International Journal of Computer Vision
https://doi.org/10.1007/s11263-020-01299-x

Anchor-Based Self-Ensembling for Semi-Supervised Deep Pairwise Hashing

Xiaoshuang Shi[1] · Zhenhua Guo[2] · Fuyong Xing[3] · Yun Liang[1] · Lin Yang[1]

Received: 8 March 2019 / Accepted: 2 February 2020

Abstract
Deep hashing has attracted considerable attention to tackle large-scale retrieval tasks, because of automatic and powerful feature extraction of convolutional neural networks and the gain of hashing in computation and storage costs. Most current supervised deep hashing methods only utilize the semantic information of labeled data without exploiting unlabeled data. However, data annotation is expensive and thus only scarce labeled data are available, which are difficult to represent the true distribution of all data. In this paper, we propose a novel semi-supervised deep pairwise hashing method to leverage both labeled and unlabeled data to learn hash functions. Our method utilizes the transduction of anchors to preserve the pairwise similarity relationship among both labeled and unlabeled samples. Additionally, to explore the semantic similarity information hidden in unlabeled data, it adopts self-ensembling to create strong ensemble targets for latent binary vectors of training samples and form a consensus predicting similarity relationship to multiple anchors. Unlike previous pairwise based hashing methods without maintaining the relevance among similar neighbors, we further explain and exhibit the capability of our method on preserving their relevance through calculating their similarities to anchors. Finally, extensive experiments on benchmark databases demonstrate the superior performance of the proposed method over recent state-of-the-art hashing methods on multiple retrieval tasks. The source codes of the proposed method are available on: https://github.com/xsshi2015/Semi-supervised-Deep-Pairwise-Hashing.

Keywords Anchor · Self-ensembling · Deep pairwise hasing · Semi-seupervised · Large-scale

1 Introduction

Content-based image retrieval (CBIR), which retrieves the most relevant content of one query from a given database, has attracted considerable attention to tackle large-scale tasks in machine learning and computer vision communities over the past two decades (Datta et al. 2008). Most large-scale CBIR applications usually contain hundreds of thousands or millions documents, images or videos, where each data descriptor often contains hundreds or even thousands of dimensions (Torralba et al. 2008; Shi et al. 2017b). As a result, these applications usually suffer from the curse of dimensionality, so that it is infeasible to exhaustively search the nearest neighbors and expensive to store the original data (Wang et al. 2012). To address these issues, hashing has become an effective and efficient technique to achieve significant gains in computation and storage costs (Indyk and

Communicated by Li Liu, Matti Pietikäinen, Jie Qin, Jie Chen, Wanli Ouyang, Luc Van Gool.

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s11263-020-01299-x) contains supplementary material, which is available to authorized users.

✉ Xiaoshuang Shi
xsshi2015@ufl.edu

Zhenhua Guo
zhenhua.guo@sz.tsinghua.edu.cn

Fuyong Xing
fuyong.xing@ucdenver.edu

Yun Liang
yunliang@ufl.edu

Lin Yang
Lin.Yang@bme.ufl.edu

[1] Department of Biomedical Engineering, University of Florida, Gainesville, USA

[2] Tsinghua Shenzhen International Graduate School, Tsinghua University, Shenzhen, People's Republic of China

[3] Department of Biostatistics and Informatics, University of Colorado Denver, Denver, USA

Published online: 18 February 2020

Motwani 1998; Liu et al. 2014). Because it can encode the original data into tens of binary codes by projecting them from the high-dimensional space into a low-dimensional binary space, with maintaining their similarity relationship.

Early efforts focus on data-independent hashing methods (Indyk and Motwani 1998; Broder et al. 2000), which adopt random projections or permutations to construct hash functions. They usually require long bits per hash table to achieve high precision and multiple hash tables to boost recall (Liu et al. 2014). The large storage might restrict their applications. To generate more compact binary codes with attaining higher retrieval precision, data-dependent hashing algorithms (Weiss et al. 2009; Kulis and Darrell 2009; Shen et al. 2015a), using training data to learn hash functions, have attracted increasing attention over the past decade, and they have attained promising performance on many distance measures (Datar et al. 2004; Kulis et al. 2009; Raginsky and Lazebnik 2009). Based on whether using semantic labels, data-dependent hashing methods can be roughly categorized into two major classes: unsupervised and supervised. Unsupervised hashing explores the intrinsic data structure or similarity to maintain the similarity among neighbors without using semantic information (Broder et al. 2000; Weiss et al. 2009; Kulis and Darrell 2009; Song et al. 2018). Supervised hashing exploits semantic labels to project the originally high-dimensional data into a low-dimensional binary space, with maintaining their semantic similarity relationship. Therefore, supervised hashing usually performs better than unsupervised hashing on semantic similarity measures.

Traditional supervised hashing methods (Liu et al. 2012; Shen et al. 2015b; Shi et al. 2016; Kang et al. 2016) learn hash functions to produce binary codes using hand-crafted features, which are usually unable to optimally represent the image content, thereby limiting their retrieval performance. To address this problem, many supervised deep hashing methods (Li et al. 2015; Zhu et al. 2016) utilize convolutional neural networks (CNNs) to simultaneously learn image representations and latent binary vectors, because CNNs can automatically extract powerful features and have achieved great success on many applications (Krizhevsky et al. 2012; He et al. 2016; Xu et al. 2018, 2019). Unfortunately, CNNs usually require a large amount of labeled images to obtain desired performance, and labeling images is laborious, time-consuming and expensive. To alleviate this issue, several semi-supervised deep hashing algorithms using graphs (Zhang and Peng 2017; Yan et al. 2017) or generative adversarial networks (GANs) (Qiu et al. 2017; Wang et al. 2018; Zhang et al. 2018) have been proposed to exploit the semantic information of labeled data as well as explore the similarity information hidden in unlabeled data.

Recently, self-ensembling (Laine and Aila 2016), which firstly creates strong ensemble targets for unlabeled data by using the outputs of a single network under different configurations, such as different training epochs, dropout and input augmentation conditions, and then utilizes these targets as the teacher to guide each unlabeled sample to form a consensus prediction, has been successfully applied to semi-supervised scenarios and outperforms graph or GANs based semi-supervised methods on many classification tasks. Thus, it is promising to apply self-ensembling to semi-supervised deep hashing. However, it is still a challenging task to effectively employ self-ensembling to learn binary codes, with outperforming state-of-the-art hashing methods on multiple retrieval tasks. One major reason is that self-ensembling does not take advantage of the similarity relationship of training data, which plays a significant role in image retrieval. Additionally, most of the existing semi-supervised deep hashing algorithms (Li et al. 2015; Qiu et al. 2017; Wang et al. 2018; Zhang et al. 2018), which belong to multi-wise based hashing that takes advantage of a triplet ranking loss to maximize the agreement of similarity orders over three items, usually take high training time costs or might be difficult to fully make use of the semantic information (Sapkota et al. 2018; Jiang and Li 2018). This is because multi-wise based hashing methods will generate $O(n^3)$ triplets for n labeled training samples. To reduce the cost on computation and computer memory, most of them sample part of triplets, thereby potentially losing some semantic information and decreasing the model retrieval performance. By contrast, pairwise based hashing only produces $O(n^2)$ pairs to fully exploit the semantic information. Moreover, anchor-based pairs (Shi et al. 2017a; Jiang and Li 2018), which select n^a ($n^a << n$) samples from n labeled training data as anchors to represent their distribution and then calculate the pair similarity between labeled training data and anchors, can be utilized to further reduce the complexity of pairwise based hashing, e.g. $O(n^a n)$. This is caused by the transduction of anchors, i.e. the similarity relationship between any two training samples can be inferred from their relations to anchors (Delalleau et al. 2005; Liu et al. 2010). Unfortunately, most previous pairwise hashing methods only leverage labeled training data without exploring the similarity information hidden in unlabeled data. And most of them only consider the similarity between query and training data but neglect their relevance, because they define two samples as one similar pair if they share common labels but do not consider how many common labels shared.

Motivated by the aforementioned observations, in this paper, we propose a novel semi-supervised deep pairwise hashing (SSDPH) to exploit both labeled and unlabeled data to learn hash functions. Specifically, the proposed method first selects multiple labeled anchors from training data, and then utilizes the pair similarity between labeled data and anchors to take advantage of semantic information. Additionally, to explore the similarity information in unlabeled data, it adopts self-ensembling to create strong ensemble targets for latent binary vectors of training data, by applying the

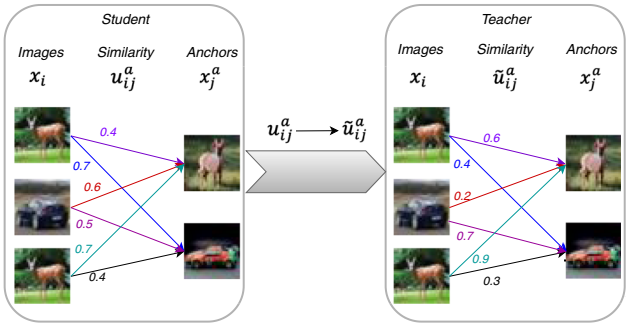


Fig. 1 The idea of anchor-based self-ensembling in the proposed hashing method. The arrows with digits represent the similarity between images and anchors. The student aims to form an analogous similarity relationship between images and anchors to that in the teacher

exponential moving average (EMA) (Laine and Aila 2016) to the predicting latent binary vectors within multiple previous training epochs, and then it extrapolates their similarity relationship to anchors, afterwards, it utilizes the extrapolated similarity relationship as the teacher to guide training data to form a consensus similarity relationship to multiple anchors under different configurations. For better illustration, we present the idea of using anchor-based self-ensembling to form a consensus predicting similarity relationship in Fig. 1. The major contributions of this paper are summarized as follows:

- We propose a novel semi-supervised deep pairwise hashing framework, which mainly utilizes anchor-based self-ensembling to take advantage of unlabeled data to preserve their similarity relationship into a low-dimensional binary space. For clarity, we show the proposed framework in Fig. 2.
- We propose a novel loss function, consisting of two cross-entropy functions to calculate the pairwise similarity relationship between labeled data and anchors as well as the relations among sampled labeled data, and a consistency cost to form a consensus similarity relationship between training data and multiple anchors.
- Unlike most of previous pairwise based deep hashing algorithms only considering the similarity between query and training samples but neglecting their relevance, we theoretically explain and present how to utilize anchors to encode query and training samples into binary codes while preserving their relevance.
- Extensive experiments on benchmark datasets demonstrate the superior retrieval performance of the proposed method over recent state-of-the-art methods on multiple tasks, including similarity, ranking order and unseen category retrieval.

This paper is structured as follows: Sect. 2 briefly reviews some related work and introduces their differences to our method; Sect.3 mainly presents the proposed semi-supervised deep hashing method; Sect. 4 exhibits, analyzes and discusses experimental results; Sect. 5 concludes this paper and points out the future work.

2 Related Work

Because numerous hashing algorithms (Wang et al. 2017) have been proposed, in this section we mainly review the most related work, supervised and semi-supervised deep hashing.

2.1 Supervised Deep Hashing

Based on the way to utilize semantic labels, current supervised deep hashing methods can be roughly categorized into three classes: point-wise, multi-wise and pairwise. Point-wise based hashing formulates the searching into a classification problem based on the rule that the classification accuracy with binary codes should be maximized. Deep learning of binary hash codes (DLBHC) (Lin et al. 2015) is an early point-wise based deep hashing method. It directly utilizes labels to train a CNN with latent binary vectors as features for image classification tasks. Unlike DLBHC attaining binary codes by simple thresholding, deep supervised convolutional hashing (DSCH) (Sapkota et al. 2018) introduces a regularization term to reduce the gap between latent binary vectors (low-dimensional embedded features with values in the continuous region) and desired binary codes (features with values in a low-dimensional binary space). Multi-wise based hashing (or ranking based hashing) learns hash functions to maximize the agreement of similarity over items between original and Hamming spaces. Network in network hashing (NINH) (Lai et al. 2015), deep semantic ranking hashing (DSRH) (Zhao et al. 2015), bit-scalable deep hashing (DRSCH) (Zhang et al. 2015) and triplet based deep binary embedding (TDBE) (Zhuang et al. 2016) adopt triplet ranking loss functions over three items to simultaneously learn CNN features and latent binary vectors. Discrete semantic ranking hashing (DSeRH) (Liu et al. 2017) directly learns image features and binary codes with maintaining the ranking order between original and binary spaces.

Pairwise based hashing is to preserve pair similarity relations of original data into a low-dimensional binary space. Convolutional neural network hashing (CNNH) (Xia et al. 2014) is one of the earliest pairwise based deep hashing methods. It learns binary codes and CNN features separately, and thus it is difficult to obtain optimal binary codes. To address this issue, deep pairwise based supervised hashing (DPSH) (Li et al. 2015), deep hashing network (DHN) (Zhu et al. 2016) and deep supervised hashing (DSH) (Liu et al. 2016) utilize pairwise labels to simultaneously learn CNN features and latent binary representations. HashNet (Cao et al. 2017)

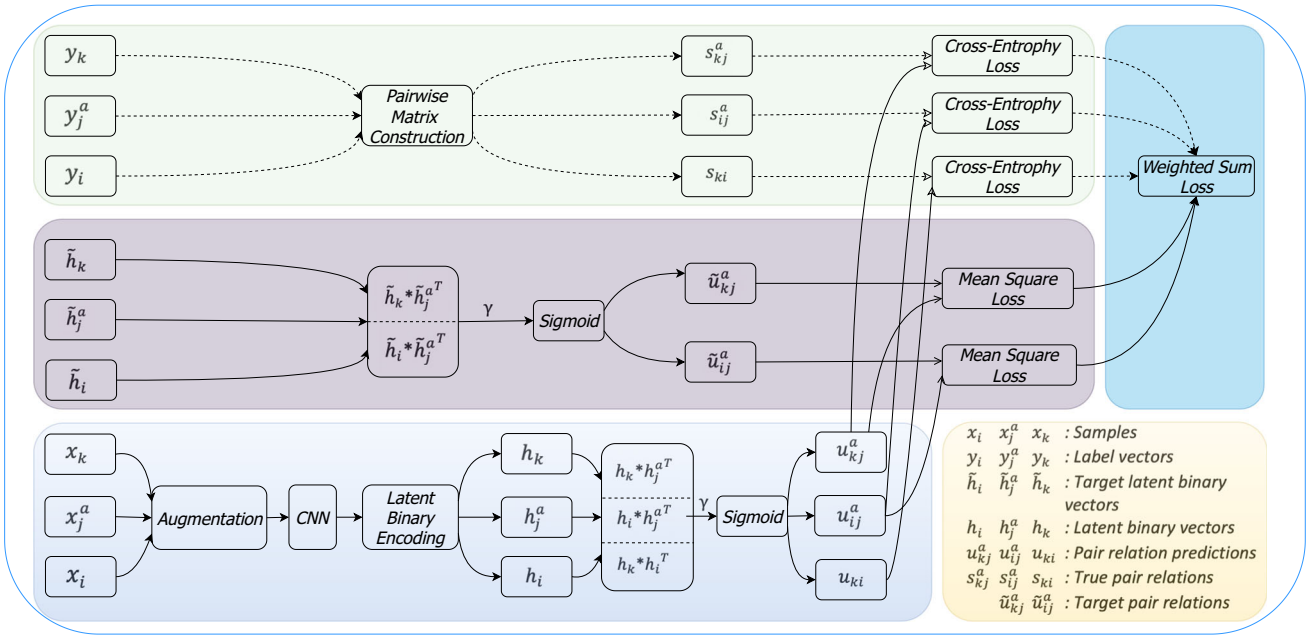


Fig. 2 The flowchart of the proposed deep framework

addresses the ill-posed gradient and imbalanced data when learning binary codes. Deep cauchy hashing (DCH) (Cao et al. 2018) is to maintain a similarity relationship of each pair with short Hamming distances. Unlike previous deep hashing methods using two identical relaxed binary matrices to maintain pair relations, deep asymmetric pairwise hashing (DAPH) (Shen et al. 2017) and asymmetric deep supervised hashing (ADSH) (Jiang and Li 2018) directly learn two distinct binary matrices to preserve pairwise similarity relations. Pairwise based deep ranking hashing (PDRH) (Shi et al. 2018) learns hash functions using pairwise matrices to preserve the pair relationship with considering the relevance of similar data.

Among the three manners to exploit semantic labels, point-wise based hashing is scalable but its rule is inferior to that of multi-wise and pairwise based hashing, because it usually fails to take the similarity relationship of training data into consideration; Multi-wise based hashing using a triplet ranking loss usually consumes higher training time costs than point-wise and pairwise based hashing (Sapkota et al. 2018; Jiang and Li 2018), because of large-scale triplets. Additionally, it often exhibits inferior performance to pairwise based hashing on the single-label databases for preserving the similarity relationship of samples, probably because the triplets focus on the similarity difference over any three items rather than directly emphasize the similarity relationship of training data. Generally pairwise based hashing better preserves the similarity relationship among samples than multi-wise or point-wise based hashing, and consumes less time costs than multi-wise based hashing. However, most pairwise based hashing algorithms neglect the relevance of similar samples. *By contrast, the proposed method SSDPH utilizes the pairwise similarity to preserve the similarity information of samples, and meanwhile maintains the relevance between the query and training samples through calculating their similarities to anchors.*

2.2 Semi-Supervised Deep Hashing

Supervised deep hashing only exploits the semantic information of labeled data and fails to explore the similarity information in unlabeled data. To address this issue, several semi-supervised deep hashing methods have been proposed to simultaneously leverage the similarity information of both labeled and unlabeled data. Semi-supervised deep hashing (SSDH) (Zhang and Peng 2017) and bipartite graph deep hashing (BGDH) (Yan et al. 2017) exploit the transduction of graphs to explore the underlying structure of unlabeled data. SSDH adopts the triplet ranking loss function to exploit the semantic labels, while BGDH utilizes pairwise labels to preserve the similarity information of labeled data but neglect the relevance among similar samples. Deep semantic hashing with generative adversarial networks (DSH-GANs) (Qiu et al. 2017), SSH-GAN (Zhang et al. 2018) and semi-supervised generative adversarial hashing (SSGAH) (Wang et al. 2018) adopt triplet ranking loss functions to maintain the similarity relationship over items and apply GANs to both labeled and unlabeled data to explore their similarity information. SSDH-GAN and SSGAH make use of unlabeled data by using generative models to model unlabeled data or select margin examples, while DSH-GANs explores

the similarity information of unlabeled data by pre-training semi-supervised GANs.

Different from previous semi-supervised hashing methods using graphs or GANs to explore the similarity information of unlabeled data, the proposed method SSDPH exploits unlabeled data using self-ensembling to extrapolate their predictions and form consensus similarity relations to anchors, which can effectively preserve the similarity relationship of unlabeled data. Additionally, SSDH, DSH-GANs, SSH-GAN and SSGAH adopt triplet ranking loss functions to exploit semantic information of labeled data, while SSDPH utilizes pairwise labels to leverage semantic information with lower training time costs. Moreover, unlike BGDH neglecting the relevance of similar samples, SSDPH can maintain their relevance through calculating their similarities to anchors.

3 Methods

3.1 Temporal Ensembling

Given N labeled and unlabeled training samples $\mathbf{X} = [\mathbf{X}^l; \mathbf{X}^u] = \{\mathbf{x}_i\}_{i=1}^N$, let $\mathbf{y} = \{y_i\}_{i=1}^n$ ($y_i \in \{0, 1, \ldots, C-1\}$) represent the labels of n labeled samples, where $\mathbf{X}^l$ and $\mathbf{X}^u$ stand for labeled and unlabeled data, respectively, $\mathbf{x}_i$ denotes the ith training sample and C is the number of classes. Suppose that $\mathbf{z}_i \in \mathbb{R}^C$ is a label prediction of $\mathbf{x}_i$, whose target vector $\tilde{\mathbf{z}}_i \in \mathbb{R}^C$ is obtained by applying EMA to label predictions within multiple previous training epochs. Specifically, in each training epoch, $\mathbf{z}_i$ is firstly accumulated into an ensemble vector $\mathbf{z}_i^e \in \mathbb{R}^C$, i.e. $\mathbf{z}_i^e = \alpha \mathbf{z}_i^e + (1-\alpha)\mathbf{z}_i$, and then $\tilde{\mathbf{z}}_i$ is calculated by $\tilde{\mathbf{z}}_i = \mathbf{z}_i^e/(1-\alpha^t)$, where α is a momentum term to control how far the ensemble reaches into training history, and t is the current number of training epochs.

Let L be the index set of labeled data $\mathbf{X}^l$, and B represent the index set of data selected from $\mathbf{X}$. Temporal ensembling (TE) (Laine and Aila 2016) adopts a cross-entropy function to exploit the semantic labels and a consistency cost to explore the semantic information of unlabeled data. Its loss function is:

$$J_{TE} = \frac{1}{|B|} \sum_{i \in (B \cap L)} log\ \mathbf{z}_i[y_i] + \frac{\omega(t)}{C\,|B|} \sum_{i \in B} \|\mathbf{z}_i - \tilde{\mathbf{z}}_i\|_2^2, \quad (1)$$

where $|B|$ is the number of selected data, $\omega(t)$ is a time-dependent weighting function and it is to gradually enhance the weight of unlabeled data. Eq. (1) infers that TE utilizes the ensemble target $\tilde{\mathbf{z}}_i$ ($i \in B$) as the teacher to guide the predictions of unlabeled data through minimizing the difference between the predictions $\mathbf{z}_i$ and $\tilde{\mathbf{z}}_i$.

3.2 SSDPH Using Anchor-Based Self-Ensembling

TE suggests that using ensemble targets as the teacher to guide the predictions of unlabeled data can effectively and efficiently explore the semantic information of unlabeled data (Laine and Aila 2016; Miyato et al. 2018). Additionally, although pairwise based hashing consumes less time costs than multi-wise based hashing, it still takes high computation and storage costs on exploiting the pairwise relation between any two samples, e.g. $\mathcal{O}(n^2)$ for n labeled samples. To further reduce the costs, we propose to select multiple anchors from labeled training samples and preserve a pairwise relationship between anchors and training samples, because the relations among samples can be calculated through computing their similarities to anchors (Liu et al. 2010; Shi et al. 2017a; Hu et al. 2019). To this end, we propose a novel semi-supervised deep pairwise hashing method using anchors and self-ensembling. For clarity, we present the flowchart of our method in Fig. 2.

3.2.1 Definitions of Pairwise Label Matrix and Hash Function

Given n labeled samples $\mathbf{X}^l = \{\mathbf{x}_i\}_{i=1}^n$, let $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{M}$ if any two samples $\mathbf{x}_i$ and $\mathbf{x}_j$ share at least one common label; otherwise, $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{C}$, where $\mathbb{M}$ and $\mathbb{C}$ stand for neighbor-pair and nonneighbor-pair sets, respectively. Suppose that one mini-batch contains n^b selected labeled samples and $\mathbf{S} \in \mathbb{R}^{n^b \times n^b}$ represents the pair relationship of selected labeled samples, it is defined as:

$$s_{ij} = \begin{cases} 1 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{M} \\ 0 & (\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{C}. \end{cases} \quad (2)$$

Hashing is to encode the high-dimensional data into compact binary codes with maintaining their similarity relations. Specifically, given a stochastic neural network $f_\theta(\cdot)$ with parameters $\theta = \{\theta_1, \theta_2, \ldots, \theta_m\}$, where m is the number of bits, for any sample $\mathbf{x}$, its k-th hash function is defined as:

$$h_k(\mathbf{x}) = sgn(f_{\theta_k}(\mathbf{x})), \quad (3)$$

where $sgn(\cdot)$ is a non-linear function with the definition: $sgn(f_{\theta_k}(\mathbf{x})) = 1$ if $f_{\theta_k}(\mathbf{x}) \geq 0$; otherwise, $sgn(f_{\theta_k}(\mathbf{x})) = -1$.

Let $h(\mathbf{x}) = [h_1(\mathbf{x}), h_2(\mathbf{x}), \ldots, h_m(\mathbf{x})]$ be m hash codes of $\mathbf{x}$. For any two samples $\mathbf{x}_i$ and $\mathbf{x}_j$, there exists $-m \leq h(\mathbf{x}_i) \circ h(\mathbf{x}_j) \leq m$, where $\circ$ denotes the inner product. Usually, a larger value of $h(\mathbf{x}_i) \circ h(\mathbf{x}_j)$ means a higher similarity between $\mathbf{x}_i$ and $\mathbf{x}_j$. Because $sgn(\cdot)$ is non-differential, in our deep architecture we replace it with a differential hyperbolic tangent (tanh) function, $tanh(f_{\theta_k}(\mathbf{x})) \in [-1, 1]$, which is the closest convex region of the non-convex region $\{-1, 1\}$.

In our modeling, $\mathbf{h} = tanh(f_\theta(\mathbf{x})) \in [-1, 1]^m$ generated by the latent binary encoding layer is a latent binary vector of $\mathbf{x}$.

3.2.2 Formulation and Procedure

For N labeled and unlabeled training samples $\mathbf{X} = [\mathbf{X}^l; \mathbf{X}^u]$, suppose that $\mathbf{X}^l = \{\mathbf{x}_i\}_{i=1}^n$ belongs to C categories and $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n$ ($\mathbf{y}_i \in \{0, 1\}^C$) denotes the corresponding label vectors. We randomly select n^a ($n^a << n$) samples $\mathbf{X}^a$ from $\mathbf{X}^l$ as anchors, with each category containing $\frac{n^a}{C}$ ones. For any one sample $\mathbf{x}_i \in \mathbf{X}^l$ and one anchor $\mathbf{x}_j^a \in \mathbf{X}^a$, let $\mathbf{h}_i$ and $\mathbf{h}_j^a$ represent their latent binary vectors, respectively. Because a larger value of $\mathbf{h}_i\mathbf{h}_j^{aT}$ means that $\mathbf{x}_i$ and $\mathbf{x}_j$ are more similar, we aim to learn hash functions so that $\mathbf{h}_i\mathbf{h}_j^{aT} > 0$ if $(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{M}$; otherwise, $\mathbf{h}_i\mathbf{h}_j^{aT} < 0$. To achieve this goal, we define $u_{ij}^a = \sigma(\gamma\mathbf{h}_i\mathbf{h}_j^{aT}) = \frac{1}{1+e^{-\gamma\mathbf{h}_i\mathbf{h}_j^{aT}}} \in [0, 1]$ to denote their pair relations in the low-dimensional binary space, and $u_{ij}^a \to 1$ if $(\mathbf{x}_i, \mathbf{x}_j^a) \in \mathbb{M}$, otherwise $u_{ij}^a \to 0$, where $\sigma(\cdot)$ represents the sigmoid function. $\gamma > 0$ is a constant to regularize the value of $\mathbf{h}_i\mathbf{h}_j^{aT}$, i.e. the larger γ, the smaller $\mathbf{h}_i\mathbf{h}_j^{aT}$ leading to $u_{ij}^a \to 1$, and vice versa.

Let B denote the index set of selected data from $\mathbf{X}$, L represent the index set of labeled data $\mathbf{X}^l$, and L^a be the index set of anchors. To maintain the relationship between anchors and selected labeled data in the transformed binary space, we utilize a cross-entropy function to learn hash functions:

$$\begin{aligned} J_1 = \frac{1}{f_1} \sum_{i\in(B\cap L), j\in L^a} & -s_{ij}^a log\, u_{ij}^a + (s_{ij}^a - 1) log\,(1 - u_{ij}^a) \\ s.t.\ u_{ij}^a &= \frac{1}{1+e^{-\gamma\mathbf{h}_i\mathbf{h}_j^{aT}}}, \\ \mathbf{h}_i = tanh(f_\theta(\mathbf{x}_i)),\ \mathbf{h}_j^a &= tanh(f_\theta(\mathbf{x}_j^a)), \end{aligned} \tag{4}$$

where $f_1 = |B \cap L|\,|L^a|$ is the product of $|B \cap L|$ and $|L^a|$, $|\cdot|$ is to denote the number of elements in a set, and when $i \in (B \cap L)$ and $j \in L^a$, $s_{ij}^a \in \mathbf{S}^a$ denotes the similarity between selected labeled data and anchors.

To further take advantage of the semantic information of selected labeled data and enhance their connection, we also preserve their pair relationship in the transformed binary space by using the following cross-entropy function:

$$J_2 = \frac{1}{f_2} \sum_{i,j\in(B\cap L)} -s_{ij} log\, u_{ij} + (s_{ij} - 1) log\,(1 - u_{ij}), \tag{5}$$

where $f_2 = |B \cap L|^2$.

Eq. (4) and Eq. (5) are to make use of semantic information of labeled data. Next, we show how to explore the semantic similarity information hidden in unlabeled data. For one selected sample $\mathbf{x}_i \in \mathbf{X}$, $\mathbf{h}_i$ denotes its latent binary vector. Let $\tilde{\mathbf{h}}_i$ represent its target latent binary vector obtained by applying EMA to $\mathbf{h}_i$ within multiple previous training epochs. Specifically, in each training epoch, $\mathbf{h}_i$ is firstly accumulated into an ensemble vector $\mathbf{h}_i^e$ by $\mathbf{h}_i^e = \alpha_1\mathbf{h}_i^e + (1 - \alpha_1)\mathbf{h}_i$, and then $\tilde{\mathbf{h}}_i$ is generated by $\tilde{\mathbf{h}}_i = \mathbf{h}_i^e/(1 - \alpha_1^t)$, where α_1 is a momentum term to control how far the ensemble latent binary vector $\mathbf{h}_i^e$ reaches into training history. Similarly, for one anchor sample $\mathbf{x}_j^a$, let $\mathbf{h}_j^a$ be its latent binary vector. Its ensemble and target latent binary vectors are obtained by $\mathbf{h}_j^{ae} = \alpha_2\mathbf{h}_j^{ae} + (1-\alpha_2)\mathbf{h}_j^a$ and $\tilde{\mathbf{h}}_j^a = \mathbf{h}_j^{ae}/(1-\alpha_2^t)$, respectively. In order to form a consensus similarity relationship to anchors for each selected training sample, we adopt a least-squares model as follows:

$$J_3 = \frac{1}{f_3} \sum_{i\in B, j\in L^a} \left\| u_{ij}^a - \tilde{u}_{ij}^a \right\|_2^2, \tag{6}$$

where $f_3 = |B|\,|L^a|$, and $\tilde{u}_{ij}^a = \frac{1}{1+e^{-\gamma\tilde{\mathbf{h}}_i\tilde{\mathbf{h}}_j^{aT}}}$ can be regarded as a teacher to guide the similarity relationship between the selected sample $\mathbf{x}_i$ and the anchor $\mathbf{x}_j^a$. Note that $\tilde{u}_{ij}^a$ is a known variable in Eq. (6), because $\tilde{\mathbf{h}}_i$ and $\tilde{\mathbf{h}}_j^a$ are determined by the latent binary vectors of multiple previous training epochs.

Combining the three terms J_1, J_2 and J_3, we obtain the proposed loss function for SSDPH as follows:

$$\begin{aligned} J = \frac{1}{f_1} \sum_{i\in(B\cap L), j\in L^a} & -s_{ij}^a log\, u_{ij}^a + (s_{ij}^a - 1) log\,(1 - u_{ij}^a) \\ + \frac{\lambda}{f_2} \sum_{i,j\in(B\cap L)} & -s_{ij} log\, u_{ij} + (s_{ij} - 1) log\,(1 - u_{ij}) \\ + \frac{\omega(t)}{f_3} \sum_{i\in B, j\in L^a} & \left\| u_{ij}^a - \tilde{u}_{ij}^a \right\|_2^2 \\ s.t.\ u_{ij}^a &= \frac{1}{1+e^{-\gamma\mathbf{h}_i\mathbf{h}_j^{aT}}}, \\ \mathbf{h}_i = tanh(f_\theta(\mathbf{x}_i)),\ \mathbf{h}_j^a &= tanh(f_\theta(\mathbf{x}_j^a)), \end{aligned} \tag{7}$$

where λ is to weight the usage of pairwise labels among selected labeled data, and $\omega(t)$ is a time-dependent weighting function to gradually explore the semantic similarity information in unlabeled data.

In Eq. (7), the first two terms J_1 and J_2 to exploit the semantic information of labeled data are different from previous supervised pairwise based deep hashing methods (Jiang and Li 2018; Liu et al. 2016; Zhu et al. 2016), which only utilize the semantic similarity relationship between training data and anchors or leverage the relations between selected labeled data in each mini-batch, thereby potentially losing some semantic information. Additionally, the third term J_3 is mainly to explore the semantic similarity relationship among unlabeled data. It is worth noting that although the third term in Eq. (7) appears to be the same as the consistency cost in Eq. (1), they have totally different meanings. This is because the third term aims to form a consensus similarity relationship between training data and anchors, while the consistency cost in Eq. (1) does not consider the relationship between any two samples.

Algorithm 1: SSDPH

Input: Training data $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{N}$, label index set L,
label vectors $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{n}$, anchor index set L^a, bit number m,
parameters γ, λ, ensembling momentum α_1, α_2
number of training epochs: T
unsupervised ramp-up function: $\omega(t)$,
stochastic neural network with parameters θ: $f_\theta(\cdot)$,
stochastic input augmentation function: $g(\cdot)$
Output: Parameters θ

1. **Initialization:**
   $\mathbf{H}^e \leftarrow \mathbf{0}_{N\times m}$, ▷ ensemble latent binary vectors
   $\tilde{\mathbf{H}} \leftarrow \mathbf{0}_{N\times m}$, ▷ target latent binary vectors
   $\mathbf{H}^{ae} \leftarrow \mathbf{0}_{n^a\times m}$, ▷ ensemble latent binary vectors of anchors
   $\tilde{\mathbf{H}}^a \leftarrow \mathbf{0}_{n^a\times m}$, ▷ target latent binary vectors of anchors
2. **for** t **in** $[1, T]$ **do**
3. **for** each minibatch B **do**
4. $\mathbf{h}_{i\in B}, \mathbf{h}^a_{j\in L^a} \leftarrow f_\theta(g(\mathbf{x}_{i\in B}), g(\mathbf{x}^a_{j\in L^a}))$
5. $\mathbf{S}_a, \mathbf{S} \leftarrow$ Eq. (2)
6. loss $\leftarrow$ Eq. (7)
7. updating θ using optimizers, e.g. Adam
8. $\mathbf{H}^{ae} \leftarrow \alpha_2\mathbf{H}^{ae} + (1-\alpha_2)\mathbf{H}^{ae}$
9. $\tilde{\mathbf{H}}^a \leftarrow \mathbf{H}^{ae}/(1-\alpha_2^t)$
10. **end for**
11. $\mathbf{H}^e \leftarrow \alpha_1\mathbf{H}^e + (1-\alpha_1)\mathbf{H}$
12. $\tilde{\mathbf{H}} \leftarrow \mathbf{H}^e/(1-\alpha_1^t)$
13. **end for**

Based on Eq. (7), the parameters θ can be optimized using any optimizer, e.g. Adam (Kingma and Ba 2014). For better illustration, we present the detailed procedure of solving Eq. (7) to learn hash functions in Algorithm 1: SSDPH.

3.2.3 Data Encoding

In the query stage, encoding one query $\mathbf{x}_q$ and training samples $\mathbf{X}$ can be divided into two cases: (i) only preserving the similarity (i.e. without ranking) between query and training samples; (ii) preserving the relevance between query and training samples to maintain their rank orders. Specifically, given one query $\mathbf{x}_q$ and two training samples $\mathbf{x}_1$ and $\mathbf{x}_2$, suppose that $\mathbf{x}_q$ has two common labels with $\mathbf{x}_1$ and shares one common label with $\mathbf{x}_2$. Based on the definition in Section 3.2.1, $\mathbf{x}_q$ is similar to both $\mathbf{x}_1$ and $\mathbf{x}_2$. But the relevance between $\mathbf{x}_q$ and $\mathbf{x}_1$ is larger than that between $\mathbf{x}_q$ and $\mathbf{x}_2$, i.e. $r(\mathbf{x}_q, \mathbf{x}_1) > r(\mathbf{x}_q, \mathbf{x}_2)$.

When only considering the similarity between any two samples $\mathbf{x}_i$ and $\mathbf{x}_j$, based on the definition in Eq. (7), their similarity relationship is defined by $u_{ij} = \frac{1}{1+e^{-\gamma\mathbf{h}_i\mathbf{h}_j^T}}$, which is determined by $\mathbf{h}_i\mathbf{h}_j^T$. Therefore, to preserve the similarity relationship between the query $\mathbf{x}_q$ and training samples $\mathbf{X}$, we encode them by:

$$h(\mathbf{x}_q) = sgn(f_\theta(\mathbf{x}_q)). \tag{8a}$$

$$h(\mathbf{X}) = sgn(f_\theta(\mathbf{X})). \tag{8b}$$

When taking the relevance of training samples to the query into account, we utilize the transduction of anchors to calculate their relevance. In Eq. (7), u^a_{ik} denotes the similarity relationship between $\mathbf{x}_i$ and $\mathbf{x}^a_k$. Because $u^a_{ik} \in [0, 1]$ is not a discrete value, we utilize $\frac{sgn(2u^a_{ik}-1)+1}{2} \in \{0, 1\}$ to represent their similarity relationship. Given two samples $\mathbf{x}_i$, $\mathbf{x}_j$ and one anchor $\mathbf{x}^a_k \in \mathbf{X}^a$, let $p(\mathbf{x}^a_k|\mathbf{x}_i) = \frac{sgn(2u^a_{ik}-1)+1}{2}$ denote the similarity relationship of $\mathbf{x}_i$ to $\mathbf{x}^a_k$, where $p(\mathbf{x}^a_k|\mathbf{x}_i) = 1$ if $\mathbf{x}^a_k$ and $\mathbf{x}_i$ are similar, otherwise, $p(\mathbf{x}^a_k|\mathbf{x}_i) = 0$. $p(\mathbf{x}_j|\mathbf{x}^a_k) = \frac{sgn(2u^a_{kj}-1)+1}{2}$ represents the similarity relationship of $\mathbf{x}^a_k$ to $\mathbf{x}_j$. Then the relevance of $\mathbf{x}_i$ to $\mathbf{x}_j$ is calculated by:

$$r(\mathbf{x}_j|\mathbf{x}_i) = \sum_{k\in L^a} p(\mathbf{x}_j|\mathbf{x}^a_k)p(\mathbf{x}^a_k|\mathbf{x}_i). \tag{9}$$

Because of $sgn(2u^a_{kj} - 1) = sgn(sgn(f_\theta(\mathbf{x}_j))\, sgn(f_\theta(\mathbf{x}^a_k))^T)$, $r(\mathbf{x}_j|\mathbf{x}_i)$ equals:

$$\begin{aligned} r(\mathbf{x}_j|\mathbf{x}_i) &= \frac{\sum_{k\in L_a} \frac{sgn(sgn(f_\theta(\mathbf{x}_j))sgn(f_\theta(\mathbf{x}^a_k))^T)+1}{2}}{\frac{sgn(sgn(f_\theta(\mathbf{x}^a_k))sgn(f_\theta(\mathbf{x}_i))^T)+1}{2}} \\ &= \frac{\frac{sgn(sgn(f_\theta(\mathbf{x}_j))sgn(f_\theta(\mathbf{X}^a))^T)+\mathbf{1}_{n^a}}{2}}{\frac{sgn(sgn(f_\theta(\mathbf{X}^a))sgn(f_\theta(\mathbf{x}_i))^T)+\mathbf{1}^T_{n^a}}{2}}, \end{aligned} \tag{10}$$

where $\mathbf{1}_{n^a} \in \mathbb{R}^{n^a}$ is a row vector with all entries being ones. Additionally, because $u^a_{jk} = u^a_{kj} = \frac{1}{1+e^{-\gamma\mathbf{h}_j\mathbf{h}_k^{aT}}}$, there exists $r(\mathbf{x}_j|\mathbf{x}_i) = r(\mathbf{x}_i|\mathbf{x}_j)$. Based on Eq. (10), to calculate the relevance between query and training data, we encode them by:

$$h^r(\mathbf{x}_q) = \frac{sgn(sgn(f_\theta(\mathbf{x}_q))sgn(f_\theta(\mathbf{X}^a))^T) + \mathbf{1}_{n^a}}{2}, \tag{11a}$$

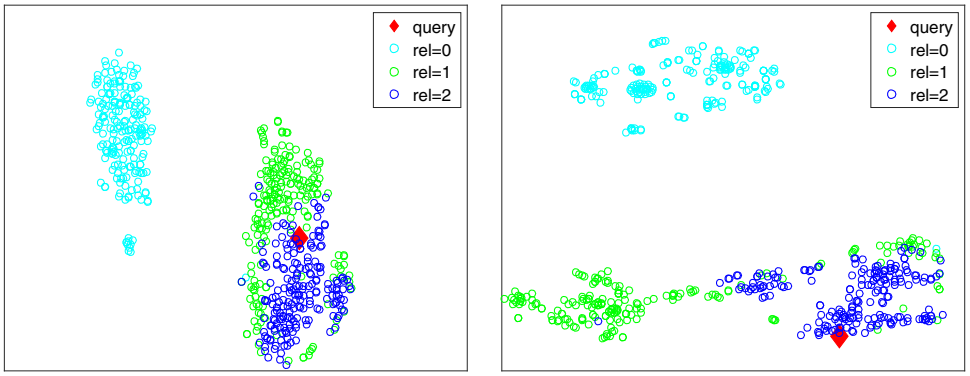
$$h^r(\mathbf{X}) = \frac{sgn(sgn(f_\theta(\mathbf{X}))sgn(f_\theta(\mathbf{X}^a))^T) + \mathbf{1}_{N\times n^a}}{2}, \tag{11b}$$

where $\mathbf{1}_{N\times n^a} \in \mathbb{R}^{N\times n^a}$ is a matrix with all entries being ones, $h^r(\mathbf{x}_q) \in \{0, 1\}^{n^a}$ and $h^r(\mathbf{X}) \in \{0, 1\}^{N\times n^a}$.

Next, we theoretically demonstrate that Eq. (11) can maintain the relevance of similar data.

Theorem 1 *Suppose that each category in the n^a anchors contains $\frac{n^a}{C}$ labels. For any three samples $\mathbf{x}_q$, $\mathbf{x}_1$ and $\mathbf{x}_2$, and y_q, y_1 and y_2 represent their label sets, respectively, when $r(\mathbf{x}_q, \mathbf{x}_1) > r(\mathbf{x}_q, \mathbf{x}_2)$, based on Eq. (11), there exists $h^r(\mathbf{x}_q) \circ h^r(\mathbf{x}_1) > h^r(\mathbf{x}_q) \circ h^r(\mathbf{x}_2)$.*

Proof Because $\mathbf{x}_q$ consists of $|y_q|$ labels, based on Eq. (11), we have $h^r(\mathbf{x}_q) \in \left\{\{0\}^{n^a-\frac{|y_q|n^a}{C}}, \{1\}^{\frac{|y_q|n^a}{C}}\right\}$, which contains $\frac{|y_q|n^a}{C}$ positive ones and $n^a - \frac{|y_q|n^a}{C}$ zeros, because $\mathbf{x}_q$ are similar to $\frac{|y_q|n^a}{C}$ anchors and dissimilar to $n^a - \frac{|y_q|n^a}{C}$ anchors.



(a) @Eq. (8) **(b)** @Eq. (11)

Fig. 3 The projection of query and training data onto a two-dimensional plane using 48-bit binary codes generated by Eq. (8) and Eq. (11). $rel = 2$ and $rel = 1$ denote the similar training data with different relevance to the query, while $rel = 0$ means the dissimilar samples to the query

Similarly, we can obtain $h^r(\mathbf{x}_1) \in \left\{\{0\}^{n^a - \frac{|y_1|n^a}{C}}, \{1\}^{\frac{|y_1|n^a}{C}}\right\}$ and $h^r(\mathbf{x}_2) \in \left\{\{0\}^{n^a - \frac{|y_2|n^a}{C}}, \{1\}^{\frac{|y_2|n^a}{C}}\right\}$. Then $h^r(\mathbf{x}_q) \circ h^r(\mathbf{x}_1) = \frac{|y_q \cap y_1|n^a}{C}$, because $h^r(\mathbf{x}_q)$ and $h^r(\mathbf{x}_1)$ share $\frac{|y_q \cap y_1|n^a}{C}$ positive ones. And $h^r(\mathbf{x}_q) \circ h^r(\mathbf{x}_2) = \frac{|y_q \cap y_2|n^a}{C}$. Because of $r(\mathbf{x}_q, \mathbf{x}_1) > r(\mathbf{x}_q, \mathbf{x}_2)$, there exists $|y_q \cap y_1| > |y_q \cap y_2|$, which leads to $h^r(\mathbf{x}_q) \circ h^r(\mathbf{x}_1) > h^r(\mathbf{x}_q) \circ h^r(\mathbf{x}_2)$. Therefore, Theorem 1 is proved.

For better illustration, we utilize Fig. 3 to clearly show the role of anchors in preserving the relevance between query and training samples, where we randomly select one query and 600 training images from one multi-label database NUS-WIDE, and then utilize SSDPH with Eq. (8) or (11) to encode them into binary codes. Fig. 3 displays the projection of their 48-bit binary codes onto a two-dimensional plane using t-SNE (Maaten and Hinton 2008). As we can see, Eq. (8) only preserves the similarity relationship between the query and training samples, while Eq. (11) can preserve their relevance, thereby maintaining their rank orders.

3.2.4 Implementation Details

The proposed SSDPH can be applied to different multimedia data. Here we test it on image retrieval. We implement SSDPH with the Pytorch framework and utilize AlexNet (Krizhevsky et al. 2012) or ResNet18 (He et al. 2016) as the backbone network, which is pre-trained on the ImageNet database (Russakovsky et al. 2015). By default, AlexNet is the backbone network of SSDPH. Additionally, we adopt the optimizer Adam (Kingma and Ba 2014) to update the network parameters. We initialize Adam momentum parameters $\beta_1 = 0.9$ and $\beta_2 = 0.99$. The size of input images is 224×224, and each image is augmented with random translations ($\{\Delta x, \Delta y\} \sim [-32, 32]$) and horizontal flip ($p = 0.5$). We totally run 100 epochs for training with mini-batch size being 100. Similar to (Laine and Aila 2016), the function $\omega(t)$ increases from 0 to the maximum ω_{max} during the first 40 epochs. $\omega(t)$ is a Gaussian ramp-up function $e^{-5\|1-T_1\|^2}$, where T_1 linearly advances from 0 to 1 during the first 40 epochs and then keeps unchanged. The learning rate η also ramps up to the maximum η_{max} during the first 40 epochs and then keeps unchanged from 40 to 70 epochs, but gradually decreases to 0 using a Gaussian ramp-down function $e^{-7.5\|1-T_2\|^2}$, where T_2 linearly decreases from 1 to 0 during the last 30 epochs. β_1 also goes down to 0.5 using the Gaussian ramp-down function while it keeps unchanged during the first 70 epochs. β_2 becomes 0.999 after the first 40 epochs.

4 Experimental Results and Analysis

We evaluate the proposed method SSDPH on three kinds of image retrieval tasks, including similarity, ranking order and unseen categories. We conduct similarity retrieval experiments on CIFAR-10 (Krizhevsky and Hinton 2009), NUS-WIDE (Chua et al. 2009) and COCO (Lin et al. 2014) databases, ranking order retrieval experiments on multi-label databases NUS-WIDE and COCO, and run experiments for unseen category retrieval on CIFAR-10, MNIST (LeCun et al. 1998) and NUS-WIDE databases. These four datasets are introduced as follows.

CIFAR-10 consists of 60K single-label color images belonging to ten categories, and each category has 6K images with the size of 32×32 pixels.

NUS-WIDE is composed of around 270,000 images collected from Flickr, and each image contains one or multiple labels of 81 semantic concepts. Following (Shen et al. 2015b; Zhang and Peng 2017), we totally select 195,834 images belonging to the 21 most frequent concepts, where each concept is made up at least 5000 images. We crop and resize each image to 128×128 pixels.

COCO has around 328,000 images belonging to 91 object types. We utilize the 2014 training and validation sets. They

are made up by 123,287 images in total, where each one contains one or multiple labels from 80 object categories. Each images is cropped and resized to 128 × 128 pixels.

MNIST consists of handwritten digits from '0' to '9', and totally contains 70,000 images, with each one represented by a 784-dimensional vector.

All images of the four databases are scaled to 224 × 224 before feeding into the networks, because 224 × 224 is the default size for pre-trained models.

4.1 Experimental Setting

We compare SSDPH against nineteen state-of-the-art methods including nine non-deep and ten deep hashing algorithms. Non-deep hashing methods include three unsupervised algorithms: LSH (Indyk and Motwani 1998), SH (Weiss et al. 2009), ITQ (Gong et al. 2013), and six supervised algorithms: SDH (Shen et al. 2015b), KSDH (Shi et al. 2016), COSDISH (Kang et al. 2016), ADGH (Shi et al. 2017a), CGH (Li et al. 2013), and DSeRH_L (Liu et al. 2017) . Deep hashing methods contain six supervised deep hashing algorithms: DSCH (Sapkota et al. 2018), NINH (Lai et al. 2015), DSRH (Zhao et al. 2015), DSH (Liu et al. 2016), DPSH (Li et al. 2015) and ADSH (Jiang and Li 2018), and four semi-supervised deep hashing approaches: SSDH (Zhang and Peng 2017), BGDH (Yan et al. 2017), DSH-GANs (Qiu et al. 2017) and SSGAH (Zhang et al. 2018). Additionally, we show the supervised case of SSDPH, namely SSDPH*, which only utilizes labeled data for training. For all non-deep hashing methods, we report their performance on CIFAR-10 and NUS-WIDE databases with public available hand-craft features. Each image is represented by a 512-dimensional GIST (Oliva and Torralba 2001) feature vector for CIFAR-10 and a 500-dimensional bag-of-words (BoW) feature vector for NUS-WIDE. For the non-deep supervised hashing methods, we also present their performance on all the four databases with 4096-dimensional CNN feature vectors extracted from the $fc7$ layer of the pre-trained AlexNet, which is firstly trained on the four databases in a manner of classification. For deep hashing methods, we re-implement DSCH, NINH, DSRH, DSH, DPSH, ADSH and SSDH with the Pytorch framework, and adopt AlexNet as their backbone networks based on their provided codes for a fair comparison. For BGDH, DSH-GANs and SSGAH, we do not find public codes, but directly copy their results from some publications. So, in some experiments, we do not report results of these three methods. For the parameters including α_1, α_2, η_{max}, ω_{max}, n^a, γ and λ in SSDPH, we present their detailed settings for each experiment in the Appendix.

To evaluate the similarity between query and training samples, we utilize mean average precision (MAP), precision and precision-recall (PR) curve as evaluation metrics; For the retrieval performance on ranking order, we adopt normalized discounted cumulative gain (NDCG) (Järvelin and Kekäläinen 2000) as the evaluation metric; For unseen category retrieval, we mainly utilize the evaluation metric MAP. Given a set of queries, MAP is the mean of average precision (AP) for each query. AP is defined as:

$$AP@R = \frac{\sum_{k=1}^{R} P(k)\delta(k)}{\sum_{k=1}^{R} \delta(k)}, \tag{12}$$

where R is the number of top returned samples, $P(k)$ is the precision at cut-off k in the list, and $\delta(k) = 1$ if the sample ranked at k-th position contains at least one common label with the query, otherwise, $\delta(k) = 0$.

NDCG is the normalization of the discounted cumulative gain (DCG) calculated by (Järvelin and Kekäläinen 2000):

$$DCG = rel_1 + \sum_{k=2}^{R} \frac{rel_k}{log_2(k)}, \tag{13}$$

where k is the ranking position and rel_k is the relevance between the k-th returned sample and the query.

4.2 Experiments for Similarity Retrieval

We run experiments on CIFAR-10, NUS-WIDE and COCO databases to evaluate the similarity retrieval performance of SSDPH* and SSDPH using Eq. (8), and compare them against LSH, SH, ITQ, SDH, KSDH, COSDISH, ADGH, NINH, DSH, DPSH, DSCH, ADSH and SSDH. Among them, ADGH is only used for the single-label database CIFAR-10, because it cannot be directly applied to multi-label tasks. We do not show the results of non-deep hashing algorithms with hand-craft features on the COCO database, which does not have public available non-deep features. Additionally, we do not display the results of BGDH, DSH-GANs and SSGAH on NUS-WIDE, because their experimental settings might be not exactly same as ours and there are no public available codes. We also do not show their results on the COCO database, because there are no reported results. Following (Zhang and Peng 2017), for the CIFAR-10 database, we randomly select 100 images per class to construct a query set, which contains 1,000 images in total. The remaining 59,000 images are used as a base set, from which we randomly select 500 images per class as a labeled set (5,000 images in total) and use the rest of images as unlabeled ones. For the NUS-WIDE database, we randomly select 2,100 images as a query set, with each concept containing at least 100 images, and adopt the rest of 193,734 images as a base set, from which we randomly choose 10,500 images (at least 500 images per concept) to construct a labeled set and utilize the remaining ones as unlabeled images. For the

Table 1 Similarity retrieval performance (MAP) by using the labeled set for retrieval on top 5,000 retrieved samples of different hashing methods on the CIFAR-10 database

Method	CIFAR-10 (MAP)			
	12-bit	24-bit	32-bit	48-bit
LSH	0.163	0.171	0.179	0.190
SH	0.171	0.174	0.172	0.166
ITQ	0.195	0.212	0.220	0.227
SDH	0.344	0.601	0.605	0.620
KSDH	0.552	0.584	0.583	0.604
COSDISH	0.571	0.603	0.604	0.614
ADGH	0.587	0.620	0.629	0.634
SDH+CNN	0.503	0.836	0.847	0.846
KSDH+CNN	0.834	0.850	0.842	0.849
COSDISH+CNN	0.835	0.842	0.850	0.847
ADGH+CNN	0.745	0.775	0.779	0.779
NINH	0.800	0.833	0.858	0.843
DSH	0.871	0.891	0.890	0.883
DPSH	0.845	0.861	0.866	0.878
DSCH	0.830	0.848	0.834	0.842
ADSH	0.889	0.869	0.876	0.887
SSDPH*	0.895	0.897	0.896	0.895
SSDH	0.862	0.868	0.881	0.885
SSDPH	**0.923**	**0.925**	**0.924**	**0.923**

The bold and underline values denote the best and second best results of each setting, respectively

Table 2 Similarity retrieval performance (MAP) by using the labeled set for retrieval on top 5,000 retrieved samples of different hashing methods on the NUS-WIDE database

Method	NUS-WIDE (MAP)			
	12-bit	24-bit	32-bit	48-bit
LSH	0.404	0.413	0.436	0.445
SH	0.435	0.437	0.436	0.427
ITQ	0.473	0.480	0.485	0.488
SDH	0.609	0.612	0.610	0.623
KSDH	0.594	0.597	0.599	0.606
COSDISH	0.571	0.596	0.567	0.617
SDH+CNN	0.709	0.729	0.727	0.720
KSDH+CNN	0.684	0.692	0.684	0.692
COSDISH+CNN	0.717	0.722	0.708	0.755
NINH	0.808	0.818	0.827	0.824
DSH	0.747	0.757	0.759	0.755
DPSH	0.802	0.820	0.828	0.834
DSCH	0.597	0.683	0.688	0.710
ADSH	0.767	0.787	0.736	0.724
SSDPH*	0.828	0.860	0.861	0.870
SSDH	0.819	0.824	0.829	0.823
SSDPH	**0.838**	**0.877**	**0.879**	**0.888**

The bold and underline values denote the best and second best results of each setting, respectively

Table 3 Similarity retrieval performance (MAP) by using the labeled set for retrieval on top 5,000 retrieved samples of different hashing methods on the COCO database

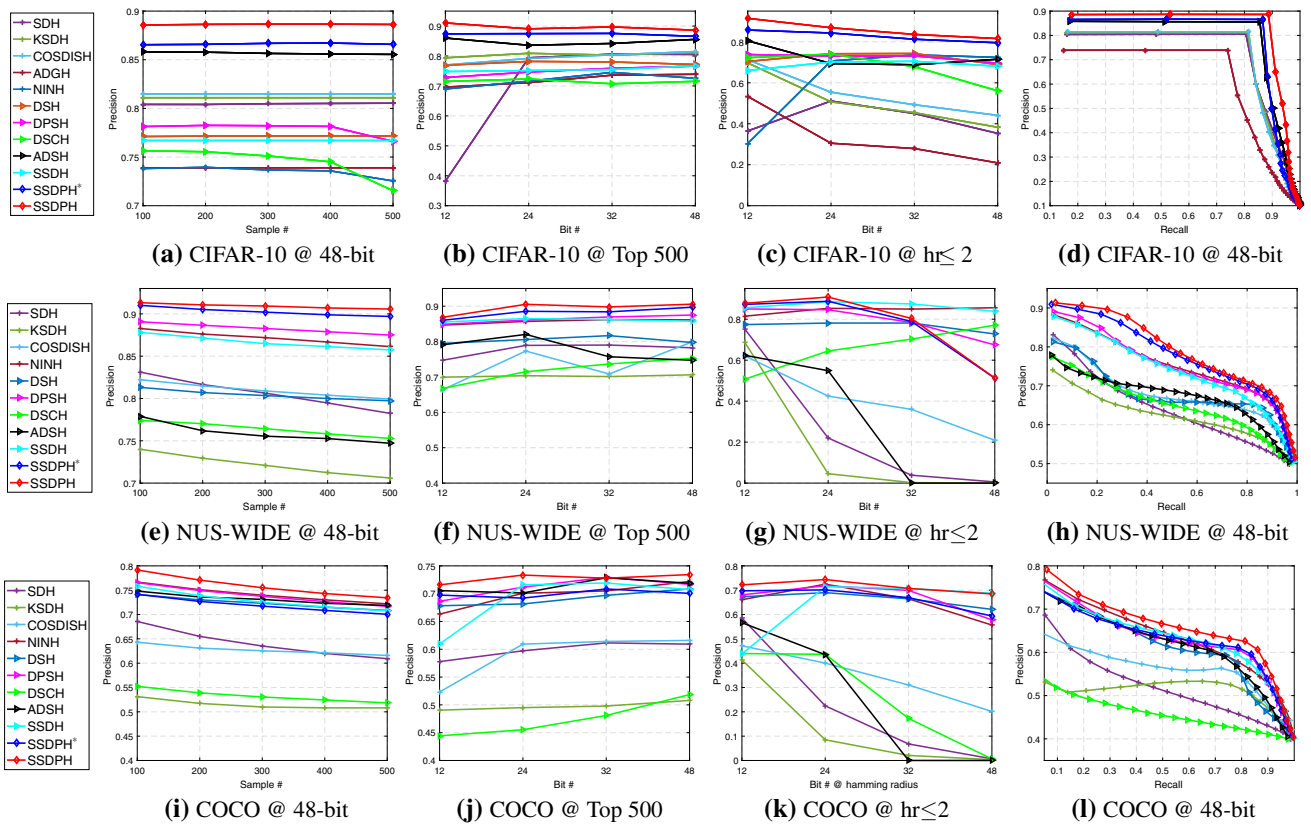
Method	COCO (MAP)			
	12-bit	24-bit	32-bit	48-bit
SDH+CNN	0.565	0.577	0.584	0.586
KSDH+CNN	0.553	0.578	0.585	0.588
COSDISH+CNN	0.589	0.606	0.608	0.626
NINH	0.626	0.667	0.667	0.688
DSH	0.665	0.666	0.686	0.690
DPSH	0.665	0.686	0.690	0.691
DSCH	0.436	0.457	0.437	0.436
ADSH	0.684	0.682	0.686	0.678
SSDPH*	0.672	0.674	0.694	0.696
SSDH	0.628	0.706	0.697	0.692
SSDPH	**0.691**	**0.715**	**0.716**	**0.727**

The bold and underline values denote the best and second best results of each setting, respectively

COCO database, we randomly select 1600 images as a query set, where each category consists of at least 20 images. We utilize the remaining 121,687 images as a base set, from which we randomly choose at least 100 images per category as labeled ones (totally 8000 images) and exploit the rest of images as unlabeled ones. Unsupervised algorithms LSH, SH, ITQ utilize the base set for training and supervised algorithms only adopt the labeled images for training, while semi-supervised algorithms exploit both labeled and unlabeled data for training. We show two cases of all algorithms for retrieval: (i) using the labeled set only as a retrieval database; (ii) utilizing the base set as a retrieval database. In addition to using AlexNet as the backbone network, we also show the retrieval performance of deep hashing algorithms using ResNet18 as their backbone networks on the three databases, with the labeled set used for retrieval. We repeat all experiments five times and report average results.

4.2.1 Experimental Results And Analysis

Tables 1, 2 and 3 show the MAPs of various algorithms at 12-, 24-, 32- and 48-bit on CIFAR-10, NUS-WIDE and COCO databases, respectively, with using the labeled set for retrieval. SSDPH* attains higher MAPs than the unsupervised and supervised algorithms on all the three databases except on COCO at 12- and 24-bit. SSDPH obtains the best performance among all algorithms on the three databases. Specifically, except SSDPH*, the gain of SSDPH in term of MAP ranges from 3.8 to 4.1% over the best competitors at the four settings on the CIFAR-10 database; On the NUS-WIDE dataset, the gain of SSDPH in MAP ranges from 2.3 to 6.4% over the best competitors; The gain of SSDPH is from 1.3



(a) CIFAR-10 @ 48-bit **(b)** CIFAR-10 @ Top 500 **(c)** CIFAR-10 @ hr$\leq$ 2 **(d)** CIFAR-10 @ 48-bit

(e) NUS-WIDE @ 48-bit **(f)** NUS-WIDE @ Top 500 **(g)** NUS-WIDE @ hr$\leq$2 **(h)** NUS-WIDE @ 48-bit

(i) COCO @ 48-bit **(j)** COCO @ Top 500 **(k)** COCO @ hr$\leq$2 **(l)** COCO @ 48-bit

Fig. 4 Similarity retrieval performance of different algorithms using the labeled set for retrieval on CIFAR-10, NUS-WIDE and COCO databases. 'hr' denotes the Hamming radius

to 5.1% over the best competitors on the COCO database. Compared to SSDPH*, SSDPH consistently exhibits superior performance on all the three databases at the four settings, because it can effectively explore the similarity information of unlabeled data. Additionally, Tables 1-3 demonstrate that non-deep supervised hashing algorithms with CNN features can attain better MAPs than that with hand-craft features, while they are inferior to most of the deep hashing algorithms due to learning feature representations and binary vectors separately. The point-wise based algorithm DSCH obtains worse performance than the other supervised hashing algorithms on the three databases, especially on multi-label databases NUS-WIDE and COCO, probably because it fails to consider the similarity relationship of training data. The multi-wise based hashing algorithm NINH attains worse performance than pairwise based supervised hashing algorithms on the single-label database CIFAR-10. The reason might be that the triplets emphasize the similarity difference over items, while pairwise labels can directly represent the similarity relationship of training data.

Fig. 4 displays the precision with different numbers of samples and bits, Hamming radius (hr) within 2, and PR curves of twelve hashing algorithms, including SDH, KSDH, COSDISH and ADGH with CNN features, NINH, DSH, DPSH, DSCH, ADSH, SSDH, SSDPH* and SSDPH, on the CIFAR-10 database, and eleven algorithms on NUS-WIDE and COCO databases except ADGH. It illustrates that SSDPH* can obtain superior precision and PR curves over the others except SSDPH on CIFAR-10 and NUS-WIDE databases. Moreover, SSDPH can achieve the best precision and PR curves among all hash algorithms on the three databases.

Time comparison: We run seven deep hashing algorithms using labeled training sets of the three databases on a machine with one Nvidia GeForce GTX 1080 Ti, and then calculate their average time on each training epoch. Table 4 presents training time costs of the seven supervised deep hashing algorithms, including two multi-wise hashing algorithms NINH and SSDH*, one point-wise based hashing algorithm DSCH, and four pairwise based hashing algorithms DSH, DPSH, ADSH and SSDPH*, on CIFAR-10, NUS-WIDE and COCO databases at 48-bit. SSDH* is the supervised case of SSDH. As we can see, a point-wise based hashing algorithm DSCH and an anchor-based pairwise based hashing method ADSH consumes less time than the others, while multi-wise based hashing algorithms NINH and SSDH* spend more time than

Table 4 Training time (seconds) of seven supervised deep hashing algorithms using the backbone network, AlexNet, on CIFAR-10, NUS-WIDE and COCO databases at 48-bit. Note that training time means the average cost of hashing algorithms on one epoch with batch size being 100

Method	CIFAR-10	NUS-WIDE	COCO
NINH	25.27	49.78	40.29
DSH	11.90	25.51	20.85
DPSH	13.47	27.11	24.67
DSCH	9.26	22.63	18.84
ADSH	9.41	17.46	15.53
SSDH*	15.22	28.48	30.38
SSDPH*	11.55	24.66	22.14

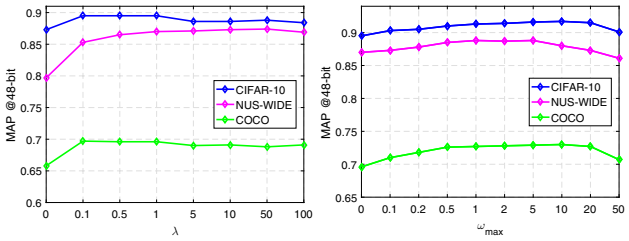
the others, especially NINH. This might be because more triplets are generated than pairwise labels so as to better leverage semantic information, and the generation of triplets consumes more time than that for pairwise labels. In our experiments, we sample three triplets for each data in one mini-batch to reduce training time costs, but each sample will theoretically generate $O(n^2)$ triplets, because there are existing n labeled samples. SSDH* takes less time than NINH, because SSDH* alternatively learns hash codes using triplets and trains the network for classification in a point-wise manner. Although SSDPH* consumes more time than DSCH and ADSH, it spends similar and even less training time than DSH and DPSH in most cases. Hence, its time cost is acceptably low. Note that we do not show their test time, because these are no triplets or pairwise labels generated in the test stage and they spend almost the same test time. For the convergence speed of the seven deep hashing algorithms, it is significantly affected by many factors, such as the number of training data, the data type and the number of bits, so that it is difficult to be objectively evaluated. But most of them usually converge and attain the best performance within 100 or 50 epochs.

Ablation study and parameter analysis: Table 5 shows the effect of the three terms J_1, J_2 and J_3 in Eq. (7) on CIFAR-10. It suggests that the second term J_2 in Eq. (7) can boost the model performance, because it can further exploit the semantic information of labeled training data. The third term J_3 will very slightly decrease the model performance when only using labeled training data, probably because the created targets do not increase semantic information and even might contain noisy information. However, the term J_3 can significantly boost the model performance when using both labeled and unlabeled training data, because it can explore the semantic information of unlabeled data. Similar findings can be observed on NUS-WIDE and COCO databases. Fig. 5 shows the influence of λ in SSDPH* and ω_{max} in SSDPH on CIFAR-10, NUS-WIDE and COCO databases, because λ affects the usage of semantic information and ω_{max} determines the usage of similarity information in unlabeled data.

Table 5 Ablation study of the proposed loss function Eq. (7) on CIFAR-10 with using the labeled set for retrieval

SSDPH*	Term	MAP			
		12-bit	24-bit	32-bit	48-bit
	J_1	0.880	0.887	0.891	0.892
	$J_1 + J_2$	**0.898**	**0.899**	**0.896**	**0.897**
	$J_1 + J_2 + J_3$	0.895	0.897	**0.896**	0.895
SSDPH	$J_1 + J_3$	0.912	0.914	0.913	0.911
	$J_2 + J_3$	0.910	0.916	0.919	0.911
	$J_1 + J_2 + J_3$	**0.923**	**0.925**	**0.924**	**0.923**

The bold and underline values denote the best and second best results of each setting, respectively



(a) @SSDPH* **(b)** @SSDPH

Fig. 5 The effects of λin SSDPH* and ω_{max} in SSDPH on CIFAR-10, NUS-WIDE and COCO databases

Fig. 5a shows that when $\lambda \in [0.1, 1]$, SSDPH* obtains the sub-best or best MAP on CIFAR-10 and COCO databases; when $\lambda \in [1, 50]$, SSDPH* attains the sub-best or best MAP on NUS-WIDE. Thus we empirically set $\lambda = 1$ on the three databases. Fig. 5b suggests that when $\omega_{max} \in [1, 20]$, SSDPH obtains the sub-best or best MAP on CIFAR-10; When $\omega_{max} \in [0.5, 5]$, SSDPH obtains the sub-best or best performance on NUS-WIDE; When $\omega_{max} \in [0.5, 20]$, SSDPH obtains the sub-best or best MAP on COCO. For the parameter n^a, when $n^a \geq 2C$, SSDPH* and SSDPH empirically achieve the best or sub-best MAP on all the three databases. We show the detailed parameter setting in the Appendix.

Table 6 presents the MAPs of eleven deep hashing algorithms using the base set for retrieval. Among supervised deep hashing algorithms, SSDPH* achieves better MAPs on CIFAR-10 and NUS-WIDE databases at 12-, 24-, 32- and 48-bit than the others, and it can obtain comparable performance to the best competitors on the COCO database. SSDPH achieves the best performance among all supervised and semi-supervised hashing methods, and it consistently outperforms SSDPH* on all the three databases, because SSDPH can effectively explore the semantic similarity information hidden in unlabeled data. It is worth noting that SSGAH attains similar MAPs to SSDPH on CIFAR-10 at 24- and 32-bit. It adopts three networks including generative, discriminative and hashing models, but SSDPH employs only one single network.

Table 6 Similarity retrieval performance (MAP) of eleven deep hashing methods using the base set for retrieval on CIFAR-10, NUS-WIDE and COCO databases. Note that the MAP on CIFAR-10 is calculated with all returned samples, and the MAP on NUS-WIDE and COCO is computed with top 5000 retrieved samples

Method	CIFAR-10 (MAP)			
	12-bit	24-bit	32-bit	48-bit
NINH	0.723	0.760	0.771	0.767
DSH	0.750	0.769	0.781	0.796
DPSH	0.759	0.775	0.779	0.798
DSCH	0.781	0.786	0.788	0.794
ADSH	0.763	0.786	0.799	0.806
SSDPH*	0.785	0.812	0.816	0.822
SSDH	0.769	0.798	0.799	0.814
BGDH†	0.805	0.824	0.826	0.833
DSH-GANs†	0.735	0.781	0.787	0.802
SSGAH†	0.819	0.837	0.847	0.855
SSDPH	**0.829**	**0.839**	**0.852**	**0.865**
	NUS-WIDE (MAP)			
NIH	0.774	0.785	0.802	0.812
DSH	0.711	0.731	0.730	0.741
DPSH	0.792	0.815	0.824	0.831
DSCH	0.598	0.663	0.696	0.771
ADSH	0.717	0.710	0.755	0.753
SSDPH*	0.799	0.822	0.830	0.839
SSDH	0.769	0.780	0.796	0.795
SSDPH	**0.808**	**0.834**	**0.842**	**0.848**
	COCO (MAP)			
NIH	0.642	0.664	0.670	0.683
DSH	0.613	0.652	0.658	0.656
DPSH	0.648	0.677	0.682	0.707
DSCH	0.404	0.395	0.422	0.415
ADSH	0.590	0.604	0.601	0.613
SSDPH*	0.638	0.671	0.687	0.704
SSDH	0.585	0.681	0.695	0.676
SSDPH	**0.662**	**0.687**	**0.698**	**0.715**

The bold and underline values denote the best and second best results of each setting, respectively

† denotes the results directly copied from the original papers

To further evaluate the proposed method, we utilize a recent popular network ResNet18 (He et al. 2016) as the backbone of the eight deep hashing algorithms. Table 7 presents their MAPs under the same experimental setting as Tables 1, 2 and 3. As we can see, SSDPH* can achieve the best MAPs among all supervised hashing algorithms on the three databases except at 32-bit on the CIFAR-10 database. SSDPH obtains the superior performance over the other supervised and semi-supervised hashing algorithms at the four settings on all the three datasets. It suggests that SSDPH is general and effective, and it can outperform other deep hashing algorithms on multiple backbone networks. The detailed parameter setting of SSDPH using ResNet18 is also listed in the Appendix.

4.3 Experiments for Ranking Order

We run experiments on multi-label databases, NUS-WIDE and COCO, to evaluate the performance of SSDPH* and SSDPH using Eq. (11) on maintaining the relevance between query and training samples, and compare them against non-deep hashing methods, including non-ranking algorithms: LSH, SH, ITQ, SDH, KSDH and COSDISH, and ranking based ones: CGH and DSeRH_L, and deep hashing methods including ranking based algorithms: NINH, DSRH and SSDH, and one point-wise based algorithm DSCH. We do not show the performance of pairwise based hashing algorithms DSH, DPSH and ADSH, because they only preserve the similarity relationship among samples without considering their relevance. Additionally, we do not present results of BGDH, DSH-GANs and SSGAH, because there are no reported results and public available codes. However, we show results of three more ranking based algorithms CGH, DSeRH_L and DSRH for a better comparison. Using the same protocols as that in Sect. 4.2, we construct different sets for the above unsupervised, supervised and semi-supervised algorithms. We evaluate their performance by calculating the values of NDCG, and utilize the labeled set as a retrieval database. All experiments are repeated five times and the average NDCG is reported.

Table 8 presents the NDCGs of aforementioned hashing algorithms with top 100 retrieved samples on NUS-WIDE and COCO databases. As we can see, SSDPH obtains the higher NDCGs than the other hash algorithms on the two databases. Its gain in term of NDCG ranges from 5.4 to 27.1% over the best competitors except SSDPH* on the NUS-WIDE database, and the gain is from 1.3 to 7.2% on the COCO database. SSDPH* attains higher NDCGs than the other supervised deep hashing approaches except NINH at 12- and 24-bit on the COCO database. Additionally, SSDPH consistently outperforms SSDPH* because of effectively exploring the similarity information hidden in unlabeled data.

To further evaluate the performance of deep hashing algorithms, Fig. 6 displays the NDCGs of NINH, DSRH, DSCH, SSDH, SSDPH* and SSDPH on NUS-WIDE and COCO databases, with the number of returned samples from 10 to 200 at 24- and 48-bit. Obviously, SSDPH outperforms the other deep hashing algorithms on the two databases. Therefore, SSDPH can successfully maintain the relevance between query and training samples through anchors, and effectively exploit the unlabeled data. Fig. 7 shows the effects of different numbers of anchors on ranking retrieval performance of SSDPH*. It suggests that SSDPH* can achieve sub-best or best NDCGs when $n^a >= 126$ and $n^a >= 400$ on NUS-WIDE and COCO databases, respectively. Simi-

Table 7 Similarity retrieval performance (MAP) by using ResNet18 as the backbone network and the labeled set as a retrieval database on top 5000 retrieved samples of eight deep hashing methods on CIFAR-10, NUS-WIDE and COCO databases

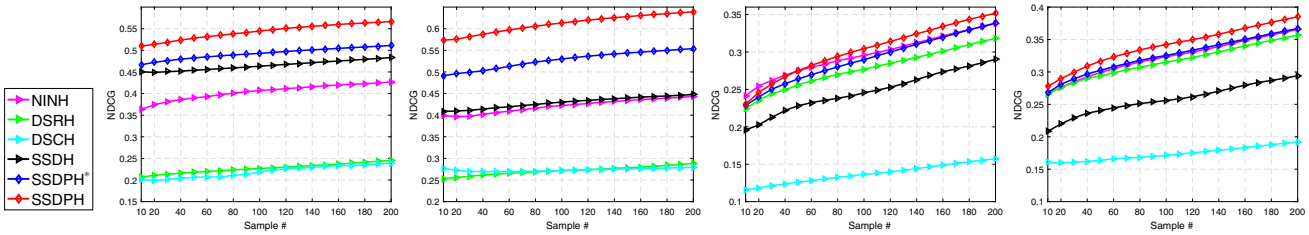
Method	CIFAR-10 (MAP)				NUS-WIDE (MAP)				COCO (MAP)			
	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	48-bit
NINH	0.896	0.891	0.884	0.902	0.818	0.823	0.826	0.815	0.644	0.699	0.705	0.723
DSH	0.904	0.900	0.902	0.902	0.790	0.795	0.787	0.786	0.698	0.706	0.715	0.722
DPSH	0.925	0.904	0.938	0.933	0.777	0.792	0.803	0.815	0.636	0.658	0.670	0.666
DSCH	0.916	0.933	0.932	0.935	0.582	0.610	0.653	0.668	0.472	0.460	0.464	0.456
ADSH	0.901	0.925	0.917	0.929	0.822	0.856	0.865	0.877	0.698	0.707	0.719	0.732
SSDPH*	0.935	0.936	0.933	0.940	0.837	0.863	0.870	0.880	0.698	0.732	0.726	0.732
SSDH	0.914	0.928	0.923	0.932	0.812	0.833	0.826	0.827	0.655	0.664	0.675	0.692
SSDPH	**0.957**	**0.962**	**0.962**	**0.964**	**0.850**	**0.876**	**0.886**	**0.900**	**0.708**	**0.740**	**0.750**	**0.756**

The bold and underline values denote the best and second best results of each setting, respectively

Table 8 Ranking performance (NDCG) on top 100 retrieved samples of different hashing methods using the labeled set for retrieval on NUS-WIDE and COCO databases

Method	NUS-WIDE				COCO			
	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	48-bit
LSH+CNN	0.196	0.266	0.296	0.330	0.145	0.173	0.180	0.200
SH+CNN	0.328	0.345	0.355	0.363	0.190	0.207	0.215	0.219
ITQ+CNN	0.363	0.411	0.425	0.438	0.206	0.236	0.243	0.256
SDH+CNN	0.376	0.456	0.470	0.469	0.193	0.249	0.279	0.305
KSDH+CNN	0.258	0.246	0.239	0.229	0.111	0.153	0.159	0.169
COSDISH+CNN	0.163	0.320	0.161	0.253	0.175	0.208	0.185	0.259
CGH+CNN	0.307	0.375	0.413	0.430	0.171	0.226	0.235	0.244
DSeRH_L+CNN	0.323	0.336	0.341	0.351	0.199	0.211	0.224	0.230
NINH	0.402	0.432	0.439	0.462	0.246	0.300	0.307	0.324
DSRH	0.215	0.229	0.264	0.272	0.215	0.278	0.290	0.317
DSCH	0.196	0.242	0.234	0.261	0.113	0.136	0.146	0.172
SSDPH*	0.416	0.494	0.504	0.511	0.235	0.292	0.313	0.325
SSDH	0.448	0.478	0.462	0.445	0.230	0.255	0.265	0.262
SSDPH	**0.472**	**0.544**	**0.564**	**0.596**	**0.254**	**0.304**	**0.329**	**0.345**

The bold and underline values denote the best and second best results of each setting, respectively



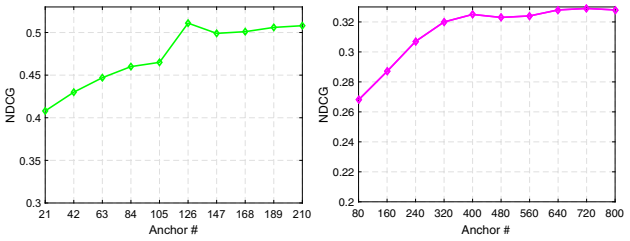
(a) NUS-WIDE @ 24-bit **(b)** NUS-WIDE @ 48-bit **(c)** COCO @ 24-bit **(d)** COCO @ 48-bit

Fig. 6 Ranking retrieval performance (NDCG) of different deep hashing algorithms on NUS-WIDE and COCO databases

lar observations can be found in SSDPH. Therefore, we set $n^a = 6C = 126$ and $n^a = 5C = 400$ for both SSDPH* and SSDPH on NUS-WIDE and COCO databases, respectively.

4.4 Experiments for Unseen Category Retrieval

We compare SSDPH and SSDPH* using Eq. (8) against seven non-deep hashing algorithms LSH, SH, ITQ, SDH, KSDH, COSDISH and ADGH, and six deep hashing algorithms NINH, DSH, DPSH, DSCH, ADSH and SSDH on unseen class retrieval. Following the protocol in (Sablayrolles et al. 2017), 75% known classes are adopted for training, and the remaining 25% classes are utilized as unknown ones for evaluation. Specifically, one dataset is divided into four sets: $train75$, $test75$, $train25$ and $test25$. The $set75$ ($train75 + test75$) is composed of the data of



(a) NUS-WIDE @ 48-bit **(b)** COCO @ 48-bit

Fig. 7 Ranking retrieval performance (NDCG) of SSDPH* with different numbers of anchors on NUS-WIDE and COCO databases

75% classes, and the $set25$ ($train25 + test25$) consists of the data belonging to the remaining 25% classes.

We run experiments on CIFAR-10, MNIST and NUS-WIDE databases. Following the experimental setup in (Zhang and Peng 2017), we randomly select 9,000 images from the CIFAR-10 database to construct a test set, with each class containing 900 images. The remaining ones are used as a base set, from which we randomly choose 21,000 images to construct a training labeled set, where each class consists of 2,100 images. The rest of 30,000 images are used as unlabeled ones. For the MNIST database, we randomly choose 10,910 images as a test set, select 24,080 images to construct a training labeled set, and utilize the rest of images as unlabeled ones. For the NUS-WIDE database, we only select the images containing one concept, totally we obtain 89,862 images. Because it is difficult to select images from multi-label databases such that $set75$ and $set25$ do not contain any common categories. Then we randomly select 8,400 images to construct a test set and adopt the rest of images as a base set, from which we randomly choose 21,000 images to build a training labeled set and utilize the remaining ones as unlabeled images. After that, we divide the test and training labeled sets of all the three databases into $train75$, $test75$, $train25$ and $test25$. The $set75$ of CIFAR-10, MNIST and NUS-WIDE contains 7, 7, 15 classes, respectively. We repeat above process five times and report the average MAPs of aforementioned algorithms. Non-deep hashing algorithms utilize CNN features extracted from the $fc7$ layer of the AlexNet pre-trained on $train75$ in a manner of classification. Supervised algorithms only adopt $train75$ for training; Unsupervised hashing algorithms LSH, SH, ITQ and semi-supervised algorithms exploit $train75$ and unlabeled ones in the base set for training. Additionally, we utilize $test25$ as a query set and $train25+test75$ as a database set for retrieval. MAP scores are calculated based on all retrieved images.

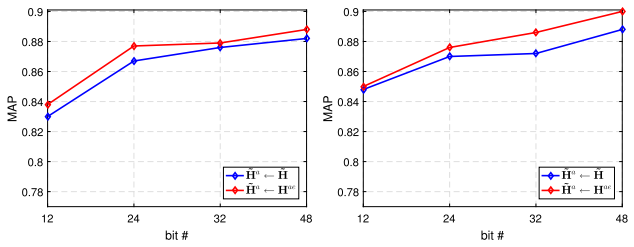
Table 9 presents the MAPs of fifteen algorithms on CIFAR-10, MNIST and NUS-WIDE databases. The unsupervised algorithm ITQ can obtains better MAPs than the others except SSDPH on CIFAR-10 and MNIST databases, probably because it can effectively capture the intrinsic structure of images of these two databases to represent their true distribution. SSDPH can significantly outperform the other algorithms on the three databases. Specifically, the gain of SSDPH in MAP is from 7.2 to 9.1% over the best competitors on the CIFAR-10 database, its gain ranges from 3.2 to 19.4% over the best competitors on the MNIST database, and its gain

Table 9 Similarity retrieval performance (MAP) with all returned samples of different hashing methods on CIFAR-10, MNIST and NUS-WIDE databases for unseen category retrieval

Method	CIFAR-10				MNIST				NUS-WIDE			
	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	48-bit	12-bit	24-bit	32-bit	48-bit
LSH+CNN	0.290	0.307	0.320	0.345	0.469	0.537	0.534	0.586	0.172	0.185	0.191	0.202
SH+CNN	0.457	0.409	0.404	0.404	0.653	0.598	0.601	0.587	0.226	0.222	0.223	0.226
ITQ+CNN	0.506	0.512	0.524	0.526	0.720	0.748	0.752	0.754	0.256	0.266	0.272	0.281
SDH+CNN	0.455	0.370	0.379	0.388	0.624	0.629	0.619	0.622	0.320	0.334	0.337	0.343
KSDH+CNN	0.430	0.407	0.406	0.405	0.624	0.568	0.606	0.604	0.313	0.315	0.330	0.339
COSDISH+CNN	0.462	0.435	0.443	0.428	0.602	0.603	0.592	0.582	0.315	0.330	0.334	0.336
ADGH+CNN	0.433	0.408	0.397	0.387	0.617	0.600	0.595	0.596	0.312	0.326	0.333	0.339
NINH	0.429	0.419	0.407	0.398	0.543	0.612	0.547	0.590	0.387	0.391	0.326	0.382
DSH	0.421	0.410	0.433	0.431	0.554	0.511	0.499	0.510	0.162	0.194	0.204	0.228
DPSH	0.421	0.409	0.414	0.408	0.529	0.553	0.477	0.517	0.346	0.381	0.348	0.413
DSCH	0.457	0.446	0.451	0.454	0.588	0.562	0.553	0.553	0.377	0.330	0.431	0.369
ADSH	0.398	0.391	0.370	0.390	0.472	0.581	0.475	0.562	0.331	0.318	0.404	0.330
SSDPH*	0.397	0.403	0.402	0.428	0.541	0.503	0.522	0.590	0.320	0.385	0.312	0.344
SSDH	0.399	0.406	0.409	0.408	0.538	0.560	0.677	0.715	0.388	0.368	0.366	0.437
SSDPH	**0.548**	**0.549**	**0.563**	**0.574**	**0.860**	**0.800**	**0.776**	**0.795**	**0.440**	**0.428**	**0.435**	**0.459**

The bold and underline values denote the best and second best results of each setting, respectively

Fig. 8 Several retrieved unseen examples of seven deep hashing algorithms using 48-bit binary codes on the CIFAR-10 database. The red rectangles denote the wrong retrieved images



(a) @ AlexNet **(b)** @ ResNet18

Fig. 9 The effect of anchor targets $\tilde{\mathbf{H}}^a$ in SSDPH by using the labeled set for similarity retrieval on NUS-WIDE at 12-, 24-, 32- and 48-bit. $\tilde{\mathbf{H}}^a \leftarrow \tilde{\mathbf{H}}$ means obtaining $\tilde{\mathbf{H}}^a$ from $\tilde{\mathbf{H}}$ every epoch, and $\tilde{\mathbf{H}}^a \leftarrow \mathbf{H}^{ae}$ means attaining $\tilde{\mathbf{H}}^a$ every iteration

is from 0.9 to 13.4% over the best competitors on the NUS-WIDE database. One main possible reason for the superior performance of SSDPH is that anchor-based self-ensembling can effectively preserve the similarities of unlabeled data and meanwhile better represent the true distribution of all data. To better understand the unseen category retrieval, we display several retrieved unseen samples of SSDPH, SSDH, ADSH, DSCH, DPSH, DSH and NINH in Fig. 8, which further illustrates the strength of SSDPH on unseen category retrieval.

4.5 Discussion

Experiments on the four benchmark databases and three retrieval tasks illustrate that: (i) On the task of similarity retrieval, SSDPH* achieves better MAPs than the other supervised deep hashing algorithms in most of the cases. This might be attributed to the algorithm design of exploiting the pairwise similarity between training data and anchors and utilizing the pairwise similarity information of selected images in each min-batch; (ii) SSDPH obtains better performance than semi-supervised algorithms SSDH, BGDH, DSH-GANs and SSGAH on CIFAR-10 for similarity retrieval, probably because of (1) better usage of the semantic information of labeled data and (2) effective exploitation of the similarity information hidden in unlabeled data by using self-ensembling; (iii) SSDPH and SSDPH* can achieve the sub-best or best ranking performance compared to the other algorithms. One main possible reason is that they can better leverage the semantic similarity information of labeled data and preserve the relevance among samples through calculating their similarities to anchors; (iv) SSDPH consistently obtains superior performance over the SSDPH* on the four benchmark databases and three retrieval tasks. This suggests that anchor-based self-ensembling can effectively explore the semantic similarity information of unlabeled data.

The self-ensembling method TE (Laine and Aila 2016) updates the targets once per epoch, and it might decrease its performance for learning large datasets (Tarvainen and Valpola 2017). Our method can alleviate this issue by using anchors, because it updates the targets of anchors once per iteration. Additionally, our method also significantly outperforms its supervised case on the backbone network ResNet18 without using one regularization technique, dropout, which is very significant in many self-ensembling based methods (Laine and Aila 2016; Tarvainen and Valpola 2017). It might suggest that our method is not largely affected by the dropout. This is possibly attributed to the changed targets of anchors in each iteration. For better illustration, Fig. 9 presents the effect of anchor targets $\tilde{\mathbf{H}}^a$ in SSDPH on NUS-WIDE using the backbone networks AlexNet and ResNet18. Note that $\tilde{\mathbf{H}}^a \leftarrow \tilde{\mathbf{H}}$ means we remove steps 8-9 in SSDPH and obtain $\tilde{\mathbf{H}}^a$ from $\tilde{\mathbf{H}}$ based on the index of anchors every epoch; $\tilde{\mathbf{H}}^a \leftarrow \mathbf{H}^{ae}$ means we obtain $\tilde{\mathbf{H}}^a$ based on the steps 8–9 in SSDPH every iteration. As shown in Fig. 9, the update of anchor targets in each iteration usually performs better than updating that every epoch. Moreover, previous supervised hashing algorithms require the relevance of training data to preserve their ranking orders. By contrast, our method does not need their relevance and only require similarity relations of training data to maintain their ranking orders. It means that our method can be used to obtain the relevance of samples when only having their similarity relations.

5 Conclusion and Future Work

In this paper, we propose a novel semi-supervised deep pairwise hashing framework using anchor-based self-ensembling. The proposed method leverages the semantic information via the pairwise similarity between labeled samples and anchors, and explores the similarity information hidden in unlabeled data by applying the EMA to latent binary vectors within multiple previous training epochs so as to create a strong target for unknown latent binary vectors, and forming a con-

sensus similarity relationship prediction to multiple anchors. Additionally, we explain and present how to encode the query and training data through anchors for preserving their relevance. Extensive experiments on similarity, ranking order and unseen category retrieval demonstrate the effectiveness and efficiency of the proposed method, which can achieve superior retrieval performance over recent state-of-the-art hashing methods.

Although the proposed method achieves promising performance on the three tasks, it can still be improved at least two major aspects: (i) Many previous discrete hashing methods have demonstrated that directly learning binary codes can improve the retrieval performance (Shen et al. 2017; Jiang and Li 2018). Our method does not adopt this strategy because we empirically find that directly learning binary codes might hurt the model smoothness, which is significant to explore the semantic information in unlabeled data. Therefore, it is important to further study whether learning discrete binary codes directly is better for semi-supervised deep hashing;(ii) The proposed method only utilizes a single view of data and focuses on single-view retrieval tasks. Hence, it is worth boosting and extending our method to exploit multiple views of data and tackle multi-view tasks. Moreover, we only evaluate our method on images, it can also be extended to multimedia.

Acknowledgements The work is partially supported by the Natural Science Foundation of China (NSFC) (No. 61772296), Shenzhen fundamental research fund (No. JCYJ20170412170438636).

References

Broder, A. Z., Charikar, M., Frieze, A. M., & Mitzenmacher, M. (2000). Min-wise independent permutations. *Journal of Computer and System Sciences*, *60*(3), 630–659.

Cao, Y., Long, M., Liu, B., & Wang, J. (2018). Deep cauchy hashing for hamming space retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition(CVPR)*, (pp. 229–1237).

Cao, Z., Long, M., Wang, J., & Yu, P.S. (2017). Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision (ICCV)*.

Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., & Zheng, Y. (2009). Nus-wide: A real-world web image database from national university of singapore. In *Proceedings of the ACM international conference on image and video retrieval (ACM-CIVR)*, ACM, (p. 48).

Datar, M., Immorlica, N., Indyk, P., & Mirrokni, V.S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on computational geometry (SoCG), ACM*, (pp. 253–262).

Datta, R., Joshi, D., Li, J., & Wang, J. Z. (2008). Image retrieval: Ideas, influences, and trends of the new age. *ACM Computing Surveys*, *40*(2), 5.

Delalleau, O., Bengio, Y., & Le Roux, N. (2005). Efficient non-parametric function induction in semi-supervised learning. In *AISTATS*, vol 27, (p. 100).

Gong, Y., Lazebnik, S., Gordo, A., & Perronnin, F. (2013). Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(12), 2916–2929.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, (pp 770–778).

Hu, H., Wang, K., Lv, C., Wu, J., & Yang, Z. (2019). Semi-supervised metric learning-based anchor graph hashing for large-scale image retrieval. *IEEE Transactions on Image Processing*, *28*(2), 739–754.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on theory of computing (STOC), ACM*, (pp 604–613).

Järvelin, K., & Kekäläinen, J. (2000). Ir evaluation methods for retrieving highly relevant documents. In *Proceedings of international ACM SIGIR conference on research and development in information retrieval (SIGIR), ACM*, (pp 41–48).

Jiang, Q.Y., & Li, W.J. (2018). Asymmetric deep supervised hashing. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*.

Kang, W.C., Li, W.J., & Zhou, Z.H. (2016). Column sampling based discrete supervised hashing. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, (pp 1230–1236).

Kingma, D.P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Krizhevsky, A., & Hinton, G. (2009). *Learning multiple layers of features from tiny images*. Technical report, Citeseer.

Krizhevsky, A., Sutskever, I., & Hinton, G.E. (2012). Imagenet classification with deep convolutional neural networks. In *Proceedings of the advances in neural information processing systems (NIPS)*, (pp. 1097–1105).

Kulis, B., & Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. In *Proceedings of the advances in neural information processing systems (NIPS)*, (pp. 1042–1050).

Kulis, B., Jain, P., & Grauman, K. (2009). Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(12), 2143–2157.

Lai, H., Pan, Y., Liu, Y., & Yan, S. (2015). Simultaneous feature learning and hash coding with deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, (pp. 3270–3278).

Laine, S., & Aila, T. (2016). Temporal ensembling for semi-supervised learning. arXiv preprint arXiv:1610.02242.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, W.J., Wang, S., & Kang, W.C. (2015). Feature learning based deep supervised hashing with pairwise labels. arXiv preprint arXiv:1511.03855.

Li, X., Lin, G., Shen, C., Hengel, Avd, & Dick, A. (2013). Learning hash functions using column generation. arXiv preprint arXiv:1303.0339.

Lin, K., Yang, H.F., Hsiao, J.H., & Chen, C.S. (2015). Deep learning of binary hash codes for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops (CVPR workshops)*, (pp. 27–35).

Lin TY, Maire M, Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C.L. (2014). Microsoft coco: Common objects in context. In *Proceedings of the European conference on computer vision (ECCV)*, Springer, (pp. 740–755).

Liu, H., Wang, R., Shan, S., & Chen, X. (2016). Deep supervised hashing for fast image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, (pp. 2064–2072).

Liu, L., Shao, L., Shen, F., & Yu, M. (2017). Discretely coding semantic rank orders for supervised image hashing. In *Proceedings of*

the IEEE conference on computer vision and pattern recognition (CVPR), (pp. 1425–1434).
Liu, W., He, J., & Chang, S.F. (2010). Large graph construction for scalable semi-supervised learning. In *Proceedings of the international conference on machine learning (ICML)*, (pp. 679–686).
Liu, W., Wang, J., Ji, R., Jiang, Y.G., & Chang, S.F. (2012). Supervised hashing with kernels. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, IEEE, (pp. 2074–2081).
Liu, W., Mu, C., Kumar, S., & Chang, S.F. (2014). Discrete graph hashing. In *Proceedings of the advances in neural information processing systems (NIPS)*, (pp. 3419–3427).
Maaten, Lvd, & Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, *9*, 2579–2605.
Miyato, T., Maeda, Si, Ishii, S., & Koyama, M. (2018). Virtual adversarial training: A regularization method for supervised and semi-supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(8), 1979–1993.
Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, *42*(3), 145–175.
Qiu, Z., Pan, Y., Yao, T., & Mei, T. (2017). Deep semantic hashing with generative adversarial networks. In *Proceedings of the international ACM SIGIR conference on research and development in information retrieval (SIGIR), ACM*, (pp. 225–234).
Raginsky, M., & Lazebnik, S. (2009). Locality-sensitive binary codes from shift-invariant kernels. In *Proceedings of the advances in neural information processing systems (NIPS)*, (pp. 1509–1517).
Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, *115*(3), 211–252.
Sablayrolles, A., Douze, M., Usunier, N., & Jégou, H. (2017). How should we evaluate supervised hashing? In *Proceedings of the IEEE conference on acoustics, speech and signal processing (ICASSP), IEEE*, (pp. 1732–1736).
Sapkota, M., Shi, X., Xing, F., & Yang, L. (2018). Deep convolutional hashing for low dimensional binary embedding of histopathological images. *IEEE Journal of Biomedical and Health Informatics*, *23*(2), 805–816.
Shen, F., Liu, W., Zhang, S., Yang, Y., & Tao, Shen H. (2015a) Learning binary codes for maximum inner product search. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, (pp. 4148–4156).
Shen, F., Shen, C., Liu, W., & Tao, Shen H. (2015b). Supervised discrete hashing. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, (pp. 37–45).
Shen, F., Gao, X., Liu, L., Yang, Y., & Shen, H.T. (2017). Deep asymmetric pairwise hashing. In *Proceedings of the ACM on multimedia conference (ACMMM), ACM*, (pp. 1522–1530).
Shi, X., Xing, F., Cai, J., Zhang, Z., Xie, Y., & Yang, L. (2016). Kernel-based supervised discrete hashing for image retrieval. In *Proceedings of the European conference on computer vision (ECCV)*, Springer, (pp. 419–433).
Shi, X., Xing, F., Xu, K., Sapkota, M., & Yang, L. (2017a). Asymmetric discrete graph hashing. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, (pp. 2541–2547).
Shi, X., Xing, F., Xu, K., Xie, Y., Su, H., & Yang, L. (2017b). Supervised graph hashing for histopathology image retrieval and classification. *Medical image analysis*, *42*, 117–128.
Shi, X., Sapkota, M., Xing, F., Liu, F., Cui, L., & Yang, L. (2018). Pairwise based deep ranking hashing for histopathology image classification and retrieval. *Pattern Recognition*, *81*, 14–22.
Song, J., Zhang, H., Li, X., Gao, L., Wang, M., & Hong, R. (2018). Self-supervised video hashing with hierarchical binary auto-encoder. *IEEE Transactions on Image Processing*, *27*(7), 3210–3221.
Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the advances in neural information processing systems (NIPS)*, (pp. 1195–1204).
Torralba, A., Fergus, R., & Freeman, W. T. (2008). 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(11), 1958–1970.
Wang, G., Hu, Q., Cheng, J., & Hou, Z. (2018). Semi-supervised generative adversarial hashing for image retrieval. In *Proceedings of the European conference on computer vision (ECCV)*, (pp. 469–485).
Wang, J., Kumar, S., & Chang, S. F. (2012). Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *12*, 2393–2406.
Wang, J., Zhang, T., Sebe, N., Shen, H. T., et al. (2017). A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 769–790.
Weiss, Y., Torralba, A., & Fergus, R. (2009). Spectral hashing. In *Proceedings of the advances in neural information processing systems (NIPS)*, (pp. 1753–1760).
Xia, R., Pan, Y., Lai, H., Liu, C., & Yan, S. (2014). Supervised hashing for image retrieval via image representation learning. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, vol 1, (p 2).
Xu, K., Liu, S., Zhao, P., Chen, P.Y., Zhang, H., Erdogmus, D., Yanzhi, W., & Xue, L. (2018). Structured adversarial attack: Towards general implementation and better interpretability. In *Proceedings of international Conference on Learning Representations (ICLR)*.
Xu, K., Chen, H., Liu, S., Chen, P.Y., Weng, T.W., Hong, M., & Xue, L. (2019). Topology attack and defense for graph neural networks: An optimization perspective. In *Proceedings of international joint conference on artificial intelligence (IJCAI)*.
Yan, X., Zhang, L., & Li, W.J. (2017). Semi-supervised deep hashing with a bipartite graph. In *Proceedings of the international joint conference on artificial intelligence (IJCAI)*, AAAI Press, (pp. 3238–3244).
Zhang, J., & Peng, Y. (2017). Ssdh: semi-supervised deep hashing for large scale image retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, *29*(1), 212–225.
Zhang, J., Peng, Y., & Yuan, M. (2018). Sch-gan: Semi-supervised cross-modal hashing by generative adversarial network. arXiv preprint arXiv:1802.02488.
Zhang, R., Lin, L., Zhang, R., Zuo, W., & Zhang, L. (2015). Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing*, *24*(12), 4766–4779.
Zhao, F., Huang, Y., Wang, L., & Tan, T. (2015). Deep semantic ranking based hashing for multi-label image retrieval. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, (pp. 1556–1564).
Zhu, H., Long, M., Wang, J., & Cao, Y. (2016). Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI conference on artificial intelligence (AAAI)*, (pp. 2415–2421).
Zhuang, B., Lin, G., Shen, C., & Reid, I. (2016). Fast training of triplet-based deep binary embedding networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, (pp. 5955–5964).