Deep Hashing via Discrepancy Minimization

Zhixiang Chen[a,b,c], Xin Yuan[a,b,c], Jiwen Lu[a,b,c,*], Qi Tian[d], Jie Zhou[a,b,c]
[a]Department of Automation, Tsinghua University, China
[b]State Key Lab of Intelligent Technologies and Systems, Tsinghua University, China
[c]Beijing National Research Center for Information Science and Technology, China
[d]Department of Computer Science, University of Texas at San Antonio, USA

Abstract

This paper presents a discrepancy minimizing model to address the discrete optimization problem in hashing learning. The discrete optimization introduced by binary constraint is an NP-hard mixed integer programming problem. It is usually addressed by relaxing the binary variables into continuous variables to adapt to the gradient based learning of hashing functions, especially the training of deep neural networks. To deal with the objective discrepancy caused by relaxation, we transform the original binary optimization into differentiable optimization problem over hash functions through series expansion. This transformation decouples the binary constraint and the similarity preserving hashing function optimization. The transformed objective is optimized in a tractable alternating optimization framework with gradual discrepancy minimization. Extensive experimental results on three benchmark datasets validate the efficacy of the proposed discrepancy minimizing hashing.

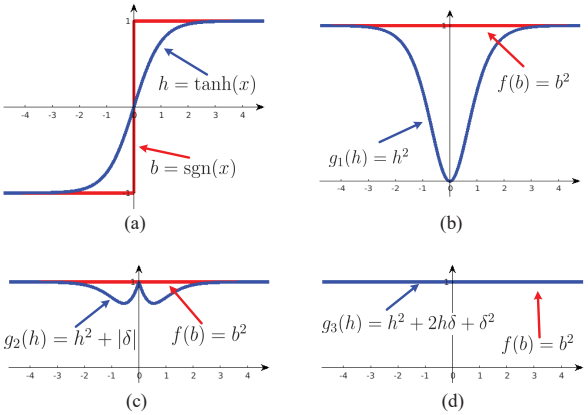


Figure 1. Illustration of the objective discrepancy minimization through series expansion. (a) The sign function $b = \mathrm{sgn}(x)$ is usually relaxed as $h = \tanh(x)$ for gradient based optimization. (b) The objective, e.g. $f(b) = b^2$, is then relaxed as $g_1(h) = h^2$, where large discrepancy is observed. (c) The discrepancy is reduced by considering the quantization loss $|\delta| = |b - h|$ in g_2. (d) We minimize the discrepancy through series expansion in g_3.

1. Introduction

Content-based image retrieval finds similar images from the database given a query image. To measure the similarity, images are characterized by relevant representative features, and then the distances between features are utilized to identify relevant images or nearest neighbors. In the presence of high dimensionality of features and large scale of database, hashing methods have become a promising solution for similarity search [1, 7, 9, 19, 34, 37, 39, 41, 49]. Hashing methods encode images as compact binary codes with similarity preservation in the Hamming space. Learning based hashing [13, 24, 26, 33, 38, 45, 50] mines the data properties and the semantic affinities and shows superior performance than data-independent hashing methods [9, 16].

In learning based hashing, the optimization of similarity preserving objective is a mixed integer optimization problem due to the discrete constraint of binary codes [2, 20, 21, 27, 51, 52], which is incompatible to the gradient based training of neural networks. The key challenge of hashing learning lies on the discrete constraint optimization. Relaxation on the binary code transforms the discrete optimization problem into a continuous one [32, 42, 46]. The discrepancy introduced by relaxation leads to the deviation from the optimal binary codes and the optimal hashing functions. The coupling of the solution to binary codes with the optimization over hashing functions aggravates the optimization of the discrete constraint problem. Although the quantization error is taken into consideration [10, 11, 25] and direct solutions on binary codes are proposed [28, 30, 36], the hashing functions are still learned with a separated optimization over continuous outputs and a post-step sign thresholding. This may deteriorate the quality of hashing functions and fail to generate optimal binary codes.

*Corresponding author (lujiwen@tsinghua.edu.cn).

In this paper, we present a discrepancy minimizing deep hashing (DMDH) method by minimizing the discrepancy between two objectives, the similarity preserving objective over binary codes of training samples and the learnable objective over continuous hashing functions as shown in Figure 1, to improve the quality of learned hashing functions and hence the performance of similarity search. Specifically, we transform the discrete objective over binary codes to a continuous objective over hashing functions through optimal series expansion. The discrepancy between these two objectives is minimized such that the hashing functions capture the similarity preservation and the binary constraint. This guarantees the quality of out-of-sample extension. To minimize the discrepancy, we gradually increase the weight of high order terms of the expanded series in the optimization procedure. Superior experimental results on three benchmark datasets validate the efficacy of the objective discrepancy minimization for hashing.

2. Related Work

A variety of learning based hashing approaches have been proposed in recent years, which can be broadly categorized into unsupervised approaches and supervised approaches [43, 44].

Unsupervised hashing approaches utilize the data distribution of training samples to learn hashing functions for encoding samples as binary codes [3, 10, 11, 25, 30, 32, 46]. Weiss *et al.* [46] proposed Spectral Hashing (SH) to generate binary codes by solving eigenvectors of the graph Laplacian. Liu *et al.* [32] proposed Anchor Graph Hashing (AGH) to exploit the neighborhood structure in a tractable graph based hashing method. Liong *et al.* [25] proposed Deep Hashing (DH) by utilizing a multi-layer neural network as hashing functions to preserve the nonlinear neighborhood relationship. Gong *et al.* [10] proposed ITerative Quantization (ITQ) to balance the variance of PCA results, as a post-step to reduce the quantization loss from real-value features to binary codes. Liu *et al.* [30] proposed Discrete Graph Hashing (DGH) by introducing a tractable alternating optimization method for similarity preservation in a discrete code space. Due to the absence of the label information, the performance of unsupervised hashing approaches is usually surpassed by supervised hashing approaches.

Supervised hashing approaches learn hashing functions on the base of both the label information and the data representation [5, 12, 15, 18, 22, 23, 31, 36, 42, 48]. Wang *et al.* [42] proposed Semi-Supervised Hashing (SSH) to sequentially update hashing functions by leveraging both labelled and unlabelled data. Liu *et al.* [31] proposed Supervised Hashing with Kernels (KSH) to train hashing functions in kernel formulation and measure the similarity with code inner product. Lin *et al.* [23] proposed Two Step Hashing (TSH) to decouple the optimization of binary codes and the optimization of hashing functions. Kulis *et al.* [18] proposed Binary Reconstructive Embedding (BRE) to learn hashing functions through coordinate-descent to minimize the reconstruction error. Shen *et al.* [36] proposed Supervised Discrete Hashing (SDH) to solve the discrete optimization directly with cyclic coordinate descent in conjunction with classification. While these approaches take handcrafted features as input, deep learning based hashing demonstrates retrieval performance breakthrough with the aid of convolutional neural networks [2, 20, 21, 27, 29, 47, 51, 52]. Xia *et al.* [47] proposed CNNH to learn hash codes and convolutional neural network based hashing functions in two separated stages. Lai *et al.* [20] proposed DNNH to simultaneously learn the image feature representation and the hashing coding in a joint optimizing process. Liu *et al.* [27] proposed DSH by utilizing a regularizer to encourage the real-valued outputs of neural networks to be close to binary values. Cao *et al.* [2] proposed HashNet by gradually approximating the non-smooth sign activation with a smoothed activation by a continuation method.

3. Approach

3.1. General Relaxation Framework

Let $\boldsymbol{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ denote a set of n training points, where data points represent either images in the raw RGB space Ω or extracted real value features in $\mathbb{R}^d$. We aim to learn a mapping $\boldsymbol{\Psi}$ from $\boldsymbol{X}$ to $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_x, \ldots, \boldsymbol{b}_n\} \in \{-1, 1\}^{n \times l}$, where l denotes the length of binary codes. We are interested in constructing binary codes such that some notion of similarity is preserved between data points. Thus, we can formulate hashing learning problem as

$$\begin{aligned} &\{\boldsymbol{\Psi}, \boldsymbol{B}\} = \arg\min_{\boldsymbol{\Psi}, \boldsymbol{B}} \mathcal{L}(\boldsymbol{B}), \\ &s.t. \quad \boldsymbol{B} \in \{-1, 1\}^{n \times l} \end{aligned} \tag{1}$$

where $\mathcal{L}(\boldsymbol{B})$ is the predefined loss related to similarity preservation. Given the binary similarity function over the training set $\boldsymbol{S} : \boldsymbol{X} \times \boldsymbol{X} \mapsto \{\pm 1\}$, the hashing learning is conducted under the guidance of the similarity function in the supervised manner with difference minimization, such as similarity-similarity difference minimization [31].

The main bottleneck of optimizing the problem in (1) lies on the discrete constraint on $\boldsymbol{B}$, which makes it intractable to directly optimize the binary codes. Currently, the sign function $b = \mathrm{sgn}(h)$ is adopted to convert the continuous result h as a binary output b. Since the sign function is indifferentiable at zero and with zero gradient for a non-zero input, it is improper to directly employ the gradient-based methods. Most approaches in the literature relax the optimization problem with signed magnitude relaxation [10, 42] or by approximating the sign function with the sigmoid or tanh function [2, 20]. This leads to the relaxed optimization

problem

$$\{\boldsymbol{\Phi}, \tilde{\boldsymbol{B}}\} = \arg\min_{\boldsymbol{\Phi}, \tilde{\boldsymbol{B}}} \tilde{\mathcal{L}}(\tilde{\boldsymbol{B}}), \tag{2}$$

where $\boldsymbol{\Phi}$ is the mapping from $\boldsymbol{X}$ to the relaxed $\tilde{\boldsymbol{B}}$. In the case of tanh relaxation, $\tilde{\mathcal{L}}(\cdot)$ is same as $\mathcal{L}(\cdot)$ and $\tilde{\boldsymbol{B}} = \tanh(\boldsymbol{H})$ with $\boldsymbol{B} = \operatorname{sgn}(\boldsymbol{H})$. To obtain feasible gradients, such nonlinear relaxation would ineluctably slow down or suppress the convergence of the training model. In the case of signed magnitude relaxation, $\tilde{\boldsymbol{B}}$ is the signed magnitude relaxed form of $\boldsymbol{B}$ with $\boldsymbol{B} = \operatorname{sgn}(\tilde{\boldsymbol{B}})$. We denote $\tilde{\boldsymbol{B}}$ as $\boldsymbol{H}$. Then, the loss function $\tilde{\mathcal{L}}(\cdot)$ usually inherits from the original loss function $\mathcal{L}(\cdot)$ in addition to some form of quantization loss between $\boldsymbol{B}$ and $\boldsymbol{H}$, $\mathcal{Q}(\boldsymbol{B}, \boldsymbol{H})$. The objective can be reformulated as

$$\begin{aligned} &\{\boldsymbol{\Phi}, \boldsymbol{H}, \boldsymbol{B}\} = \arg\min_{\boldsymbol{\Phi}, \boldsymbol{H}, \boldsymbol{B}} \left(\mathcal{L}(\boldsymbol{H}) + \mathcal{Q}\left(\boldsymbol{B}, \boldsymbol{H}\right)\right), \\ &s.t. \quad \boldsymbol{B} \in \{-1, 1\}^{n \times l} \end{aligned} \tag{3}$$

where the quantization loss in the literature is computed as $\mathcal{Q}\left(\boldsymbol{B}, \boldsymbol{H}\right) = \left\|\boldsymbol{B} - \boldsymbol{H}\right\|_F^2$ or $\mathcal{Q}\left(\boldsymbol{B}, \boldsymbol{H}\right) = \left\|\boldsymbol{B} - \boldsymbol{H}\boldsymbol{R}\right\|_F^2$ by seeking an orthogonal rotation on the continuous result. Although $\mathcal{L}(\boldsymbol{H})$ is now differentiable with respect to $\boldsymbol{H}$, the optimization of $\mathcal{Q}(\boldsymbol{B}, \boldsymbol{H})$ is still an NP-complete problem. A local minimum could be obtained through the alternating optimization over $\boldsymbol{H}$, through gradient based approaches, over $\boldsymbol{B}$, with optimal solution $\boldsymbol{B} = \operatorname{sgn}(\boldsymbol{H})$ or $\boldsymbol{B} = \operatorname{sgn}(\boldsymbol{H}\boldsymbol{R})$ elementwise, over $\boldsymbol{R}$, if any, as a classic Orthogonal Procrustes problem solved based on SVD. The binary codes of query samples are computed by applying the hash function $\operatorname{sgn}\left(\boldsymbol{\Phi}\left(\boldsymbol{x}\right)\right)$.

3.2. Hashing by Discrepancy Minimization

Optimal Expansion: The relaxation of the objective function (1) transforms the optimization problem over discrete variables $\boldsymbol{B}$ to an optimization problem over continuous variables $\boldsymbol{H}$. Discrepancy between the objective functions is observed by simply replacing the variables $\boldsymbol{B}$ with continuous form $\boldsymbol{H}$ in the loss function $\mathcal{L}(\cdot)$, even with the quantization loss taken into account. Since the desired optimal binary codes and hashing functions are expected to minimize the original objective function over $\boldsymbol{B}$, the optimality of the results is questionable by solving the discrepant objective function. In order to diminish the discrepancy introduced by relaxation, we propose to expand the original objective function at $\boldsymbol{H}$ by the Taylor series

$$\begin{aligned} \mathcal{L}\left(\boldsymbol{B}\right) &= \mathcal{L}\left(\boldsymbol{H} + \Delta\right) \\ &= \mathcal{L}\left(\boldsymbol{H}\right) + \sum_{i=1}^{n \times l} \frac{\partial \mathcal{L}\left(\boldsymbol{H}\right)}{\partial \vec{h}_i} \vec{\Delta}_i \\ &+ \frac{1}{2} \sum_{i=1}^{n \times l} \sum_{j=1}^{n \times l} \frac{\partial^2 \mathcal{L}\left(\boldsymbol{H}\right)}{\partial \vec{h}_i \partial \vec{h}_j} \vec{\Delta}_i \vec{\Delta}_j + \ldots, \end{aligned} \tag{4}$$

where $\Delta = \boldsymbol{B} - \boldsymbol{H}$ are regarded as the increments of variables, $\operatorname{vec}(\cdot)$ denotes the column-wise concatenation, $\vec{h}_i = \left(\operatorname{vec}(\boldsymbol{H})\right)_i$ and $\vec{\Delta}_i = \left(\operatorname{vec}(\Delta)\right)_i$ are the i-th elements of $\operatorname{vec}(\boldsymbol{H})$ and $\operatorname{vec}(\Delta)$, respectively, and the ellipsis represents the higher order terms. By omitting the terms higher than first order, (4) is reduced to the linear approximation, which is similar to (3). One step further, (2) could be derived by even omitting the first order term in (4). While the increments Δ are small, (3) or (2) can be a good approximation of the original objective function. However, we can hardly optimize the objective with gradient based approaches, since the gradients tend to be zero for almost all non-zero inputs. Otherwise, the approximation would deteriorate the retrieval performance. While existing methods control the sharpness of the nonlinear activation function to alleviate this dilemma by introducing an additional coefficient [2], the discrepancy between the objectives before and after relaxation still exists and the value of the coefficient requires careful design. Through expansion by Taylor series, we can resolve such dilemma.

Learning Model: Given a binary similarity function $\boldsymbol{S} : \boldsymbol{X} \times \boldsymbol{X} \mapsto \{\pm 1\}$, where $s_{ij} = \boldsymbol{S}(i, j)$ indicates samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are similar or dissimilar. We aim to mapping samples into the Hamming space as binary codes through the learned hashing functions. We are interested in constructing binary codes such that the similarity relationship between samples is preserved in the Hamming space. That is, Hamming distance is small for the similar pair and vice versa. To this end, we define the specific form of $\mathcal{L}$ over a pair of samples $(\boldsymbol{b}_i, \boldsymbol{b}_j)$ and their similarity indicator s_{ij} as

$$\begin{aligned} &\mathcal{L}\left(\boldsymbol{b}_i, \boldsymbol{b}_j, s_{ij}\right) = s_{ij} \left\|\boldsymbol{b}_i - \boldsymbol{b}_j\right\|^2, \\ &s.t. \quad \boldsymbol{b}_i, \boldsymbol{b}_j \in \{+1, -1\}^l \end{aligned} \tag{5}$$

where $\|\cdot\|$ denotes the L2 norm. In the context of supervised hashing, the similarity function is consistent with the semantic labelling. That is, $s_{ij} = 1$ if and only if samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are assigned with the same label, and vice versa. Thus, many more pairs are observed to be dissimilar rather than similar [35]. To compensate the imbalance of positive and negative pairs, we modified the loss function in (5) by introducing a weighted similarity measurement,

$$\begin{aligned} &\mathcal{L}\left(\boldsymbol{b}_i, \boldsymbol{b}_j, \hat{s}_{ij}\right) = \hat{s}_{ij} \left\|\boldsymbol{b}_i - \boldsymbol{b}_j\right\|^2, \\ &s.t. \quad \boldsymbol{b}_i, \boldsymbol{b}_j \in \{+1, -1\}^l \end{aligned} \tag{6}$$

where

$$\hat{s}_{ij} = \begin{cases} \beta, & \text{if} s_{ij} = 1 \\ \beta - 1, & \text{if} s_{ij} = -1 \end{cases} \tag{7}$$

is the weighted similarity measurement. The parameter β allows the different weighting on the positive and negative pairs. To make the objective reasonable, we set $0 < \beta < 1$, which ensures positive coefficients for similar pairs and

negative coefficients for dissimilar pairs. In the case of $\beta = 0.5$, (6) falls back to (5) with a scaling factor of 0.5.

By summing the loss defined in (5) over all sample pairs, we can reach our objective as

$$\begin{aligned} &\arg\min_{\boldsymbol{B}} \mathcal{L}\left(\boldsymbol{B}, \hat{\boldsymbol{S}}\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} \hat{s}_{ij} \left\|\boldsymbol{b}_i - \boldsymbol{b}_j\right\|^2, \quad (8) \\ &s.t. \quad \boldsymbol{b}_i, \boldsymbol{b}_j \in \{+1, -1\}^l \end{aligned}$$

where $\hat{\boldsymbol{S}}(i,j) = \hat{s}_{ij}$. Rewriting this loss function in the matrix form, we can have

$$\begin{aligned} &\arg\min_{\boldsymbol{B}} \mathcal{L}(\boldsymbol{B}) = \mathrm{tr}\left(\boldsymbol{B}^T \hat{\boldsymbol{D}} \boldsymbol{B}\right), \quad (9) \\ &s.t. \quad \boldsymbol{B} \in \{-1, 1\}^{n \times l} \end{aligned}$$

where $\mathrm{tr}(\cdot)$ is the trace of a matrix and $\hat{\boldsymbol{D}} = \boldsymbol{D} - \hat{\boldsymbol{S}}$. $\boldsymbol{D}$ is a diagonal matrix with $\boldsymbol{D}_{i,i} = \sum_j \hat{\boldsymbol{S}}_{i,j}$.

Discrepancy Minimization: We propose to solve the optimization problem over binary variables $\boldsymbol{B}$ in (9) by transforming it into optimization problem over continuous variables $\boldsymbol{H} \in \mathbb{R}^{n \times l}$ through optimal expansion. Specifically, we expand loss function $\mathcal{L}(\boldsymbol{B})$ in (9) at point $\boldsymbol{H}$ by the series in (4),

$$\begin{aligned} \mathrm{tr}\left(\boldsymbol{B}^T \hat{\boldsymbol{D}} \boldsymbol{B}\right) = &\mathrm{tr}\left(\boldsymbol{H}^T \hat{\boldsymbol{D}} \boldsymbol{H}\right) \\ &+ \mathrm{tr}\left(\Delta^T \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \boldsymbol{H}\right) \\ &+ \mathrm{tr}\left(\Delta^T \hat{\boldsymbol{D}} \Delta\right), \quad (10) \end{aligned}$$

where $\boldsymbol{H}$ is the output of the differentiable mapping $\boldsymbol{\Phi}$ and the hashing function $\boldsymbol{\Psi}$ is constructed by $\mathrm{sgn}(\boldsymbol{\Phi})$. We are interested in optimizing the objective when the last two terms in the expansion are negligible. The minimization of the discrepancy between mappings $\boldsymbol{\Phi}$ and $\boldsymbol{\Psi}$ is critical since we are learning $\boldsymbol{\Phi}$ on the training samples and applying $\boldsymbol{\Psi}$ on the query samples. Note that by approximating the last two terms with a linear function over Δ, it reduces to the quantization loss in the literature. To minimize the objective discrepancy, we enlarge the coefficients of the last two terms in (10) to enforce $\mathrm{tr}\left(\boldsymbol{H}^T \hat{\boldsymbol{D}} \boldsymbol{H}\right)$ approaching $\mathrm{tr}\left(\boldsymbol{B}^T \hat{\boldsymbol{D}} \boldsymbol{B}\right)$. Combining (9) and (10), we formulate the final optimization problem as

$$\begin{aligned} &\arg\min_{\boldsymbol{H}, \Delta} \mathcal{L}(\boldsymbol{H}, \Delta) = \mathrm{tr}\left(\boldsymbol{H}^T \hat{\boldsymbol{D}} \boldsymbol{H}\right) \\ &\qquad + \lambda_1 \mathrm{tr}\left(\Delta^T \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \boldsymbol{H}\right) \\ &\qquad + \lambda_2 \mathrm{tr}\left(\Delta^T \hat{\boldsymbol{D}} \Delta\right), \quad (11) \\ &s.t. \quad (\boldsymbol{H} + \Delta) \in \{-1, 1\}^{n \times l} \end{aligned}$$

Algorithm 1: DMDH

Input: Training set $\boldsymbol{X}$ and parameters λ_1, λ_2.
Output: $\boldsymbol{H}$ and $\boldsymbol{\Phi}$.
Initialize $\boldsymbol{H}$.
for $iteration = 1, 2, \ldots, R$ **do**
  Update $\boldsymbol{H}$ by using the gradients in (13).
  Iteratively update Δ according to (16).
  Enlarge parameters λ_1 and λ_2.
end
Return: $\boldsymbol{H}$ and $\boldsymbol{\Phi}$.

where λ_1 and λ_2 weight the effects of different terms.

To solve the optimization problem in (11), we begin with an initialized value of $\boldsymbol{H}$ (setting λ_1 and λ_2 as 0) and alternately update $\boldsymbol{H}$ and Δ. In each iteration, $\boldsymbol{H}$ are first updated with fixed Δ, and then Δ are updated to minimize the objective function given $\boldsymbol{H}$. We gradually enlarge λ_1 and λ_2 after each iteration to minimize the objective discrepancy. The overall description of the optimization is presented in **Algorithm 1**.

***H*-step:** By fixing Δ, we have the objective as

$$\begin{aligned} \arg\min_{\boldsymbol{H}} \mathcal{L}(\boldsymbol{H}) = &\mathrm{tr}\left(\boldsymbol{H}^T \hat{\boldsymbol{D}} \boldsymbol{H}\right) \\ &+ \lambda_1 \mathrm{tr}\left(\Delta^T \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \boldsymbol{H}\right). \quad (12) \end{aligned}$$

Since $\boldsymbol{H}$ are continuous, we can update them by the stochastic gradient descent method. The gradients of the objective function in (12) over $\boldsymbol{H}$ are computed as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{H}} = \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \boldsymbol{H} + \lambda_1 \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \Delta. \quad (13)$$

Δ-step: By fixing $\boldsymbol{H}$, we have the objective over Δ as

$$\begin{aligned} &\arg\min_{\Delta} \mathcal{L}(\Delta) = \lambda_1 \mathrm{tr}\left(\Delta^T \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \boldsymbol{H}\right) \\ &\qquad + \lambda_2 \mathrm{tr}\left(\Delta^T \hat{\boldsymbol{D}} \Delta\right). \quad (14) \\ &s.t. \quad (\boldsymbol{H} + \Delta) \in \{-1, 1\}^{n \times l} \end{aligned}$$

We have to take effort to deal with the discrete constrained optimization problem. Nevertheless, without considering the discrete constraint, we have the gradient with respect to Δ as

$$\frac{\partial \mathcal{L}}{\partial \Delta} = \lambda_1 \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \boldsymbol{H} + \lambda_2 \left(\hat{\boldsymbol{D}}^T + \hat{\boldsymbol{D}}\right) \Delta. \quad (15)$$

We can iteratively update Δ by

$$\Delta^{(i+1)} = -\mathrm{sgn}\left(\frac{\partial \mathcal{L}}{\partial \Delta}\Big|_{\Delta = \Delta^{(i)}}\right) - \boldsymbol{H}, \quad (16)$$

where $\Delta^{(i)}$ and $\Delta^{(i+1)}$ are the values of Δ at iterations i and $i+1$. The $\mathrm{sgn}(\cdot)$ operator is applied on the matrix elementwise. Note that although the solution over Δ can be applied

Table 1. The comparison of the retrieval performance among all hashing methods in terms of mean average precision over three datasets with bit lengths of 16, 32, 48, and 64.

Methods	CIFAR-10				NUS-WIDE				ImageNet			
	16	32	48	64	16	32	48	64	16	32	48	64
LSH [9]	0.1314	0.1582	0.1723	0.1785	0.4132	0.4827	0.4933	0.5113	0.1110	0.2355	0.3011	0.3419
SH [46]	0.1126	0.1325	0.1113	0.1466	0.4792	0.4912	0.4986	0.5253	0.2088	0.3327	0.3915	0.4110
ITQ [10]	0.2312	0.2432	0.2482	0.2531	0.5573	0.5932	0.6128	0.6166	0.3115	0.4632	0.5223	0.5446
KSH [31]	0.3216	0.3285	0.3371	0.4412	0.4061	0.4182	0.4072	0.3888	0.1620	0.2818	0.3422	0.3934
ITQ-CCA [10]	0.3142	0.3612	0.3662	0.3921	0.5091	0.5443	0.5382	0.6616	0.2546	0.4276	0.5428	0.5527
FastH [22]	0.4532	0.4577	0.4672	0.4854	0.5222	0.6002	0.6472	0.6528	0.2328	0.4337	0.5277	0.5576
SDH [36]	0.4122	0.4301	0.4392	0.4465	0.5342	0.6282	0.6221	0.6335	0.2729	0.4521	0.5329	0.5893
CNNH [47]	0.5373	0.5421	0.5765	0.5780	0.6221	0.6233	0.6321	0.6372	0.2888	0.4472	0.5328	0.5436
DNNH [20]	0.5978	0.6031	0.6087	0.6166	0.6771	0.7023	0.7128	0.7200	0.2887	0.4623	0.5422	0.5586
DPSH [21]	0.6367	0.6412	0.6573	0.6676	0.7015	0.7126	0.7418	0.7423	0.3226	0.5436	0.6217	0.6534
DSH [27]	0.6792	0.6465	0.6624	0.6713	0.7181	0.7221	0.7521	0.7531	0.3428	0.5500	0.6329	0.6645
HashNet [2]	0.6857	0.6923	0.7183	0.7187	0.7331	0.7551	0.7622	0.7762	0.5016	**0.6219**	0.6613	0.6824
DMDH	**0.7037**	**0.7191**	**0.7319**	**0.7373**	**0.7511**	**0.7812**	**0.7886**	**0.7892**	**0.5128**	0.6123	**0.6727**	**0.6916**

for $\boldsymbol{B}$, the obtained binary codes become inferior due to the coupling of similarity preservation and discrete constraint.

4. Experiments

4.1. Datasets and Experimental Settings

The experiments are conducted on three benchmark datasets: CIFAR-10 [40], NUS-WIDE [4], ImageNet [6]. **CIFAR-10** consists of 60,000 manually labelled color images with the size of 32×32. They are evenly divided into 10 categories. We follow the official split of the dataset to construct the training set, 5,000 images from each category, and the test set, 1,000 images from each category. The images from the training set are also used as the database. **NUS-WIDE** is a set of 269,648 images collected from Flickr. This is a multi-label dataset, namely, each image is associated with one or multiple labels from a given 81 concepts. To ensure sufficient samples in each category, we select the images associated with the 21 most frequent concepts, at least 5,000 images per concept and a total of 195,834 images. We randomly sample 5,000 images to form the test set and use the remaining images as the database, 10,000 out of which are selected for training. **ImageNet** is a large scale single labelled image benchmark for visual recognition with over 1.2M images covering 1,000 categories. Following the settings in [2], we select 100 categories and use images associated with them in the provided training set and the validation set as the database and the test set, respectively. To train hashing methods, we sample 100 images from each of the 100 selected categories to construct the training set.

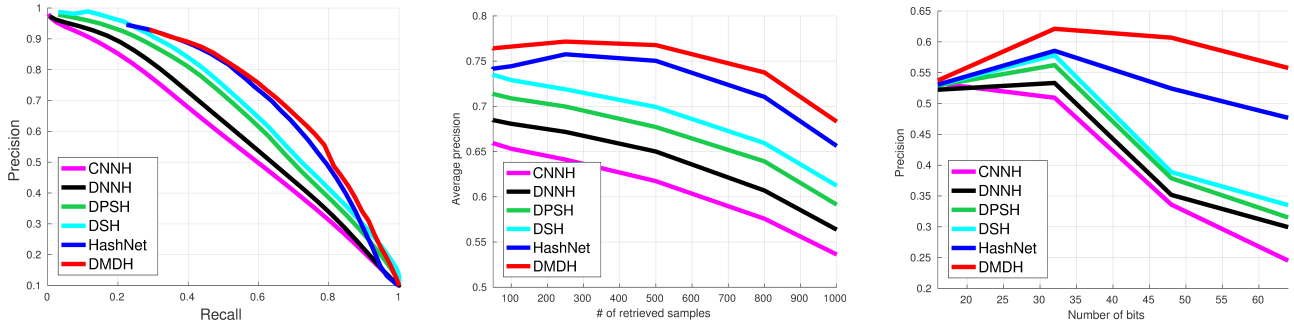
As in previous work [2, 27], the ground truth similarity relationship between images is defined according to the labels. We define the ground truth of semantically similar neighbors as images from the same category. For multiple labelled dataset, NUS-WIDE, we define the ground truth semantic neighbors as images sharing at least one label. Note that the data imbalance, different numbers of positive and negative neighbors, is observed in these datasets under the definition of semantic similarity.

We evaluate the retrieval performance of generated binary codes with three main metrics: mean average precision (MAP), precision at top N returned results (P@N), and Hamming lookup precision within Hamming radius r (HLP@r). The mean average precision provides an overall evaluation of the retrieval performance, which could be further demonstrated by the precision-recall curve. We show the results of MAP@5000 and MAP@1000 for NUS-WIDE and ImageNet datasets, respectively. We choose to evaluate the performance over binary codes with lengths of 16, 32, 48, and 64 bits.

In our implementation of DMDH, we utilize the AlexNet network structure [17] and implement it in the Caffe [14] framework. We initialize the network parameters from the pre-trained model on ImageNet [6]. In the training phase, we set the batch size as 256, momentum as 0.9, and weight decay as 0.005. The learning rate is set to an initial value of 10^{-4} with 40% decrease every 10,000 iterations. We gradually increase the values of λ_1 and λ_2 from 0.6 and 1.2 by a scaling factor of $1 + \#iter \times 5 \times 10^{-4}$ every 200 iterations. For parameter tuning, we evenly split the training set into ten parts to cross validate the parameters.

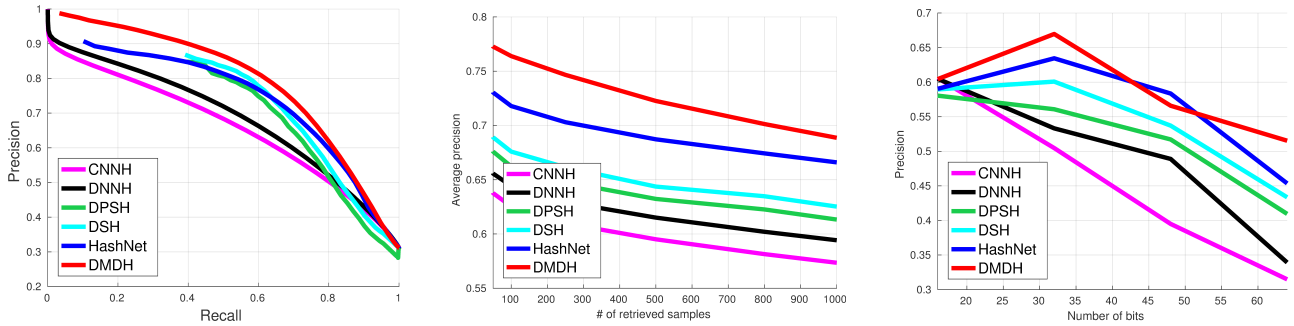
4.2. Results and Analysis

Comparison with the State-of-the-art: We compare the proposed DMDH with ten state-of-the-art hashing methods, including unsupervised hashing: LSH [9], SH [46], ITQ [10], supervised hashing: KSH [31], ITQ-CCA [10], FastH [22], SDH [36], and deep learning based hashing: CNNH [47], DNNH [20], DPSH [21], DSH [27], Hash-



(a) precision-recall (b) P@N (c) HLP@(r==2)

Figure 2. The experimental results on the CIFAR-10 dataset for different deep learning based hashing methods.



(a) precision-recall (b) P@N (c) HLP@(r=2)

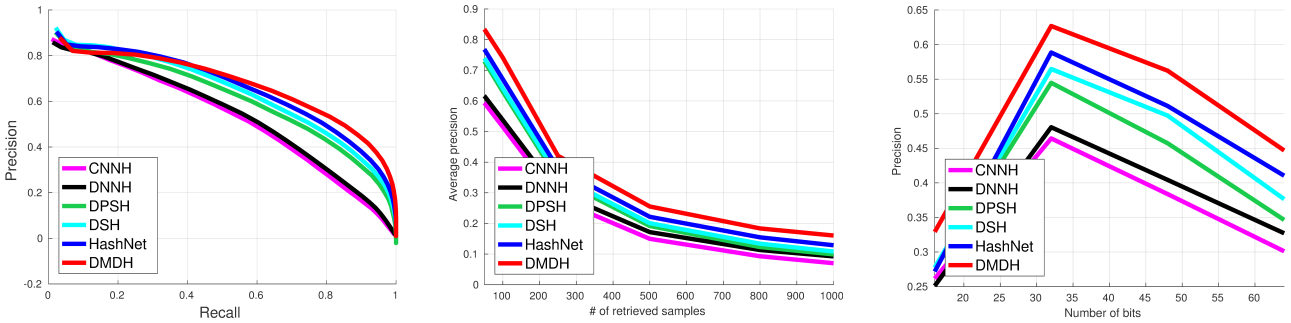
Figure 3. The experimental results on the NUS-WIDE dataset for different deep learning based hashing methods.

Net [2]. We report their results by running the source codes provided by their respective authors to train the models by ourselves, except for DNNH due to the inaccessibility of the source code. We directly use the images as input for all the deep learning based hashing methods, including the proposed DMDH. The images are resized to fit the input of the adopted model. For conventional hashing methods, both unsupervised and supervised hashing, we extract the outputs of the layer 'fc7' in the deep model [8] as input features.

Table 1 shows the retrieval performance of different hashing methods in terms of mean average precision. We can observe that our proposed DMDH delivers the best performance. Compared with the best competitor in deep learning based hashing methods, DMDH consistently outperforms by around 2%. We attribute the performance improvement to the minimization of the objective discrepancy instead of the quantization loss only. Also, we observe that the direct consideration of quantization loss in DSH and HashNet boosts the performance over CNNH, DNNH and DPSH. Compared with the best conventional hashing methods, DMDH boosts the performance by a large margin. We own the advance of the deep learning based hashing method on deep model to the end-to-end training of deep models, which allows the simultaneous learning of feature representation and binarization.

We also observe from the table that the performance of all methods increases by using longer binary codes. While most conventional unsupervised and supervised hashing methods exhibit consistent increase over the test range of bit lengths, the performance of deep learning based hashing methods exhibits saturation. This indicates that deep learning based hashing methods enable the use of more compact binary code for retrieval. While all deep learning based methods show similar trend of saturation, DMDH obtains a higher saturated performance. The saturation arises at different lengths of binary codes on different datasets. For the simple CIFAR-10 dataset with few categories, the performance saturates when the length of binary codes is 16 bits. For the challenge ImageNet dataset with more categories, the performance starts to saturate with the length of binary codes set as 48 bits.

The precision-recall curves for deep learning based hashing methods are shown for 48-bit binary codes in Figures 2(a), 3(a), and 4(a) for CIFAR-10, NUS-WIDE, and ImageNet datasets, respectively. Here we only show the results on the deep learning based hashing methods with the same network model to evaluate the effectiveness of the hashing learning. From the curves in the figures, we can find out that DMDH delivers higher precision than state-of-the-art deep learning based hashing methods at the same rate of recall. This is appreciated in approximate nearest



(a) precision-recall (b) P@N (c) HLP@(r==2)

Figure 4. The experimental results on the ImageNet dataset for different deep learning based hashing methods.

Table 2. The retrieval performance of different variants of DMDH on the CIFAR-10 dataset with different lengths of binary codes.

Methods	Mean average precision			
	16	32	48	64
DMDH	0.7037	0.7191	0.7319	0.7373
DMDH-L	0.6705	0.6955	0.7044	0.6975
DMDH-F	0.6850	0.7027	0.7108	0.7128
DMDH-LF	0.6670	0.6816	0.6845	0.6641

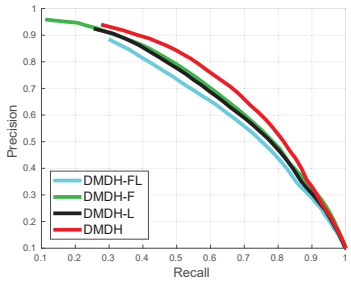


Figure 5. The precision-recall curves on the CIFAR-10 dataset for different variants of DMDH with 64-bit binary codes.

neighbor search because large scale image retrieval is interested in high probability of finding true neighbors rather than finding out the whole set of neighbors.

Figures 2(b), 3(b), and 4(b) show the average precision for 48-bit binary codes with respect to different numbers of top retrieved results on three datasets for deep learning based hashing methods. DMDH consistently provides superior precision than the compared hashing methods for the same amount of retrieved samples. This stands for that more semantic neighbors are retrieved, which is desirable in practical use. We present the Hamming lookup precision within Hamming radius 2 on different lengths of binary codes for deep learning based hashing methods on three datasets in Figures 2(c), 3(c), and 4(c). This metric measures the precision of the retrieved results falling into the buckets within the set Hamming radius. The peak performance is observed at a moderate length of binary codes rather than the longest binary codes. This is because that longer binary code makes the data distribution in Hamming space sparse and fewer samples fall within the set Hamming ball.

Investigation on Different Components: We study the effects of different terms in the objective and the optimization procedure by empirically comparing different variants of DMDH. Specifically, we implement three variants of DMDH, namely DMDH-L, DMDH-F, and DMDH-LF. DMDH-L variant preserves the first order term in (11) by assigning zero coefficients to the high order terms. This is a linear approximation by merely considering the quantization loss in the objective with a gradually increasing coefficient. DMDH-F variant fixes the coefficients in (11) to optimal values by regarding them as hyper-parameters. DMDH-LF variant simultaneously preserves merely the first order term in (11) and fixes the coefficient to a set value during optimization.

We report the performance results of different variants in Table 2 in terms of mean average precision on the CIFAR-10 dataset. The detailed precision-recall curves with binary codes of 64 bits are further shown in Figure 5. By minimizing the discrepancy between objectives before and after relaxation rather than the quantization loss from real-value features to binary features, DMDH outperforms its counterpart DMDH-L. Similar results can be observed between DMDH-F and DMDH-LF. The introduce of high order terms through series expansion guarantees that the optimized objective matches the designed objective on binary codes. The high order terms transform the quantization loss to an objective loss excluding the linear part, which is considered by the first order term. In our case, since the objective is a quadratic function with respect to the binary codes, the expanded terms higher than quadratic terms are with zero coefficients. In DMDH, the optimization is conducted in two alternating steps to enforce the generated codes preserving the similarity and approaching the desired discrete values. By gradually increasing the coefficients of the residual terms, the terms in (11) except the first

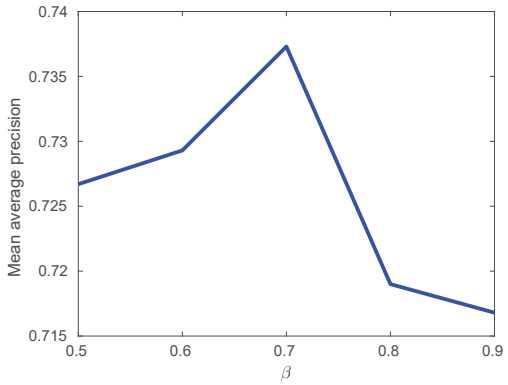


Figure 6. The mean average precision for different β on the CIFAR-10 dataset with 64-bit binary codes.

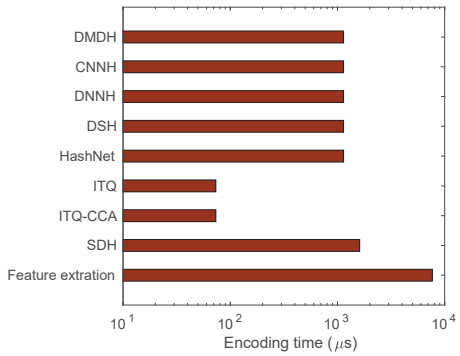


Figure 7. The encoding times of different hashing methods on the CIFAR-10 dataset with 64-bit binary codes.

term, DMDH delivers higher performance than its counterpart DMDH-F. Similar trend is observed between DMDH-L and DMDH-LF. Simultaneously enforcing the two desired properties over the generated codes makes it difficult to train the neural network, slows down the convergence, and even deteriorates the performance. With small coefficients on the residual terms, the network is firstly trained to preserve the similarity relationship between samples. With increasing coefficients for the residual terms, the neural network is trained to generate codes approaching the desired discrete value while keeping the similarity preservation.

Investigation on Weighted Similarity Measurement: We investigate the effect of weighted similarity measurement on dealing with the imbalance of positive and negative pairs. The weighted similarity measurement is controlled by parameter β in (7). By imposing large β, i.e. close to 1, the algorithm merely utilizes the positive pairs to learn hash codes. By imposing small β, i.e. close to 0, the algorithm merely utilizes the negative pairs to learn hash codes. With the definition of semantic similarity and the datasets, imbalance between positive pairs and negative pairs is observed in each training batch since the number of dissimilar samples from different categories is large than the number of similar samples from the same category. Figure 6 shows the variation of performance in terms of mean average precision with respect to β on the CIFAR-10 dataset with length of binary codes set as 64 bits. With the increase of β, the retrieval performance firstly ascends and then declines. Asymmetric similarity measurement shows promising performance enhancement over the symmetric counterpart.

Encoding Time: The encoding time, time cost to generate the binary code for a query sample, is an important factor to evaluate the practical retrieval system. Since the input query samples are originally raw images, we take into consideration both the time cost for feature extraction and the time cost for hashing encoding for conventional hashing methods. We report both the feature extraction time for conventional hashing methods and the encoding time for deep learning based hashing methods on GPU and the hashing encoding time of conventional hashing methods on CPU. The encoding times of involved hashing methods are presented in Figure 7 in logarithmic scale on the CIFAR-10 dataset with 64-bit binary codes. The computing platform is equipped with a 4.0 GHz Intel CPU, 32 GB RAM, and NVIDIA GTX 1080Ti. The encoding time basically depends on the adopted neural network model rather than the hashing method. Thus the time varies little with different lengths of binary codes.

5. Conclusion

In this work, we have proposed to learn to hash by minimizing the objective discrepancy. We transform the discrete optimization problem into a differentiable optimization problem over hashing functions through series expansion with discrepancy minimization. We solve the transformed optimization problem in a tractable alternating optimization framework. We conduct extensive experiments to validate the superiority of the proposed DMDH through comparison with the state-of-the-art hashing methods.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1001001, the National Natural Science Foundation of China under Grant 61672306, Grant U1713214, Grant 61572271, and Grant 61527808, the National 1000 Young Talents Plan Program, the National Postdoctoral Program for Innovative Talents under Grant BX201700137, the National Basic Research Program of China under Grant 2014CB349304, the Shenzhen Fundamental Research Fund (Subject Arrangement) under Grant JCYJ20170412170602564.

References

[1] F. Cakir, K. He, S. Adel Bargal, and S. Sclaroff. Mihash: Online hashing with mutual information. In *ICCV*, pages 437–445, 2017. 1

[2] Z. Cao, M. Long, J. Wang, and P. S. Yu. Hashnet: Deep learning to hash by continuation. In *ICCV*, pages 5608–5617, 2017. 1, 2, 3, 5, 6

[3] M. Á. Carreira-Perpiñán and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, pages 557–566, 2015. 2

[4] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. NUS-WIDE: A real-world web image database from national university of singapore. In *CIVR*, 2009. 5

[5] C. Da, S. Xu, K. Ding, G. Meng, S. Xiang, and C. Pan. Amvh: Asymmetric multi-valued hashing. In *CVPR*, pages 736–744, 2017. 2

[6] J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F. Li. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 5

[7] T.-T. Do, D.-K. Le Tan, T. T. Pham, and N.-M. Cheung. Simultaneous feature aggregating and hashing for large-scale image search. In *CVPR*, pages 6618–6627, 2017. 1

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, pages 647–655, 2014. 6

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999. 1, 5

[10] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824, 2011. 1, 2, 5

[11] W. Hong, J. Yuan, and S. Das Bhattacharjee. Fried binary embedding for high-dimensional visual features. In *CVPR*, pages 2749–2757, 2017. 1, 2

[12] Z. Hu, J. Chen, H. Lu, and T. Zhang. Bayesian supervised hashing. In *CVPR*, pages 6348–6355, 2017. 2

[13] H. Jain, J. Zepeda, P. Perez, and R. Gribonval. Subic: A supervised, structured binary code for image search. In *ICCV*, pages 833–842, 2017. 1

[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. B. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *ACM MM*, pages 675–678, 2014. 5

[15] Q. Jiang and W. Li. Deep cross-modal hashing. In *CVPR*, pages 3232–3240, 2017. 2

[16] S. Kim and S. Choi. Bilinear random projections for locality-sensitive binary codes. In *CVPR*, pages 1338–1346, 2015. 1

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 5

[18] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, pages 1042–1050, 2009. 2

[19] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *TPAMI*, 31(12):2143–2157, 2009. 1

[20] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, pages 3270–3278, 2015. 1, 2, 5

[21] W. Li, S. Wang, and W. Kang. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, pages 1711–1717, 2016. 1, 2, 5

[22] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *CVPR*, pages 1971–1978, 2014. 2, 5

[23] G. Lin, C. Shen, D. Suter, and A. van den Hengel. A general two-step approach to learning-based hashing. In *ICCV*, pages 2552–2559, 2013. 2

[24] Z. Lin, G. Ding, M. Hu, and J. Wang. Semantics-preserving hashing for cross-view retrieval. In *CVPR*, pages 3864–3872, 2015. 1

[25] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, pages 2475–2483, 2015. 1, 2

[26] H. Liu, R. Ji, Y. Wu, F. Huang, and B. Zhang. Cross-modality binary code learning via fusion similarity hashing. In *CVPR*, pages 7380–7388, 2017. 1

[27] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *CVPR*, pages 2064–2072, 2016. 1, 2, 5

[28] L. Liu, L. Shao, F. Shen, and M. Yu. Discretely coding semantic rank orders for supervised image hashing. In *CVPR*, pages 1425–1434, 2017. 1

[29] L. Liu, F. Shen, Y. Shen, X. Liu, and L. Shao. Deep sketch hashing: Fast free-hand sketch-based image retrieval. In *CVPR*, pages 2862–2871, 2017. 2

[30] W. Liu, C. Mu, S. Kumar, and S. Chang. Discrete graph hashing. In *NIPS*, pages 3419–3427, 2014. 1, 2

[31] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012. 2, 5

[32] W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with graphs. In *ICML*, 2011. 1, 2

[33] X. Liu, J. He, C. Deng, and B. Lang. Collaborative hashing. In *CVPR*, pages 2147–2154, 2014. 1

[34] D. Mandal, K. N. Chaudhury, and S. Biswas. Generalized semantic preserving hashing for n-label cross-modal retrieval. In *CVPR*, pages 4076–4084, 2017. 1

[35] B. Neyshabur, N. Srebro, R. Salakhutdinov, Y. Makarychev, and P. Yadollahpour. The power of asymmetry in binary hashing. In *NIPS*, pages 2823–2831, 2013. 3

[36] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *CVPR*, pages 37–45, 2015. 1, 2, 5

[37] Y. Shen, L. Liu, L. Shao, and J. Song. Deep binaries: Encoding semantic-rich cues for efficient textual-visual cross retrieval. In *ICCV*, pages 4097–4106, 2017. 1

[38] J. Song, T. He, L. Gao, X. Xu, A. Hanjalic, and H. T. Shen. Binary generative adversarial networks for image retrieval. In *AAAI*, 2018. 1

[39] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *ACM SIGMOD*, pages 785–796, 2013. 1

[40] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *TPAMI*, 30(11):1958–1970, 2008. 5

[41] H. Venkateswara, J. Eusebio, S. Chakraborty, and S. Panchanathan. Deep hashing network for unsupervised domain adaptation. In *CVPR*, pages 5018–5027, 2017. 1

[42] J. Wang, S. Kumar, and S. Chang. Semi-supervised hashing for large-scale search. *TPAMI*, 34(12):2393–2406, 2012. 1, 2

[43] J. Wang, W. Liu, S. Kumar, and S. Chang. Learning to hash for indexing big data - A survey. *Proceedings of the IEEE*, 104(1):34–57, 2016. 2

[44] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen. A survey on learning to hash. *PAMI*, 40(4):769–790, 2018. 2

[45] Y.-X. Wang, L. Gui, and M. Hebert. Few-shot hash learning for image retrieval. In *ICCV*, pages 1228–1237, 2017. 1

[46] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008. 1, 2, 5

[47] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, pages 2156–2162, 2014. 2, 5

[48] E. Yang, C. Deng, W. Liu, X. Liu, D. Tao, and X. Gao. Pairwise relationship guided deep hashing for cross-modal retrieval. In *AAAI*, pages 1618–1625, 2017. 2

[49] T. Yu, Z. Wang, and J. Yuan. Compressive quantization for fast object instance search in videos. In *ICCV*, pages 726–735, 2017. 1

[50] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *TIP*, 24(12):4766–4779, 2015. 1

[51] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, pages 1556–1564, 2015. 1, 2

[52] H. Zhu, M. Long, J. Wang, and Y. Cao. Deep hashing network for efficient similarity retrieval. In *AAAI*, pages 2415–2421, 2016. 1, 2