2098	0.2229	0.2312	0.6492	0.6518	0.6546	0.6577	0.5270	0.5241	0.5334	0.5398
DSH	0.1686	0.1846	0.1955	0.2071	0.6452	0.6547	0.6551	0.6557	0.5123	0.5118	0.5110	0.5267
SGH	0.1795	0.1827	0.1889	0.1904	0.6362	0.6283	0.6253	0.6206	0.4994	0.4869	0.4851	0.4945
DeepBit	0.2204	0.2410	0.2900	0.4714	0.5934	0.5933	0.6199	0.6349	0.3844	0.4341	0.4461	0.4917
BGAN	0.4971	0.4702	0.5074	0.5236	0.6165	0.6149	0.6173	0.6123	0.3841	0.4064	0.4393	0.4750
Ours	**0.6769**	**0.6889**	**0.6964**	**0.7066**	**0.7555**	**0.7635**	**0.7698**	**0.7669**	**0.6848**	**0.6992**	**0.7054**	**0.7101**

TABLE III

COMPARISON WITH BASELINES BASED ON RESNET152 FEATURES IN TERMS OF MAP. THE BEST ACCURACY IS SHOWN IN BOLDFACE

method	CIFAR10				FLICKR25K				NUSWIDE			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
SH	0.4016	0.3337	0.2910	0.2695	0.6285	0.6147	0.6105	0.5998	0.4672	0.4384	0.4220	0.4128
SpH	0.3715	0.4417	0.4770	0.5210	0.6371	0.6578	0.6783	0.6921	0.4864	0.5050	0.5273	0.5531
ITQ	0.6368	0.6338	0.6266	0.6315	0.7107	0.7126	0.7112	0.7105	0.5867	0.5865	0.5865	0.5826
DSH	0.3354	0.4191	0.4255	0.4784	0.6663	0.6963	0.7121	0.7241	0.4475	0.5587	0.5666	0.5708
SGH	0.5454	0.5164	0.4675	0.4196	0.6830	0.6679	0.6645	0.6528	0.5395	0.5079	0.4887	0.4747
DeepBit	0.2204	0.2410	0.2900	0.4714	0.5934	0.5933	0.6199	0.6349	0.3844	0.4341	0.4461	0.4917
BGAN	0.4971	0.4702	0.5074	0.5236	0.6165	0.6149	0.6173	0.6123	0.3841	0.4064	0.4393	0.4750
Ours	**0.6769**	**0.6889**	**0.6964**	**0.7066**	**0.7555**	**0.7635**	**0.7698**	**0.7669**	**0.6848**	**0.6992**	**0.7054**	**0.7101**



(a) (b) (c)

Fig. 3. Curces for (a) Precision-recall, (b) topN-precision, and (c) Recall@K on CIFAR10 with code length 32.

baselines are then evaluated based on these deep features. Table III reports MAP values with shallow methods evaluated on resnet152 features. Comparing Table II and Table III, it can be seen that most shallow methods achieve better results by using resnet152 features. This may be because more powerful architecture captures more high-level semantic information in the corresponding features. From Table III, we can also see that the proposed UADH outperforms all baseline methods. It should be noted that our method is based on the VGG-F network, which is less powerful than the resnet152 network. As such, the results presented in Tables II and III clearly demonstrate the superiority of the proposed UADH.

Fig. 3, 4, and 5 show recall-precision curves, topN-precision curves, and recall@K curves on CIFAR10, FLICKR25K, and NUSWIDE, respectively. All the shallow methods are based on VGG-F features. Fig. 6, 7, and 8 show the results based on the same evaluation criteria and datasets, except that the features for the shallow methods are extracted from the resnet152 architecture. The left-hand part of these figures shows the recall-precision results, from which we can see that UADH always achieves the highest performance compared with both other deep hashing methods and all shallow architecture-based methods with both VGG-F and resnet152 features. The right-hand part of these figures shows the recall@K results, which also indicates that our method can outperform other methods in all cases. Since MAP values, topN-precision curves, and recall@K curves are all based on Hamming ranking, our above analysis collectively shows that UADH exhibits superior performance for Hamming ranking-based evaluations. To illustrate hash lookup results,

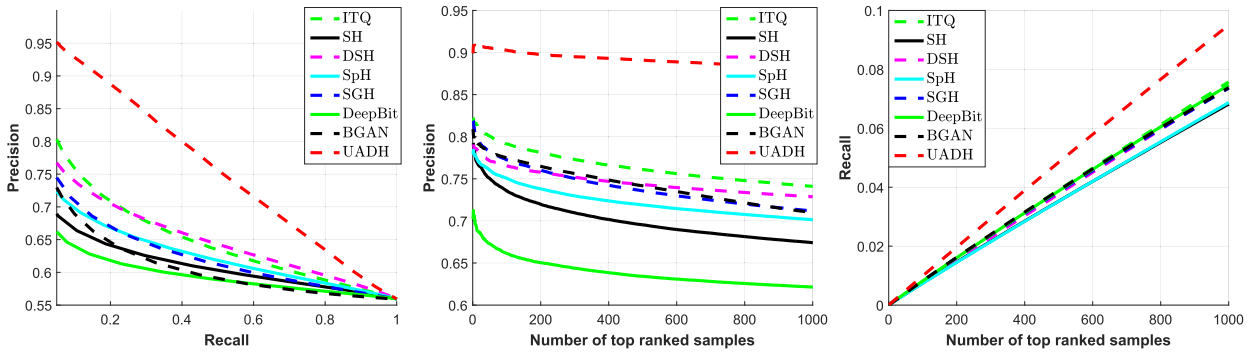


Fig. 4. Cureves for (a) Precision-recall, (b) topN-precision, and (c) Recall@K on FLICKR25K with code length 32.

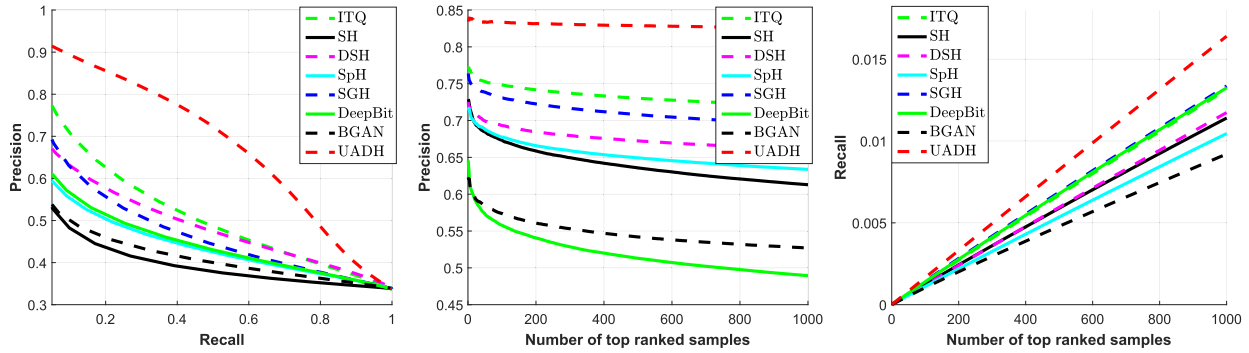


Fig. 5. Curves for (a) Precision-recall, (b) topN-precision, and (c) Recall@K on NUSWIDE with code length 32.

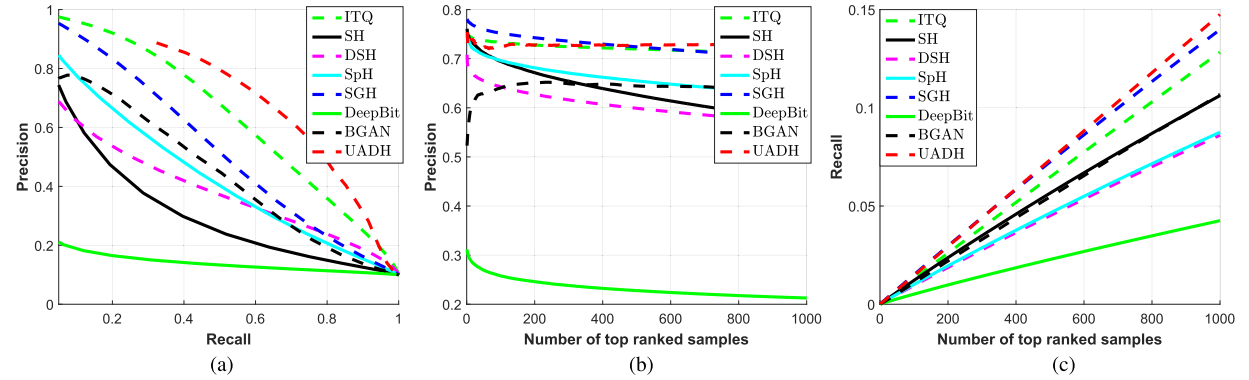


Fig. 6. Curves for (a) Precision-recall, (b) topN-precision, and (c) Recall@K on CIFAR10 with code length 32.

we plot the recall-precision curves for all deep learning-based methods and shallow architecture-based methods with VGG-F and resnet152 features by varying the Hamming radius. The results are shown in the middle part of the above figures, from which we can see that UADH always achieves the best performance in all cases.

F. Ablation Study

We verify the impact of the adversarial learning modules on our UADH's performance by designing a variant, UADH-1, which is built by removing the adversarial learning component. Table IV presents the MAP results on different bit lengths for three datasets. From the results, we can see that UADH always achieves better results, demonstrating the role of the adversarial learning.

The adversarial learning part enforces hash codes to preserve important information by generating nearly real images from hash codes and can inherently reduce the over-fitting problem. Fig. 9 illustrates synthetic image samples on CIFAR10 dataset. To measure the over-fitting problem, we adopt the MAP gap between training and testing data. Concretely, we calculate the MAP@50 results for training and testing data and show their gap for UADH-1 and

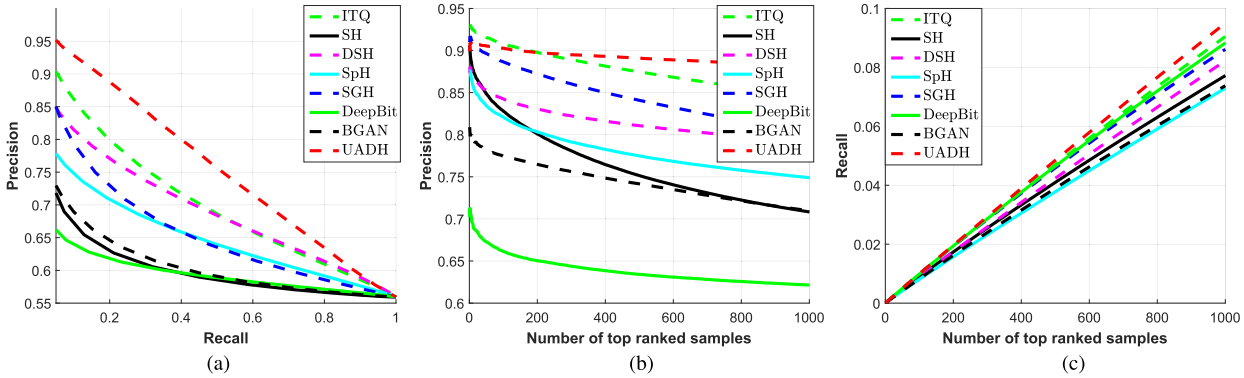


Fig. 7. Curves for (a) Precision-recall curves, (b) topN-precision, and (c) Recall@K on FLICKR25K with code length 32.

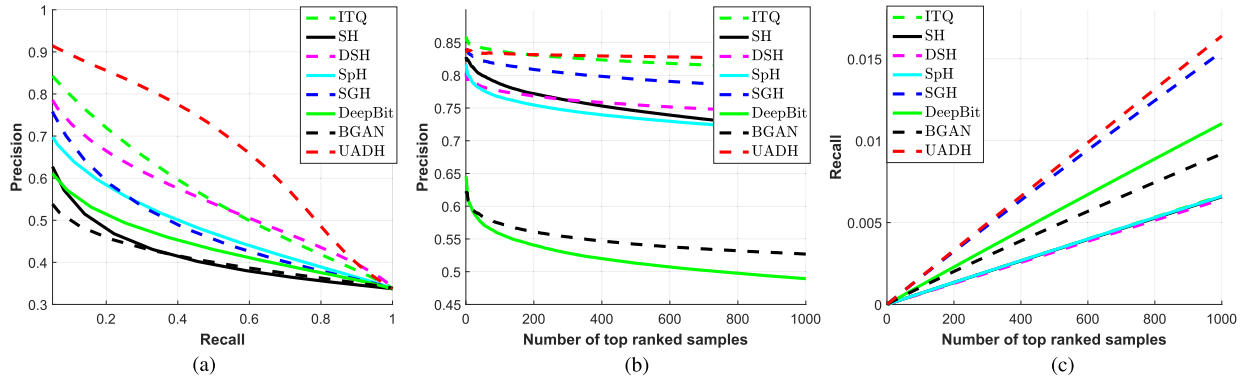


Fig. 8. Curves for (a) Precision-recall curves, (b) topN-precision, and (c) Recall@K on NUSWIDE with code length 32.

TABLE IV
MAP RESULTS OF UADH AND ITS VARIANT UADH-1

Dataset	Methods	16 bits	32 bits	64 bits	128 bits
CIFAR10	UADH-1	0.6619	0.6820	0.6890	0.6850
	UADH	**0.6769**	**0.6889**	**0.6964**	**0.7066**
FLICKR25K	UADH-1	0.7515	0.7613	0.7628	0.7648
	UADH	**0.7555**	**0.7635**	**0.7698**	**0.7669**
NUSWIDE	UADH-1	0.6784	0.6853	0.6932	0.7062
	UADH	**0.6848**	**0.6992**	**0.7054**	**0.7101**

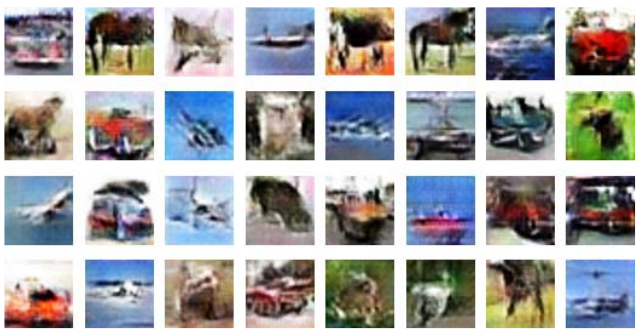


Fig. 9. Generated samples for CIFAR10 dataset.

UADH respectively. The results are illustrated in Fig. 10(a), from which we can get that UADH consistently have smaller MAP gaps than UADH-1. These results verifies that the adversarial learning can effectively ease the over-fitting problem.

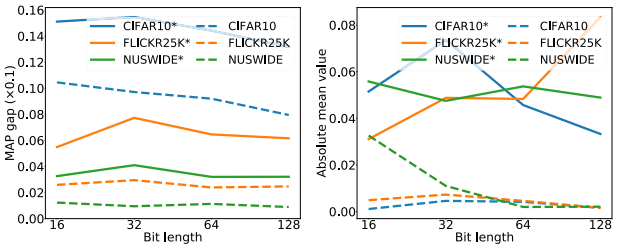


Fig. 10. (a) Comparison for learning balanced codes, where CIFAR10* represents the absolute mean values of the database hash codes for UADH-1, and CIFAR10 represents absolute mean values of the database hash codes for UADH. The symbols for other datasets are similar. (b) Comparison for reducing overfitting, where CIFAR10* represents the gap between training and testing data for UADH-1, and CIFAR10 represents the gap for UADH. The symbols for other datasets are similar.

To maximize the information capacity, binary codes are usually expected to be balance, which means that each hash bit should have equal probability to be $+1$ or -1. Note that the adversarial learning part can match the joint distribution of hash codes and images. Therefore, the marginal hash code distribution can also be matched. Since the hash codes for the generator are sampled randomly, the hash codes from the encoder will also be forced to have equal probability to be $+1$ or -1. Here, we show that the adversarial learning part can help to learn balanced codes. To give a quantitative analysis, we calculate the absolute mean values of the database

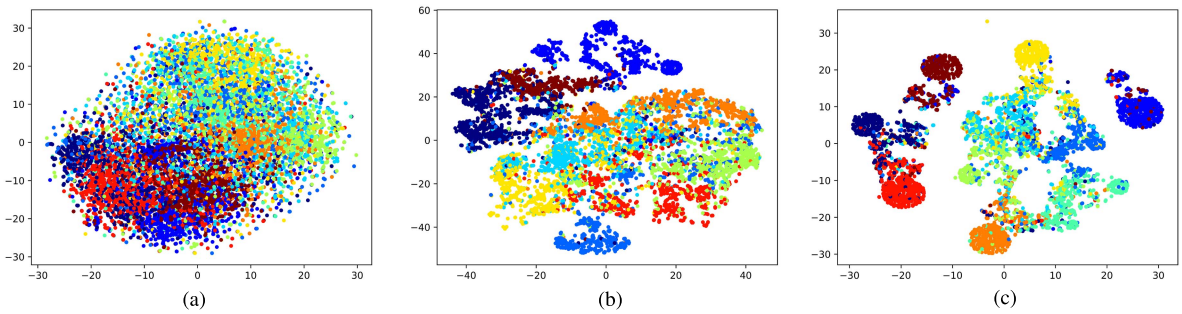


Fig. 11. t-SNE visualization of the learned hash codes for (a) Deepbit [47], (b) BGAN [61], and (c) UADH, with code length 64 on CIFAR10.

TABLE V
COMPARISON WITH SUPERVISED HASHING METHODS

Method	CIFAR10		NUSWIDE	
	16 bits	32 bits	16 bits	32 bits
DRSCH	0.615	0.629	0.618	0.623
DSCH	0.609	0.613	0.592	0.611
DSRH	0.608	0.611	0.609	0.621
DPSH	**0.763**	**0.795**	0.715	0.736
UADH	0.677	0.689	**0.756**	**0.764**

hash codes for UADH-1 and UADH. The results are shown in Fig. 10(b), which clearly show that the hash codes from UADH have smaller absolute mean values than UADH-1 and demonstrate that UADH can learn more balanced hash codes than UADH-1.

G. Comparison With Supervised Hashing Methods

We further compare UADH with several popular supervised hashing methods, including DRSCH [70], DSCH [70], DSRH [71], and DPSH [41]. The MAP values for all methods on the CIFAR10 and NUSWIDE datasets with 16 and 32 hash bit length are reported in Table V. It should be noted that we directly use the results reported in DPSH [41], which use an experimental setting that differs slightly from ours. We use these results to give a general comparison. From Table V, we can see that UADH achieves the best performance on NUSWIDE and the second best on CIFAR10 for both 16 and 32 hash bit length, even though our method does not leverages any label information. These results further demonstrate the superiority of our proposed method.

H. Visualization

Fig. 11 shows the t-SNE visualizations [72] of the learned hash codes for the proposed method and two deep baseline methods, Deepbit [47] and BGAN [61], on the CIFAR10 dataset with 64 hash bit length. As can be seen from Fig. 11, Deepbit and BGAN do not preserve semantic structures of testing data in Hamming space very well as they fail to leverage the semantic similarity of testing data. Thanks to the proposed semantic-preserving loss and adversarial inference loss, along with the adversarial training process, our methods learn hash codes that can capture rich semantic information.

VI. CONCLUSION

In this paper, we propose a novel unsupervised hashing method called Unsupervised ADversarial Hashing (UADH) for image search. UADH consists of an encoder network, a generative network, and a discriminative network, and adopts an adversarial learning framework to match the encoder and generative networks. To further preserve the semantic information of training data in Hamming space, we also integrate a semantic-preserving loss in the adversarial learning framework. Experiments conducted on CIFAR10, FLICKR25K, and NUSWIDE show that the proposed UADH yields state-of-the-art image retrieval performance on standard benchmarks, thereby validating our proposal and analysis.

REFERENCES

[1] M. Norouzi, D. J. Fleet, and R. R. Salakhutdinov, "Hamming distance metric learning," in *Proc. NIPS*, 2012, pp. 1061–1069.
[2] X. Yang, C. Deng, X. Liu, and F. Nie, "New $l_{2,1}$-norm relaxation of multi-way graph cut for clustering," in *Proc. AAAI*, Apr. 2018, pp. 1–8.
[3] J. Masci, M. M. Bronstein, A. M. Bronstein, and J. Schmidhuber, "Multimodal similarity-preserving hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 824–830, Apr. 2014.
[4] C. Deng, R. Ji, W. Liu, D. Tao, and X. Gao, "Visual reranking through weakly supervised multi-graph learning," in *Proc. ICCV*, Dec. 2013, pp. 2600–2607.
[5] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li, "Query-adaptive reciprocal hash tables for nearest neighbor search," *IEEE Trans. Image Process.*, vol. 25, no. 2, pp. 907–919, Feb. 2016.
[6] E. Yang, T. Liu, C. Deng, and D. Tao, "Adversarial examples for Hamming space search," *IEEE Trans. Cybern.*, to be published.
[7] S. You, C. Xu, Y. Wang, C. Xu, and D. Tao, "Privileged multi-label learning," in *Proc. IJCAI*, Aug. 2017, pp. 3336–3342.
[8] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. FOCS*, 2006, pp. 459–468.
[9] P. Indyk and R. Motwani, "Approximate nearest neighbors: Towards removing the curse of dimensionality," in *Proc. STOC*, 1998, pp. 604–613.
[10] J. Zhou, G. Ding, and Y. Guo, "Latent semantic sparse hashing for cross-modal similarity search," in *Proc. SIGIR*, Jul. 2014, pp. 415–424.
[11] E. Yang, C. Deng, C. Li, W. Liu, J. Li, and D. Tao, "Shared predictive cross-modal deep quantization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5292–5303, Nov. 2018.
[12] C. Deng, X. Tang, J. Yan, W. Liu, and X. Gao, "Discriminative dictionary learning with common label alignment for cross-modal retrieval," *IEEE Trans. Multimedia*, vol. 18, no. 2, pp. 208–218, Feb. 2016.
[13] Y. Cao, M. Long, J. Wang, Q. Yang, and P. S. Yu, "Deep visual-semantic hashing for cross-modal retrieval," in *Proc. SIGKDD*, Aug. 2016, pp. 1445–1454.
[14] E. Yang, C. Deng, T. Liu, W. Liu, and D. Tao, "Semantic structure-based unsupervised deep hashing," in *Proc. IJCAI*, Jul. 2018, pp. 1064–1070.

[15] W. Liu, J. Wang, Y. Mu, S. Kumar, and S.-F. Chang, "Compact hyperplane hashing with bilinear functions," in *Proc. ICML*. Jun. 2012, pp. 467–474.

[16] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 6, pp. 1092–1104, Jun. 2012.

[17] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013.

[18] J. L. Bentley, "Multidimensional binary search trees used for associative searching," in *Proc. Commun. ACM*, vol. 18, no. 9, pp. 509–517, Sep. 1975.

[19] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, vol. 14, 1984, pp. 47–57.

[20] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2018.

[21] X. Liu, J. He, C. Deng, and B. Lang, "Collaborative hashing," in *Proc. CVPR*, Jun. 2014, pp. 2147–2154.

[22] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. NIPS*, 2009, pp. 1753–1760.

[23] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. CVPR*, Jun. 2012, pp. 2074–2081.

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

[25] J. Lu, V. E. Liong, and J. Zhou, "Deep hashing for scalable image search," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2352–2367, May 2017.

[26] D. Tian and D. Tao, "Global hashing system for fast image search," *IEEE Trans. Image Process.*, vol. 26, no. 1, pp. 79–89, Jan. 2017.

[27] L. Liu, M. Yu, and L. Shao, "Multiview alignment hashing for efficient image search," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 956–966, Mar. 2015.

[28] C. Li, C. Deng, N. Li, W. Liu, X. Gao, and D. Tao, "Self-supervised adversarial hashing networks for cross-modal retrieval," in *Proc. CVPR*, 2018, pp. 4242–4251.

[29] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[30] C. D. Toth, J. O'Rourke, and J. E. Goodman, *Handbook of Discrete and Computational Geometry*. Boca Raton, FL, USA: CRC Press, 2004.

[31] M. M. Bronstein, A. M. Bronstein, F. Michel, and N. Paragios, "Data fusion through cross-modality metric learning using similarity-sensitive hashing," in *Proc. CVPR*, Jun. 2010, pp. 3594–3601.

[32] S. Kumar and R. Udupa, "Learning hash functions for cross-view similarity search," in *Proc. IJCAI*, Jun. 2011, pp. 1360–1365.

[33] S. Ferdowsi, S. Voloshynovskiy, D. Kostadinov, and T. Holotyak, "Sparse ternary codes for similarity search have higher coding gain than dense binary codes," in *Proc. ISIT*, Jun. 2017, pp. 2653–2657.

[34] D. Valsesia and E. Magli, "Binary adaptive embeddings from order statistics of random projections," *IEEE Signal Process. Lett.*, vol. 24, no. 1, pp. 111–115, Jan. 2017.

[35] W. Liu and T. Zhang, "Multimedia hashing and networking," *IEEE Multimedia*, vol. 23, no. 3, pp. 75–79, Jul./Sep. 2016.

[36] J. Wang, W. Liu, S. Kumar, and S.-F. Chang, "Learning to hash for indexing big data—A survey," *Proc. IEEE*, vol. 104, no. 1, pp. 34–57, Jan. 2016.

[37] H. Wang, Y. Yang, E. Yang, and C. Deng, "Exploring hybrid spatio-temporal convolutional networks for human action recognition," *Multimedia Tools Appl.*, vol. 76, no. 13, p. 15065–15081, 2017.

[38] Y. Du, M.-H. Hsieh, T. Liu, and D. Tao. (2018). "The expressive power of parameterized quantum circuits." [Online]. Avaliable: https://arxiv.org/abs/1810.11922

[39] X. Chen, C. Xu, X. Yang, and D. Tao, "Attention-GAN for object transfiguration in wild images," in *Proc. ECCV*, 2018, pp. 164–180.

[40] S. You, C. Xu, C. Xu, and D. Tao, "Learning from multiple teacher networks," in *Proc. SIGKDD*, 2017, pp. 1285–1294.

[41] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. IJCAI*, 2016, pp. 1711–1717.

[42] E. Yang, C. Deng, W. Liu, X. Liu, D. Tao, and X. Gao, "Pairwise relationship guided deep hashing for cross-modal retrieval," in *Proc. AAAI*, 2017, pp. 1618–1625.

[43] C. Deng, Z. Chen, X. Liu, X. Gao, and D. Tao, "Triplet-based deep hashing network for cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 27, no. 8, pp. 3893–3903, Aug. 2018.

[44] A. Krizhevsky and G. E. Hinton, "Using very deep autoencoders for content-based image retrieval," in *Proc. ESANN*, 2011, pp. 1–7.

[45] R. Salakhutdinov and G. Hinton, "Semantic hashing," *Int. J. Approx. Reasoning*, vol. 50, no. 7, pp. 969–978, 2009.

[46] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. CVPR*, 2015, pp. 2475–2483.

[47] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. CVPR*, Jun. 2016, pp. 1183–1192.

[48] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," in *Proc. ICLR*, 2017, pp. 1–18.

[49] A. Joly and O. Buisson, "A posteriori multi-probe locality sensitive hashing," in *Proc. 16th ACM Int. Conf. Multimedia*, 2008, pp. 209–218.

[50] M. Norouzi and D. M. Blei, "Minimal loss hashing for compact binary codes," in *Proc. ICML*, 2011, pp. 353–360.

[51] A. Z. Broder and A. R. Karlin, "Multilevel adaptive hashing," in *Proc. SODA*, 1990, pp. 43–53.

[52] G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proc. CVPR*, 2014, pp. 1963–1970.

[53] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[54] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. NIPS*, 2014, pp. 3419–3427.

[55] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *Proc. CVPR*, Jun. 2012, pp. 2957–2964.

[56] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. ICML*, 2011, pp. 1–8.

[57] B. Dai, R. Guo, S. Kumar, N. He, and L. Song, "Stochastic generative hashing," in *Proc. ICML*, Aug. 2017, pp. 913–922.

[58] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.

[59] X. Chen, C. Xu, X. Yang, L. Song, and D. Tao, "Gated-GAN: Adversarial gated networks for multi-collection style transfer," *IEEE Trans. Image Process.*, vol. 28, no. 2, pp. 546–560, Feb. 2019.

[60] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Trans. Evol. Comput.*, to be published.

[61] J. Song. (2017). "Binary generative adversarial networks for image retrieval." [Online]. Available: https://arxiv.org/abs/1708.04150

[62] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[63] A. Radford, L. Metz, and S. Chintala. (2015). "Unsupervised representation learning with deep convolutional generative adversarial networks." [Online]. Available: https://arxiv.org/abs/1511.06434

[64] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014, pp. 580–587.

[65] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. NIPS*, 2012, pp. 1097–1105.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[67] M. Abadi *et al.*. (2016). "Tensorflow: Large-scale machine learning on heterogeneous distributed systems." [Online]. Available: https://arxiv.org/abs/1603.04467

[68] Z. Jin, C. Li, Y. Lin, and D. Cai, "Density sensitive hashing," *IEEE Trans. Cybern.*, vol. 44, no. 8, pp. 1362–1371, Aug. 2014.

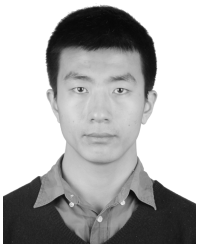
[69] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, vol. 99, no. 6, pp. 518–529, 1999.

[70] 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.

[71] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proc. CVPR*, 2015, pp. 1556–1564.

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

Cheng Deng (M'11) received the B.E., M.S., and Ph.D. degrees in signal and information processing from Xidian University, Xi'an, China. He is currently a Full Professor with the School of Electronic Engineering, Xidian University. He has authored or coauthored more than 70 scientific articles at top venues, including IEEE TNNLS, TIP, TCYB, TMM, TSMC, ICCV, CVPR, ICML, NIPS, IJCAI, and AAAI. His research interests include computer vision, pattern recognition, and information hiding.



Erkun Yang received the B.E. degree in electronic and information engineering from Xidian University, China, in 2013, where he is currently pursuing the Ph.D. degree with the School of Electronic Engineering. His research interests lie primarily in computer vision and machine learning.

Tongliang Liu received the B.E. degree in electronic engineering and information science from the University of Science and Technology of China, Hefei, China, and the Ph.D. degree in information systems from the University of Technology Sydney, NSW, Australia. He is currently a Lecturer with the School of Computer Science, Faculty of Engineering and Information Technologies, and a Core Member of the UBTECH Sydney AI Centre, The University of Sydney, NSW, Australia. He has authored and coauthored more than 50 research papers, including in IEEE T-PAMI, IEEE T-NNLS, IEEE T-IP, ICML, CVPR, and KDD. His current research interests include statistical learning theory, machine learning, and computer vision.

Jie Li received the B.Sc., M.Sc., and Ph.D. degrees in circuit and system from Xidian University, China, in 1995, 1998, and 2005, respectively. Since 1998, she has been with the School of Electronic Engineering, Xidian University, where she is currently a Professor. Her research interests include computational intelligence, machine learning, and image processing. In these areas, she has published more than 30 technical articles in refereed journals and proceedings including IEEE TCSVT and IJFS.

Wei Liu (M'14–SM'19) received the Ph.D. degree in EECS from Columbia University, New York, NY, USA. He was a Research Scientist with the IBM Thomas J. Watson Research Center, Yorktown Heights, NY, USA. He is currently a Distinguished Scientist with the Tencent AI Lab and the Director of the Computer Vision Center. He has long been devoted to research and development in the fields of machine learning, computer vision, information retrieval, and big data. His research received a number of awards and honors, such as the 2011 Facebook Fellowship, the 2013 Jury Award for Best Thesis of Columbia University, the 2016 and 2017 SIGIR Best Paper Award Honorable Mentions, and the 2018 AI's 10 Top Watch Honor. He serves as an Area Chair for several international top-tier AI conferences. He currently serves as an Associate Editor for several international leading AI journals.

Dacheng Tao (F'15) is currently a Professor of computer science and an ARC Laureate Fellow with the School of Computer Science, Faculty of Engineering and Information Technologies, and the Inaugural Director of the UBTECH Sydney Artificial Intelligence Centre, The University of Sydney. He mainly applies statistics and mathematics to artificial intelligence and data science. His research results have expounded in one monograph and more than 200 publications in prestigious journals and at prominent conferences, such as IEEE T-PAMI, T-IP, T-NNLS,T-CYB, IJCV, JMLR, NIPS, ICML, CVPR, ICCV, ECCV, ICDM, and ACM SIGKDD, with several best paper awards, such as the best theory/algorithm paper runner up award in IEEE ICDM'07, the Best Student Paper Award in IEEE ICDM'13, the 2014 ICDM 10-year highest-impact paper award, the 2017 IEEE Signal Processing Society Best Paper Award, and the Distinguished Paper Award in the 2018 IJCAI. He received the 2015 Australian Scopus-Eureka Prize and the 2018 IEEE ICDM Research Contributions Award. He is a Fellow of the Australian Academy of Science, AAAS, IAPR, OSA, and SPIE.