Supervised Short-Length Hashing

Xingbo Liu[1] , **Xiushan Nie**[2,*] , **Quan Zhou**[3] , **Xiaoming Xi**[2] , **Lei Zhu**[4] and **Yilong Yin**[3,†]
[1]School of Computer Science and Technology, Shandong University, Jinan, P.R. China
[2]School of Computer Science and Technology, Shandong Jianzhu University, Jinan, P.R. China
[3]School of Software, Shandong University, Jinan, P.R. China
[4]School of Information Science and Engineering, Shandong Normal University, Jinan, P.R. China
sclxb@mail.sdu.edu.cn, niexiushan@163.com, ylyin@sdu.edu.cn

Abstract

Hashing can compress high-dimensional data into compact binary codes, while preserving the similarity, to facilitate efficient retrieval and storage. However, when retrieving using an extremely short-length hash code learned by the existing methods, the performance cannot be guaranteed because of severe information loss. To address this issue, in this study, we propose a novel supervised short-length hashing (SSLH). In this proposed SSLH, mutual reconstruction between the short-length hash codes and original features are performed to reduce semantic loss. Furthermore, to enhance the robustness and accuracy of the hash representation, a robust estimator term is added to fully utilize the label information. Extensive experiments conducted on four image benchmarks demonstrate the superior performance of the proposed SSLH with short-length hash codes. In addition, the proposed SSLH outperforms the existing methods, with long-length hash codes. To the best of our knowledge, this is the first linear-based hashing method that focuses on both short- and long-length hash codes for maintaining high precision.

1 Introduction

With the exponential growth of data, the approximate nearest neighbor (ANN) search in which the ANNs of a query sample are determined within a large database has become vital in many applications, including web image and video retrieval [Li *et al.*, 2015] [Hao *et al.*, 2017]. Hashing provides high efficiency with respect to both storage cost and query speed, and has received significant attention in information retrieval. Hashing encodes media data into a string of complex binary codes, while preserving the similarity of the original media data. The distance in binary codes is calculated using the Hamming distance, which can be implemented on hardware using bit-wise XOR operations, and provides highly efficient computation, compared to the other distance calculations [Wang *et al.*, 2018].

*Corresponding Author.
†Corresponding Author.

Learning-based hashing, which aims at generating binary hash codes by learning the projections under the guidance of the original data, is one of the extensively-used hashing methods that can provide superior retrieval performance by analyzing the underlying data characteristics. The existing learning-based hashing methods can be roughly divided into two main categories: unsupervised and supervised. Unsupervised hashing does not use label information for the training samples, while supervised hashing methods make full use of the class labels. In general, supervised hashing can perform significantly better than the unsupervised. Therefore, many supervised hashing methods [Shen *et al.*, 2015], [Gui *et al.*, 2018], [Gui *et al.*, 2018], [Luo *et al.*, 2018a], [Lin *et al.*, 2019] been proposed in recent years.

Compactness, which implies that the obtained hash code should be compact and as short as possible, is a crucial factor in learning-based hashing. In general, given a dataset with c categories, the length, L, of the hash code should be greater than $log_2(c)$; else, the hash codes cannot distinguish the samples. Therefore, in this study, we define *short-length* as the length, which is slightly greater than $log_2(c)$. For example, for dataset, CIFAR-10, which has 10 categories, $log_2(10)$=3.3, and a length of four can be considered as a short length. Generally, when the hash-code length is smaller than the number of categories, c (note that it is always considerably greater than $log_2(c)$), or even smaller than the feature representation dimension, the performances of the existing hashing methods degrade significantly. Therefore, the length of the hash codes learned by the existing methods is always considerably greater than the number of categories c (e.g. 48, 64, or 128).

Short-length hash codes can reduce the storage cost and computation complexity, and accelerate the retrieval speed [Luo *et al.*, 2018b]. However, retrieval with short-length hash codes usually leads to unexpected poor performance due to the following: 1) *Weak classification*: Classification ability is important in information retrieval, and hash codes with shorter length lead to weaker classification ability. Therefore, enhancing the classification ability is vital for short-length hashing. 2) *Severe information loss*: Due to considerable dimension reduction, representing a sample using a short-length hash code leads to severe information loss. 3) *Easy to achieve suboptimal solution*: Trivial solutions are generally easier suboptimal solutions during short-length hash learning, particularly, when the hash code length, L, is considerably

smaller than the number of categories, c.

In order to address the aforementioned issues, we propose a novel discrete hashing method, termed supervised short-length hashing (SSLH). In the proposed SSLH, a robust estimator is presented with label information to enhance the classification ability, and mutual reconstruction between the hash codes and original features is performed for reducing information loss. In addition, to avoid suboptimal solutions, the balance constraint and regularization are utilized during hash learning.

The SSLH significantly surpasses the existing hashing methods with short-length hash codes. In addition, considerable improvement in performance is achieved with long-length hash codes. To the best of our knowledge, the SSLH is the first linear-model hashing that focuses on both long- and short-length hash codes for maintaining high precision. The main contributions of this study are summarized as follows:

- We propose a new supervised discrete hashing method for short-length hash learning. In the proposed method, to achieve better performance, robust and mutual regression, matrix factorization, discrete optimization, and balance constraint are seamlessly combined for short-length hash learning.
- Stable and high-precision for both short- and long-length hash codes are achieved using the proposed method. Experiments based on four large-scale datasets demonstrate that the proposed method can significantly improve the performance with short-length hash codes, while considerably improving the performance with the long-length ones, under various scenarios.

2 Proposed Method

2.1 Formulation

Assume that we have a training set, $\mathbf{X}$, consisting of n instances; i.e., $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$, each instance can be represented by an m-dimensional feature. Moreover, a semantic label matrix, $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^n$, is available, with $\mathbf{y}_i = \{y_{ij}\} \in \{-1, +1\}^c$ being the label vector of the i-th instance, where c is the number of categories in the training set. If the i-th instance belongs to the j-th semantic category, $y_{ij} = 1$, and -1 otherwise. The hash matrix is defined as $\mathbf{H} = \{\mathbf{h}_i\}_{i=1}^n$. $||\mathbf{H}||$ and $\mathbf{H}^T$ denote the ℓ_2-norm and transpose of matrix, $\mathbf{H}$, respectively.

To utilize the label information effectively, least squares regression is widely adopted in the existing hashing methods. However, the ordinary least squares regression might be sensitive to outliers [Lawson and Hanson, 1995], [He *et al.*, 2012], [Gui and Li, 2018]. Therefore, in this study, we adopt the concept of "correntropy" [Liu *et al.*, 2007] to enhance the classification ability with a robust estimator. Given a hash matrix, $\mathbf{H}$, and label matrix, $\mathbf{Y}$, a linear projection, $\mathbf{W}$, is defined for describing the relationship between them. Then, the robust estimator can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{H}} G(\mathbf{Y}, \mathbf{W}^T\mathbf{H}) = \min_{\mathbf{W},\mathbf{H}} Tr(\mathbf{E}\mathbf{D}\mathbf{E}^T) \\ &\text{s.t.} \quad \mathbf{H} \in \{-1, +1\}^{L\times n}, \end{aligned} \tag{1}$$

where $Tr(\cdot)$ is the trace of "$\cdot$", $\mathbf{E} = \mathbf{Y} - \mathbf{W}^T\mathbf{H}$, $\mathbf{D}$ is a diagonal matrix of size $n \times n$ and the i-th diagonal element is defined as

$$D_i = exp(-\left\|(\mathbf{Y} - \mathbf{W}^T\mathbf{H})^i\right\|^2 / \delta^2), \tag{2}$$

δ is a hyperparameter, which will be set empirically, and $(\mathbf{Y} - \mathbf{W}^T\mathbf{H})^i$ is the i-th column of $(\mathbf{Y} - \mathbf{W}^T\mathbf{H})$.

Moreover, the kernel trick is adopted to deal with the loss of feature information. The kernel trick can capture the local structure of the data and reduce the feature dimension with nonlinear projections. Given a training set, $\mathbf{X}$, a radial basis function is used to map an m-dimension feature to d-dimension features. Specifically, we first randomly select d anchor points, $\{\mathbf{p}_i\}_{i=1}^d$, from the training set, and each training sample can be transformed into a new representation through

$$\varphi(\mathbf{x}) = exp(-\left\|\mathbf{x} - \mathbf{p}_i\right\|_2^2 / 2\sigma^2)_{i=1}^d, \tag{3}$$

where $\{\mathbf{p}_i\}_{i=1}^d$. This process can be calculated in advance; we use $\mathbf{V}$ to represent $\varphi(\mathbf{X})$ in the following sections for conciseness.

According to the matrix factorization theory [Deerwester *et al.*, 1990], the latent semantic feature from source datasets can be learned by matrix factorization. As the hash matrix, $\mathbf{H}$, is a semantic representation of the samples, it can be considered as the latent semantic feature of the data. We then have the following equation, according to the matrix factorization theory:

$$\mathbf{V} = \mathbf{U}^T\mathbf{H}, \tag{4}$$

where $\mathbf{U}$ is a projection. For out-of-sample extension, we need to learn a projection, $\mathbf{P}$, from the original feature to the hash code, which is

$$\mathbf{H} = \mathbf{P}^T\mathbf{V}. \tag{5}$$

Obviously, Eqs. (4) and (5) can be considered as mutual reconstructions between the original feature and the hash matrix. Therefore, minimizing the reconstruction loss will reduce information loss in short-length hash learning. Then, this problem can be formulated as follows:

$$\begin{aligned} &\min_{\mathbf{H},\mathbf{P},\mathbf{U}} \alpha \left\|\mathbf{H} - \mathbf{P}^T\mathbf{V}\right\|^2 + \beta \left\|\mathbf{V} - \mathbf{U}^T\mathbf{H}\right\|^2 \\ &\text{s.t.} \quad \mathbf{H} \in \{-1, +1\}^{L\times n}, \end{aligned} \tag{6}$$

where α and β are parameters. Compared to the existing hashing methods that only use one type of reconstruction, the proposed method can preserve more information.

Furthermore, according to the balance property of hash codes, each hash bit has a 50% chance of being -1 or $+1$ in a hash vector [Shen *et al.*, 2017] [Jiang and Li, 2017]. In this study, the balance property can be formulated as

$$\min_{\mathbf{H}} \left\|\mathbf{H}\mathbf{1}\right\|^2 \qquad \text{s.t.} \quad \mathbf{H} \in \{-1, +1\}^{L\times n}, \tag{7}$$

where $\mathbf{1}$ is a all-ones vector. Balance constraint can avoid trivial solutions. In addition, the utilization of projection, $\mathbf{U}$, can avoid trivial solutions because trivial solutions for $\mathbf{H}$ cannot reconstruct the feature well and will render the loss considerable.

In order to obtain a smooth solution, prevent over-fitting, and improve the stability of linear regression [Hoerl and Kennard, 1970], the ℓ_2-norm regularization is adopted in this study. The regularization for $\mathbf{W}$, $\mathbf{P}$, and $\mathbf{U}$ can be formulated as

$$\min_{\mathbf{W},\mathbf{P},\mathbf{U}} R(\mathbf{W},\mathbf{P},\mathbf{U}) = \min_{\mathbf{W},\mathbf{P},\mathbf{U}} \lambda(\|\mathbf{W}\|^2 + \|\mathbf{P}\|^2 + \|\mathbf{U}\|^2), \tag{8}$$

where λ is a hyperparameter.

Finally, the SSLH is formulated as

$$\begin{aligned} &\min_{\mathbf{W},\mathbf{H},\mathbf{P},\mathbf{U}} G(\mathbf{Y},\mathbf{W}^T\mathbf{H}) + \alpha \left\|\mathbf{H}-\mathbf{P}^T\mathbf{V}\right\|^2 \\ &+ \beta \left\|\mathbf{V}-\mathbf{U}^T\mathbf{H}\right\|^2 + \gamma \|\mathbf{H}\mathbf{1}\|^2 + R(\mathbf{W},\mathbf{P},\mathbf{U}) \\ &\text{s.t.} \quad \mathbf{H} \in \{-1,+1\}^{L\times n}, \end{aligned} \tag{9}$$

where α , β, and γ are parameters.

2.2 Optimization

It is challenging to optimize Eq. (9) directly because it is nonconvex and noncontinuous. However, we try to solve this nondifferentiable problem with an iterative framework using the following steps .

W-Step: Learn the projection, $\mathbf{W}$, with the other variables fixed. The problem in Eq. (9) becomes

$$\min_{\mathbf{W}} Tr((\mathbf{Y}-\mathbf{W}^T\mathbf{H})^T\mathbf{D}(\mathbf{Y}-\mathbf{W}^T\mathbf{H})) + \lambda\|\mathbf{W}\|^2. \tag{10}$$

Setting the derivative of Eq. (10) with respect to $\mathbf{W}$ as zero yields

$$\mathbf{W} = (\mathbf{H}\mathbf{D}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{H}\mathbf{D}\mathbf{Y}^T. \tag{11}$$

H-Step: Learn the binary code, $\mathbf{H}$, with the other variables fixed. The problem in Eq. (9) becomes

$$\begin{aligned} &\min_{\mathbf{H}} Tr((\mathbf{Y}-\mathbf{W}^T\mathbf{H})^T\mathbf{D}(\mathbf{Y}-\mathbf{W}^T\mathbf{H})) \\ &+ \alpha\left\|\mathbf{H}-\mathbf{P}^T\mathbf{V}\right\|^2 + \beta\left\|\mathbf{V}-\mathbf{U}^T\mathbf{H}\right\|^2 + \gamma\|\mathbf{H}\mathbf{1}\|^2 \\ &\text{s.t.} \quad \mathbf{H} \in \{-1,+1\}^{L\times n}. \end{aligned} \tag{12}$$

Eq. (12) can be reformulated as

$$\begin{aligned} &\min_{\mathbf{H}} Tr((\mathbf{W}^T\mathbf{H})\mathbf{D}(\mathbf{H}^T\mathbf{W})) + \alpha(\|\mathbf{H}\|^2 - 2Tr(\mathbf{H}^T\mathbf{P}^T\mathbf{V})) \\ &+ \beta(-2Tr(\mathbf{H}^T\mathbf{U}\mathbf{V}) + \left\|\mathbf{U}^T\mathbf{H}\right\|^2 + \gamma\|\mathbf{H}\mathbf{1}\|^2) \\ &\text{s.t.} \quad \mathbf{H} \in \{-1,+1\}^{L\times n}. \end{aligned} \tag{13}$$

As $\|\mathbf{H}\|^2 = L * n$, Eq. (13) can be rewritten as

$$\begin{aligned} &\min_{\mathbf{H}} Tr((\mathbf{W}^T\mathbf{H})\mathbf{D}(\mathbf{H}^T\mathbf{W})) + \beta\left\|\mathbf{U}^T\mathbf{H}\right\|^2 - Tr(\mathbf{H}^T\mathbf{Q}) \\ &+ \gamma\|\mathbf{H}\mathbf{1}\|^2 \qquad \text{s.t.} \quad \mathbf{H} \in \{-1,+1\}^{L\times n}, \end{aligned} \tag{14}$$

where $\mathbf{Q} = \mathbf{W}\mathbf{Y} + \alpha\mathbf{P}^T\mathbf{V} + \beta\mathbf{U}\mathbf{V}$.

Although Eq. (14) is difficult to solve because $\mathbf{H}$ is discrete, we can directly leverage the discrete cyclic coordinate descent (DCC) approach [Shen *et al.*, 2015] to learn $\mathbf{H}$ bit-by-bit iteratively, until convergence or a fixed number of iterations.

Algorithm 1 Supervised Short-Length Hashing (SSLH)

Input: Training set: $\mathbf{X}$; semantic label: $\mathbf{Y}$; code length: L; hyperparameters: δ, d, α, β, γ, and λ; number of iterations: T.

1: Initialize $\mathbf{W}$, $\mathbf{P}$, and $\mathbf{U}$ as random zero-centered matrices, and $\mathbf{H}$ as a random $\{-1,1\}^{L\times n}$ matrix. Calculate $\mathbf{V}$ using the kernel function according to Eq. (3).
2: **repeat**
3: **W-Step** Use Eq. (11) to solve $\mathbf{W}$, while the other variables are fixed.
4: **H-Step** Use Eq. (16) to solve $\mathbf{H}$ bit-by-bit, while the other variables are fixed.
5: **P-Step** Use Eq. (18) to solve $\mathbf{P}$, while the other variables are fixed.
6: **U-Step** Use Eq. (20) to solve $\mathbf{U}$, while the other variables are fixed.
7: **D-Step** Use Eq. (2) to solve $\mathbf{D}$, while the other variables are fixed.
8: **until** Convergence or a fixed number of iterations.

Output: Hash matrix: $\mathbf{H}$; projection matrix: $\mathbf{P}$.

Specifically, define $\mathbf{h}^T$ as the l-th row of matrix, $\mathbf{H}$, $l = 1,\cdots,L$, and $\mathbf{H}^{'}$ as matrix, $\mathbf{H}$, excluding $\mathbf{h}$. Analogously, define $\mathbf{q}^T$ as the l-th row of matrix, $\mathbf{Q}$. Next, define $\mathbf{w}^T$ as the l-th row of matrix, $\mathbf{W}$, and $\mathbf{W}^{'}$ as matrix, $\mathbf{W}$, excluding $\mathbf{w}$. Then, define $\mathbf{u}^T$ as the l-th row of matrix, $\mathbf{U}$, and $\mathbf{U}^{'}$ as matrix, $\mathbf{U}$, excluding $\mathbf{u}$. The problem in Eq. (14) becomes

$$\begin{aligned} &\min_{\mathbf{h}} \mathbf{h}^T(\mathbf{D}\mathbf{H}^{'T}\mathbf{W}^{'}\mathbf{w}^T + \beta\mathbf{H}^{'T}\mathbf{U}^{'}\mathbf{u}^T + \gamma\mathbf{H}^{'T}\mathbf{1} - \mathbf{q}^T) \\ &\text{s.t.} \quad \mathbf{h} \in \{-1,+1\}^{n\times 1}. \end{aligned} \tag{15}$$

The analytic solution of $\mathbf{h}$ can be expressed as

$$\mathbf{h} = sgn(\mathbf{q} - \mathbf{w}\mathbf{W}^{'T}\mathbf{H}^{'}\mathbf{D}^T - \beta\mathbf{u}\mathbf{U}^{'T}\mathbf{H}^{'} - \gamma\mathbf{1}^T\mathbf{H}^{'}), \tag{16}$$

where $sgn(\cdot)$ is a sign function.

P-Step: Learn the projection matrix, $\mathbf{P}$, while holding the other variables fixed. The problem in Eq. (9) becomes

$$\min_{\mathbf{P}} \left\|\mathbf{H}-\mathbf{P}^T\mathbf{V}\right\|^2 + \lambda\|\mathbf{P}\|^2. \tag{17}$$

The closed-form solution of $\mathbf{P}$ is

$$\mathbf{P} = (\mathbf{V}\mathbf{V}^T + \lambda\mathbf{I})^{-1}\mathbf{V}\mathbf{H}^T. \tag{18}$$

U-Step: Learn the projection matrix, $\mathbf{U}$, while holding the other variables fixed. The problem in Eq. (9) becomes

$$\min_{\mathbf{U}} \left\|\mathbf{V}-\mathbf{U}^T\mathbf{H}\right\|^2 + \lambda\|\mathbf{U}\|^2. \tag{19}$$

The closed-form solution of $\mathbf{U}$ is

$$\mathbf{U} = (\mathbf{H}\mathbf{H}^T + \lambda\mathbf{I})^{-1}\mathbf{H}\mathbf{V}^T. \tag{20}$$

D-Step: $\mathbf{D}$ can be solved as Eq. (2).

In summary, we try to solve the nonconvex mixed integer optimization problem using above steps . Convergence is reached within a few iterations, which is demonstrated in the "Experimental" section. The algorithm for solving the SSLH is presented as Algorithm 1.

Method	CIFAR-10						CALTECH-101					
	4 bits	6 bits	8 bits	10 bits	32 bits	48 bits	8 bits	10 bits	12 bits	14 bits	32 bits	48 bits
LSH	0.1897	0.1986	0.2215	0.2349	0.2102	0.2474	0.0929	0.1064	0.1260	0.1281	0.1455	0.1801
SH	0.2564	0.2801	0.2772	0.2838	0.2898	0.2961	0.2043	0.2053	0.2450	0.2744	0.2048	0.2089
PCA-ITQ	0.1581	0.2062	0.2283	0.2657	0.3398	0.3562	0.0530	0.0777	0.0936	0.1035	0.1390	0.1681
PCA-RR	0.1897	0.2081	0.2538	0.2533	0.2938	0.3172	0.1196	0.1085	0.1301	0.1601	0.1637	0.1900
MFH	0.2590	0.2643	0.2766	0.2816	0.3473	0.3623	0.2187	0.2558	0.2750	0.2914	0.2724	0.2964
SDH	0.1683	0.3222	0.3220	0.4177	0.6507	0.6626	0.1016	0.1151	0.1448	0.1369	0.2792	0.3060
COSDISH	0.2510	0.3006	0.3899	0.3898	0.6154	0.6595	0.2220	0.2194	0.2312	0.2404	0.2745	0.3452
FSDH	0.2166	0.3014	0.4768	0.5727	0.6526	0.6632	0.1133	0.1307	0.1399	0.1587	0.2917	0.4108
SSDH	0.2413	0.2965	0.4166	0.4094	0.6022	0.6124	0.2849	0.3002	0.3396	0.3298	0.3851	0.4104
SSLH	**0.4356**	**0.5215**	**0.5722**	**0.6058**	**0.6641**	**0.6700**	**0.3695**	**0.4078**	**0.4330**	**0.4458**	**0.4926**	**0.5256**

Table 1: Performance in terms of mAP score on single-label datasets. The best results for mAP are shown in bold.

Method	MS-COCO						NUS-WIDE					
	7 bits	8 bits	9 bits	10 bits	32 bits	48 bits	5 bits	6 bits	7 bits	8 bits	32 bits	48 bits
LSH	0.4663	0.4665	0.4670	0.4712	0.4833	0.4838	0.3096	0.3157	0.3243	0.3372	0.3424	0.3541
SH	0.6426	0.6646	0.6777	0.6795	0.6529	0.6800	0.5415	0.5678	0.5689	0.5874	0.6058	0.6070
PCA-ITQ	0.4789	0.4894	0.4979	0.5090	0.6578	0.6907	0.3095	0.3095	0.3095	0.3095	0.4544	0.6016
PCA-RR	0.5333	0.5533	0.5392	0.5480	0.6481	0.6594	0.3796	0.4172	0.4192	0.4335	0.5500	0.6282
MFH	0.6054	0.6118	0.6221	0.6231	0.6470	0.6500	0.5382	0.5450	0.5557	0.5636	0.6244	0.6315
SDH	0.5246	0.5505	0.5327	0.6001	0.6489	0.6531	0.5023	0.5240	0.5775	0.6100	0.7159	0.7350
COSDISH	0.4703	0.4867	0.4764	0.5693	0.6390	0.7164	0.3096	0.3696	0.4242	0.5242	0.7242	0.7274
FSDH	0.5225	0.5664	0.5754	0.6004	0.7197	0.7235	0.5976	0.6125	0.5977	0.5938	0.7219	0.7588
SSDH	0.6588	0.6578	0.6680	0.7127	0.7219	0.7588	0.5964	0.6263	0.6384	0.6579	0.7574	0.7763
SSLH	**0.8080**	**0.8199**	**0.8252**	**0.8254**	**0.8644**	**0.8683**	**0.6635**	**0.6969**	**0.7197**	**0.7331**	**0.8042**	**0.8129**

Table 2: Performance in terms of mAP score on multi-label datasets. The best results for mAP are shown in bold.

2.3 Time Complexity

The time complexity for learning projection, $\mathbf{W}$, is $T \cdot O(2nL^2 + ncL)$, whereas the time complexities for learning projections $\mathbf{P}$ and $\mathbf{U}$ are $T \cdot O(2nd^2 + ndL)$ and $T \cdot O(2nL^2 + ndL)$, respectively. The time complexity for learning hash code, $\mathbf{H}$, is $T \cdot O(ndL^2 + ncL^2)$. As d is considerably greater than c and L, the total training time complexity of the proposed method can be simplified as $T \cdot O(ndL^2)$. The time complexity is linearly related to the size of the dataset, making the proposed method scalable to larger datasets.

3 Experiments

In this section, we present the experimental settings and results. The hyperparameter settings employed are listed below. Extensive experiments were conducted on four image datasets to evaluate the proposed method and compare it with several state-of-the-art methods. The experiments were performed on a computer with an Intel(R) Core(TM) i7-4790 CPU and 32-GB RAM.

3.1 Datasets and Experimental Settings

Four extensively-used image datasets were utilized in the experiments: CIFAR-10 [1] [Krizhevsky and Hinton, 2009], CALTECH-101 [2] [Fei-Fei *et al.*, 2007], MS-COCO [3] [Lin *et al.*, 2014], and NUS-WIDE [4] [Chua *et al.*, 2009].

CIFAR-10 is a single-label dataset containing 60,000 images belonging to 10 classes, with 6,000 images per class. We randomly selected 5,000 and 1,000 images (100 images per class) from the dataset as the training and testing sets, respectively.

CALTECH-101 is a single-label dataset including 8,677 images belonging to 101 categories, with 40 to 800 images per category. We randomly selected 5,000 and 1,000 images from the dataset as the training and testing sets, respectively.

The MS-COCO dataset is a multilabel dataset containing 82,783 images belonging to 91 categories. For the training image set, images with no category information were discarded and 82,081 remained. We randomly selected 10,000 and 5,000 images from the dataset as the training and testing sets, respectively.

The NUS-WIDE dataset contains 269,648 web images associated with 1,000 tags. In this multilabel dataset, each image may be annotated with multiple labels. We selected only 195,834 images belonging to the 21 most frequent concepts. We randomly selected 10,500 (500 from each concept) and 2,100 (100 from each concept) images from the dataset as the training and testing sets, respectively. For multi-label datasets, two images were defined as a similar pair, if they shared at least one common label.

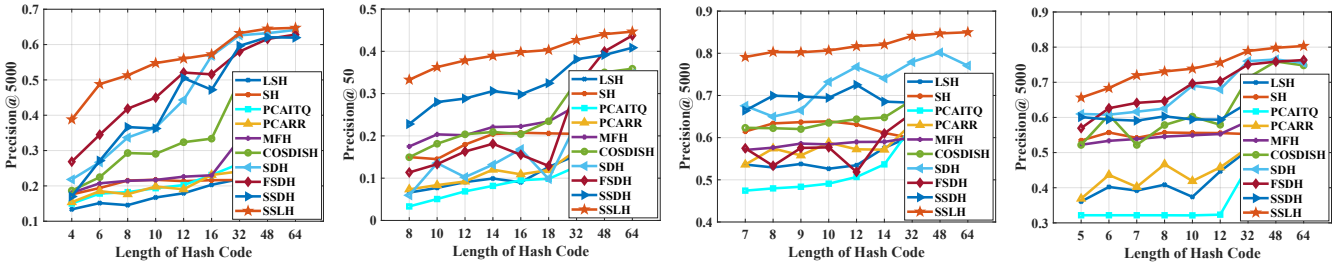
For the CIFAR-10, MS-COCO, and NUS-WIDE, we used

[1] https://www.cs.toronto.edu/∼kriz/cifar.html.

[2] http://www.vision.caltech.edu/Image_Datasets/Caltech101/.

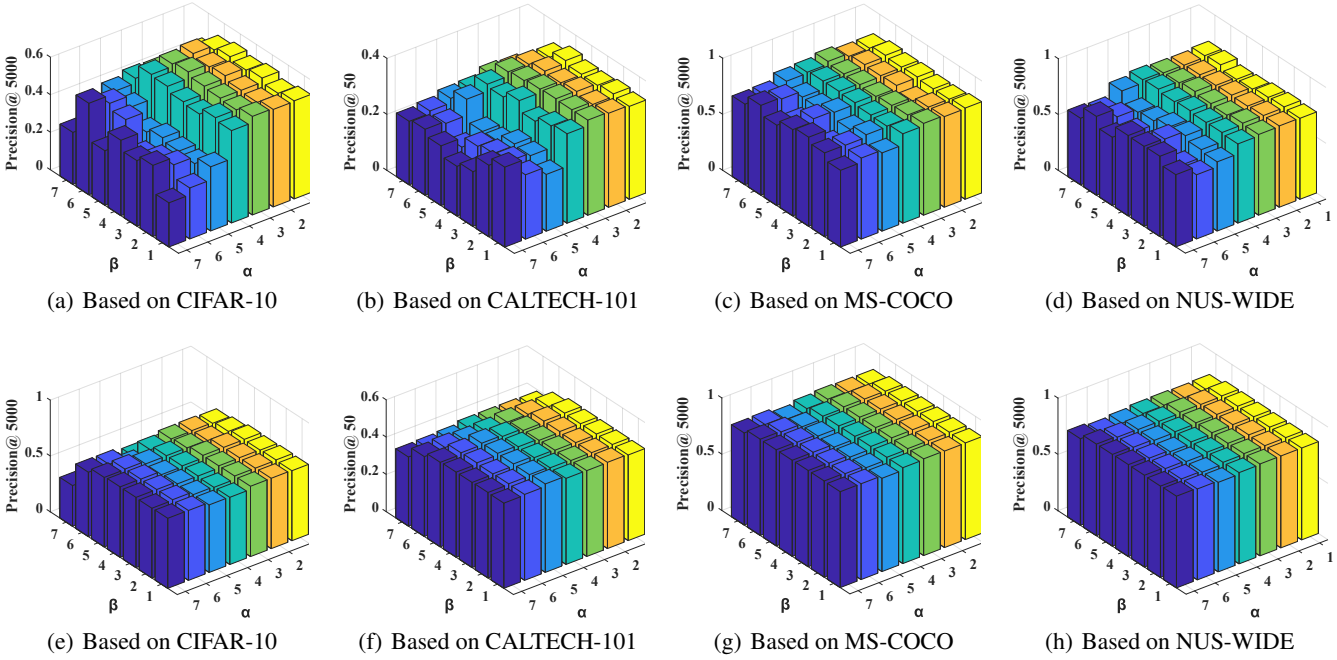
[3] http://cocodataset.org/.

[4] http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm.



(a) Based on CIFAR-10 (b) Based on CALTECH-101 (c) Based on MS-COCO (d) Based on NUS-WIDE

Figure 1: Performance in terms of the precision score based on four benchmark datasets.



(a) Based on CIFAR-10 (b) Based on CALTECH-101 (c) Based on MS-COCO (d) Based on NUS-WIDE

(e) Based on CIFAR-10 (f) Based on CALTECH-101 (g) Based on MS-COCO (h) Based on NUS-WIDE

Figure 2: Precision score with different settings of α and β, for four benchmark datasets. The hash code length is 8 in the first row (a-d), and 128 in the second (e-h).

CNN-F model [Chatfield *et al.*, 2014] to perform feature learning. For the CALTECH-101, each image was represented as a 512-dimension GIST feature. We performed ten runs of our method and averaged the performances, for comparison. As the experimental parameters, we empirically set $\alpha = \beta = \gamma = \lambda = 10^{-4}$ and $\delta = 2$.

3.2 Evaluation Metric

To evaluate the proposed method, we used an evaluation metric called mean average precision (mAP). mAP includes the mean of the average precision (AP) values obtained for the top retrieved samples. The AP is defined as

$$\text{AP} = \frac{1}{Z}\sum_{r=1}^{K}\text{Precision}(r)\sigma(r), \tag{21}$$

where Z is the number of relevant instances in the retrieved K samples. $\sigma(r)$=1, if the r-th instance is relevant to the query and $\sigma(r)$=0, otherwise. Precision(r) is the precision score for r samples; the precision score is defined as

$$\text{Precision} = \frac{N(\text{TP})}{N(\text{TP}) + N(\text{FP})}, \tag{22}$$

where $N(\cdot)$ is the number of the specific type. TP represents a true positive, whereas FP represents a false positive. In addition, Precision@n considers only the most relevant retrieved n samples.

Then, the mAP is defined as follows:

$$\text{mAP} = \frac{1}{M}\sum_{r=1}^{M}\text{AP}(i), \tag{23}$$

where M is the number of query samples and AP(i) is the average precision of the i-th instance.

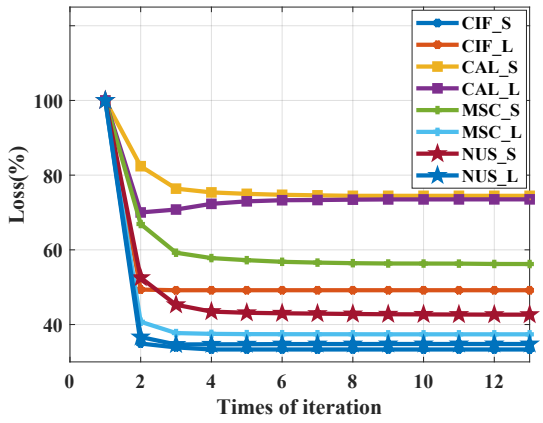


Figure 3: Convergence curves of the proposed SSLH for short and long-length hash codes for four datasets. In each curve, the loss of the proposed SSLH for the first iteration is considered to be 100%.

3.3 Experimental Results and Analysis

We compared the proposed SSLH with the following methods: locality-sensitive hashing (LSH) [Indyk and Motwani, 1998], spectral hashing (SH) [Weiss *et al.*, 2009], principle component analysis (PCA)-iterative quantization (PCA-ITQ)[Gong and Lazebnik, 2011], PCA-random rotation (PCA-RR) [Gong and Lazebnik, 2011], collective matrix factorization hashing (MFH) [Ding *et al.*, 2014], supervised discrete hashing (SDH) [Shen *et al.*, 2015], column sampling based discrete supervised hashing (COSDISH) [Kang *et al.*, 2016], fast supervised discrete hashing (FSDH) [Gui *et al.*, 2018], and scalable supervised discrete hashing (SSDH) [Luo *et al.*, 2018a]. LSH is a data-independent method. SH, PCA-ITQ, PCA-RR and MFH are unsupervised hashing methods, whereas SDH, COSDISH, FSDH, and SSDH are supervised ones. Only nondeep methods were considered for comparison because the proposed method is a linear-model-based method.

All the hyperparameters were initialized as suggested in the original methods. In the experiments, the short length, L, was slightly greater than the $log_2(c)$ value that was approximately 3.3, 6.7, 6.5, and 4.4 in the four datasets, respectively. We approximately set the short-length not greater than 10. In addition, the performances of long-length (e.g. 32 and 48) hash codes was also demonstrated in the experiments.

Table 1 lists the mAP values for each method, for single-label datasets, CIFAR-10 and CALTECH-101. The mAP performance of the SSLH was considerably better than those of the other methods for these two benchmark datasets, with short-length hash codes. Specifically, the SSLH exhibited more than 80% improvement with an extremely short-length hash code (4-bit in CIFAR-10). In addition, the proposed SSLH demonstrated considerable improvement with long-length hash codes.

In the CALTECH-101 dataset, whose training set is limited and the number of categories is large, most of the previous methods exhibited weak precision due to the controlled experimental setting. However, the proposed SSLH showed significant improvement for CALTECH-101 with short-length as well as long-length hash codes.

Table 2 depicts the mAP value for each method, for multilabel datasets, MS-COCO and NUS-WIDE. The proposed SSLH method outperformed the other methods for these two benchmark datasets. It demonstrated significant improvement with short-length hash codes, and considerable improvement with the long-length ones.

Substantial improvement can also be seen in Figure 1, in terms of the precision score, where the comparison between the proposed SSLH and the existing methods are depicted for different hash lengths. The SSLH exhibited considerably better performance, when the hash code length was shorter. With the increase in length, the improvement reduced, indicating that the proposed SSLH has a distinct advantage with short-length hash codes.

In order to verify the stability of the proposed method, we conducted experiments with different parameter settings. Figure 2 shows the precision score of the SSLH, when α and β ranged from 10^{-i} (i is the number on the coordinate axis); the SSLH method exhibited satisfactory stability and sensitivity with short-length hash codes, and distinguished stability and sensitivity with the long-length ones.

Figure 3 depicts the changes in the objective values achieved by the SSLH for four datasets, where CIF, CAL, MSC, and NUS represent CIFAR-10, CALTECH-101, MS-COCO, and NUS-WIDE, respectively. S and L indicate hash codes with lengths of 8 and 128, respectively. As the number of iterations increased, the objective values become small and stable, indicating that the SSLH appeared to reach convergence rapidly during training, thereby considerably reducing the time required for training.

4 Conclusion

In this study, we proposed a method called supervised short-length hashing (SSLH), wherein the semantic label information was leveraged by robust regression, while the information loss was restrained by mutual reconstruction. The proposed method achieved more stable and precise performances for short-length hash codes, while satisfactorily performing for long-length hash codes. Experiments conducted on four benchmark datasets indicated that the proposed method exhibits superior performance, compared to the other existing methods. In future, we will attempt to extend the proposed framework to nonlinear-based models.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61876098, 61671274, 61573219), China Postdoctoral Science Foundation (2016M592190), Shandong Provincial Key Research and Development Plan (2017CXGC1504), Special funds for distinguished professors of Shandong Jianzhu University and the Fostering Project of Dominant Discipline and Talent Team of Shandong Province Higher Education Institutions.

References

[Chatfield *et al.*, 2014] Ken Chatfield, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Return of the devil

in the details: Delving deep into convolutional nets. *arXiv preprint arXiv:1405.3531*, 2014.

[Chua *et al.*, 2009] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. 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*, page 48. ACM, 2009.

[Deerwester *et al.*, 1990] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407, 1990.

[Ding *et al.*, 2014] Guiguang Ding, Yuchen Guo, and Jile Zhou. Collective matrix factorization hashing for multimodal data. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 2083–2090, 2014.

[Fei-Fei *et al.*, 2007] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer vision and Image understanding*, 106(1):59–70, 2007.

[Gong and Lazebnik, 2011] Yunchao Gong and S Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 817–824, 2011.

[Gui and Li, 2018] Jie Gui and Ping Li. R 2 sdh: Robust rotated supervised discrete hashing. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1485–1493. ACM, 2018.

[Gui *et al.*, 2018] Jie Gui, Tongliang Liu, Zhenan Sun, Dacheng Tao, and Tieniu Tan. Fast supervised discrete hashing. *IEEE transactions on pattern analysis and machine intelligence*, 40(2):490–496, 2018.

[Hao *et al.*, 2017] Yanbin Hao, Tingting Mu, John Y Goulermas, Jianguo Jiang, Richang Hong, and Meng Wang. Unsupervised t-distributed video hashing and its deep hashing extension. *IEEE Transactions on Image Processing*, 26(11):5531–5544, 2017.

[He *et al.*, 2012] Ran He, Tieniu Tan, Liang Wang, and Wei-Shi Zheng. l 2, 1 regularized correntropy for robust feature selection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2504–2511. IEEE, 2012.

[Hoerl and Kennard, 1970] Arthur E. Hoerl and Robert W. Kennard. Ridge regression: Applications to nonorthogonal problems. *Technometrics*, pages 69–82, 1970.

[Indyk and Motwani, 1998] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[Jiang and Li, 2017] Qing Yuan Jiang and Wu Jun Li. Deep cross-modal hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[Kang *et al.*, 2016] Wang-Cheng Kang, Wu-Jun Li, and Zhi-Hua Zhou. Column sampling based discrete supervised hashing. In *AAAI*, pages 1230–1236, 2016.

[Krizhevsky and Hinton, 2009] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[Lawson and Hanson, 1995] Charles L Lawson and Richard J Hanson. *Solving least squares problems*, volume 15. Siam, 1995.

[Li *et al.*, 2015] Wu-Jun Li, Sheng Wang, and Wang-Cheng Kang. Feature learning based deep supervised hashing with pairwise labels. *arXiv preprint arXiv:1511.03855*, 2015.

[Lin *et al.*, 2014] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[Lin *et al.*, 2019] Mingbao Lin, Rongrong Ji, Hong Liu, Xiaoshuai Sun, Yongjian Wu, and Yunsheng Wu. Towards optimal discrete online hashing with balanced similarity. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2019.

[Liu *et al.*, 2007] Weifeng Liu, Puskal P Pokharel, and José C Príncipe. Correntropy: Properties and applications in non-gaussian signal processing. *IEEE Transactions on Signal Processing*, 55(11):5286–5298, 2007.

[Luo *et al.*, 2018a] Xin Luo, Ye Wu, and Xin-Shun Xu. Scalable supervised discrete hashing for large-scale search. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1603–1612. International World Wide Web Conferences Steering Committee, 2018.

[Luo *et al.*, 2018b] Yadan Luo, Yang Li, Fumin Shen, Yang Yang, Peng Cui, and Zi Huang. Collaborative learning for extremely low bit asymmetric hashing. *ArXiv e-prints*, 2018.

[Shen *et al.*, 2015] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. Supervised discrete hashing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 37–45, 2015.

[Shen *et al.*, 2017] Fumin Shen, Xin Gao, Li Liu, Yang Yang, and Heng Tao Shen. Deep asymmetric pairwise hashing. In *Proceedings of the 2017 ACM on Multimedia Conference*, pages 1522–1530. ACM, 2017.

[Wang *et al.*, 2018] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018.

[Weiss *et al.*, 2009] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.