ASYMMETRIC SPARSE HASHING

Xin Gao[†], *Fumin Shen*[†*], *Yang Yang*[†], *Xing Xu*[†], *Hanxi Li*[‡], *Heng Tao Shen*[†]

[†] Center for Future Media & School of Computer Science and Engineering,
University of Electronic Science and Technology of China, China [‡]JiangXi Normal University, China
gx_gaoxin@hotmail.com, fumin.shen@gmail.com, dlyyang@gmail.com
xing.xu@uestc.edu.cn, lihanxi2001@gmail.com, shenhengtao@hotmail.com

ABSTRACT

Learning based hashing has become increasingly popular because of its high efficiency in handling the large scale image retrieval. Preserving the pairwise similarities of data points in the Hamming space is critical in state-of-the-art hashing techniques. However, most previous methods ignore to capture the local geometric structure residing on original data, which is essential for similarity search. In this paper, we propose a novel hashing framework, which simultaneously optimizes similarity preserving hash codes and reconstructs the locally linear structures of data in the Hamming space. In specific, we learn two hash functions such that the resulting two sets of binary codes can well preserve the pairwise similarity and sparse neighborhood in the original feature space. By taking advantage of the flexibility of asymmetric hash functions, we devise an efficient alternating algorithm to optimize the hash coding function and high-quality binary codes jointly. We evaluate the proposed method on several large-scale image datasets, and the results demonstrate it significantly outperforms recent state-of-the-art hashing methods on large-scale image retrieval problems.

Index Terms— binary code, asymmetric hashing, pairwise affinity, sparse representation

1. INTRODUCTION

Hashing has become a popular and efficient technique for efficient large-scale image and video retrieval recently [1][2][3][4][5][6][7][8][9]. Hashing methods aim to map the original high-dimensional feature to compact binary code and preserve the semantic structure of the original feature in the Hamming space simultaneously. It turns out that using compact binary codes is much more efficient due to the employment of extremely fast Hamming distance computation. Hashing can be potentially used in recommendation systems [10], image classification [11][12], sketch based retrieval [13], action recognition [14], *etc*.

A common problem of hashing is to generate similarity-preserving hash functions, which encode similar inputs into nearby binary codes. Recently proposed hashing algorithms are focused on learning compact codes from data, a.k.a. *learning to hash*, which can be roughly grouped into two main categories: unsupervised and supervised hashing. Representative unsupervised hashing methods include Spectral Hashing (SH) [1], Iterative Quantization (ITQ) [15], Inductive Manifold Hashing (IMH) [16][17], *etc*. Supervised methods try to learn hash code by leveraging the supervised semantic information from data. Many famous methods fall in this category, such as Binary Reconstructive Embedding (BRE)[18], Kernel-Based Supervised Hashing (KSH) [19] and Supervised Discrete Hashing (SDH) [11] .

The appealing property of similarity-preserving hashing methods is to minimize the gap between the similarities given in the original space and in the hash coding space. In the hashing literature, pairwise similarity is widely adopted to make the distances of similarity pairs in the input and Hamming space as consistent as possible. However, most previous hashing approaches fail to explore the local neighborhood structure from data, which is essential for the similarity search. It is shown that data points in the high-dimensional feature space may lie on a low-dimensional manifold, where each point can be represented by a linear combination of its neighbors on the same manifold [16]. The linear representation is usually sparse and can well capture the local geometric structure of manifolds, which contains valuable information for nearest neighbor search.

Inspired by that, in this work, we aim to learn binary codes not only preserving the pairwise similarities as well as capturing the underlying neighborhood structure of local data, where the core idea is to construct asymmetric hash functions that simultaneously preserve pairwise affinity and the local neighborhood structure while mapping to the Hamming space. To achieve this, we optimize the asymmetric binary code inner product to minimize the Hamming distance on similar pairs and maximize on dissimilar pairs. Meanwhile, by analyzing the sparse representation of original feature

*Corresponding author: Fumin Shen. This work was supported in part by the National Natural Science Foundation of China under Project 61502081, Project 61673299, Project 61572108, Project 61602089 and Project 61632007, in part by the Fundamental Research Funds for the Central Universities under Project ZYGX2014Z007.

978-1-5090-6067-2/17/$31.00

space, we show how to capture the local linearity of manifold and preserve the learned linear structure in low-dimensional Hamming space. Our main contributions include:

- We address the importance of preserving the pairwise similarity and local geometric structure of data in the Hamming space. To this end, we propose a novel asymmetric sparse hashing (ASH) framework that jointly optimizes the Hamming affinity and encodes the geometric structure into binary codes.
- To generate better binary codes, we impose the orthogonality constraint on the projection vectors. To solve the associated binary optimization problem, we adopt an efficient algorithm, which alternatingly optimizes over binary codes and hash functions in an iterative way.
- We extensive evaluate the proposed ASH approach on the image retrieval tasks and the experiments show that our binary coding method significantly outperforms the state-of-the-art on several large-scale datasets.

2. THE PROPOSED APPROACH

In this section, we first briefly review the asymmetric hash function algorithm and then describe the detailed formulation of the proposed approach. An efficient alternating algorithm is introduced to solve the resulting problem.

2.1. Asymmetric Binary Code Learning

Inspired by the recent work [20], we dedicated to learning asymmetric hash function by dealing with Maximum Inner Product Search (MIPS) problem instead of ANN in this paper. Solving the MIPS problem has a critical practical impact. Finding hashing based algorithms for MIPS was considered hard. To solve this challenge, [21] shows that it is possible to relax the current LSH framework to allow asymmetric hash functions which can efficiently solve MIPS. Compared to the LSH which has a single hash function, [21] defines two hash functions $h(\cdot)$, $z(\cdot)$ to compute the inner product of the query and database point:

$$p = \underset{a \in S}{\operatorname{argmax}} \quad h(x)^T z(a) \tag{1}$$

where $h(\cdot)$, $z(\cdot)$ is the hash functions for the input query vector and data. ALSH is shown to be efficient to solve the MIPS problem both theoretically and empirically.

Motivated by ALSH, the asymmetric-inner product binary coding (AIBC) work [20] advocates the asymmetric form of hash function, which is learned from data rather than generated randomly. In [20], the Hamming affinity by the code inner product is optimized to match the ground truth data inner products as following:

$$\min \left\| h(X)^T z(A) - S \right\|^2 \tag{2}$$

where X and A denotes two sets of data points, S is the similarity matrix between A and X and $\|\cdot\|$ is the Frobenius norm. AIBC has been shown to achieve promising results for the MIPS problem. As aforementioned, however, AIBC fails to preserve the local neighborhood structure of data in the Hamming space, which is suboptimal for similarity search. Also, the involved optimization problem in AIBC differs from the one presented in this work, which further explores the orthogonality of hash functions and sparsity preservation.

2.2. Asymmetric sparse hashing

Let us first introduce some notations. Suppose we have two sets of points: $X = [x_1, x_2 \cdots x_m]$ and $A = [a_1, a_2 \cdots a_n]$, where $x_i \in \mathbb{R}^{d \times 1}$, $a_j \in \mathbb{R}^{d \times 1}$. Let denote by $\aleph_E(x)$ the set of neighborhood samples of x in dataset A. Let x_i be the input sample and A the dictionary, and we assume x_i can be represented as a sparse linear combination of the dictionary samples in $\aleph_E(x) \subset A$. That is, to relate x_i with its neighbors, we choose to solve the following objective to obtain its linear representation coefficients in terms of A, which contain the geometric structure information around x_i:

$$\begin{aligned} \min_{c_i} \quad & \frac{1}{2}\|x_i - Ac_i\|_2^2 + \lambda \|c_i\|_1 \\ \text{s.t.} \quad & c_{i,j} = 0 \quad \text{if} \quad a_j \notin \aleph_E(x_i) \end{aligned} \tag{3}$$

where $c_i = [c_{i,1}, \cdots, c_{i,n}]^\top$ is the coefficient vector for sample x_i. We use the neighborhood constraint to favor the locally sparse representation, therefore x_i can be represented as a sparse linear combination of its neighbors. Problem (3) can be solved with various efficient sparse coding algorithms, where we apply the Homotopy algorithm [22] to solve this ℓ_1-regularized regression problem.

Let $C = [c_1, c_2, \cdots, c_m] \in \mathbb{R}^{n \times m}$ be the sparse matrix composed of the sparse representation of the dataset X, which encodes the local neighbor structure between X and A. We would like to preserve this in Hamming space while learning binary codes. We denote $B = h(X)$, $B = [b_1, b_2, \cdots, b_m] \in \{1, -1\}^{r \times m}$ is the binary code matrix of dataset X; $Z = z(A)$, $Z = [z_1, z_2, \cdots, z_n] \in \{1, -1\}^{r \times n}$ is the binary code matrix of dataset A. We have the following optimization problem:

$$\begin{aligned} \min_{b_i, z_j} \quad \sum_i \left\| b_i - \sum_j c_{ij} z_j \right\| &= \sum \|h(x_i) - C_{\cdot i} z(A)\|_2^2 \\ &= \|h(X) - Cz(A)\|^2. \end{aligned} \tag{4}$$

which $C_{\cdot i}$ is the i-th column of matrix C. Minimizing the above objective function can well preserve the locally linear structure in the Hamming space. The local linear hashing (LLH) work [23] also studies a similar problem as (4) for hashing. However LLH is different from our work by the following aspects: 1) we are focusing on the asymmetric hashing

for MIPS problem while LLH is designed with a symmetric setting; 2) our model learns orthogonal hash functions while LLH does not; 3) we optimize binary codes in a discrete way, while a relaxed problem (minimizing quantization loss with ITQ) is adopted.

By incorporating the pairwise similarity and local neighborhood structure preservation into the asymmetric binary coding framework, we get the following optimization problem:

$$\min_{h,z} \quad \left\|h(X)^T z(A) - S\right\|^2 + \|h(X) - Cz(A)\|_2^2. \tag{5}$$

In addition, we consider to minimize the quantization loss for binary embedding in our objective function. Let $B = h(X) = \text{sgn}(W^T X)$, $Z = z(A) = \text{sgn}(R^T A)$, where $W, R \in \mathbb{R}^{d\times r}$. The final optimization problem can be rewritten as

$$\begin{aligned} \min_{B,W,Z,R} \quad & \left\|B^T Z - S\right\|^2 + \|B - CZ\|^2 \qquad (6) \\ & + \lambda\left\|B - W^T X\right\|^2 + \beta\left\|Z - R^T A\right\|^2 \\ \text{s.t.} \quad & B \in \{-1,1\}^{r\times m}, Z \in \{-1,1\}^{r\times n}, \qquad (7) \\ & W^T W = I, R^T R = I \end{aligned}$$

The projection matrix W and R are considered to be orthogonal in this work. The reason for the orthogonal transformation is that it can preserve the Euclidean distance between points, distribute the variance more evenly across the dimensions and finally generate maximally uncorrelated compact binary codes [1].

2.3. Optimization

Apparently, the above joint optimization problem (6) is non-convex, non-smooth and generally an NP-hard problem due to the binary constraint $b_i \in \{-1+1\}^r$. To find a feasible solution, we solve the original problem by iteratively optimizing its two sub-problems. In other words, we solve this problem in an alternating way: updating one variable with others fixed.

By taking advantage of the flexibility of the asymmetric hash function, we adopt to solve W and R in an alternative way. In practice, we chose to solve W and B while fixing Z and R first. We initialize R by PCA projection, and then Z can be initialize by $Z = \text{sgn}(R^T A)$. The two alternating steps are described in detail below: Equation (6) can be reduced to the following optimization problem w.r.t. B and W:

$$\min_{B,W} \quad \left\|B^T Z - S\right\|^2 + \|B - CZ\|^2 + \lambda\left\|B - W^T X\right\|^2 \tag{8}$$

$$\text{s.t.} \quad B \in \{-1,1\}^{r\times m}, W^T W = I. \tag{9}$$

(i) Fix B and update W. With B fixed, the first term is ignored in this sub-problem, so the problem can be shown as equivalent to:

$$\begin{aligned} \min_{W} \quad & \|B - WX\|^2 \qquad (10) \\ \text{s.t.} \quad & W^T W = I. \end{aligned}$$

Problem (10) is known as the orthogonal procrustes problem, and is recently widely involved in learning orthogonal projections. This procrustes problem can be solved as follows: firstly, we take the SVD of the matrix XB^T as $XB^T = U\Sigma V^T$, then let U_r be the first r singular vectors of U and finally get $W = U_r V^T$.

(ii) Fix W and update B. We have the following sub-problem:

$$\min_{B} \quad \left\|B^T Z - S\right\|^2 + \|B - CZ\|^2 + \lambda\left\|B - W^T X\right\|^2 \tag{11}$$

$$\text{s.t.} \quad B \in \{-1,1\}^{r\times m}.$$

This problem is hard to solve because of the binary constraint in matrix B. A natural idea to solve this problem is to relax the binary constraint to find a continuous embedding and then binarize it. But this approximate solution may cause large quantization error and consequently result in sub-optimal performance. This is especially the case when learning long hash codes. Following [11], we adopt the discrete cyclic coordinate descent method to solve this sub-problem.

We expand (11) and rewrite it as:

$$\begin{aligned} \min_{B} \quad & \left\|B^T Z\right\|^2 - 2Tr(B^T Q) \qquad (12) \\ s.t. \quad & B \in \{-1,1\}^{r\times m}, \end{aligned}$$

where $Q = ZS^T + CZ + \lambda W^T X$. According to [11], we choose to learn one row of B in each iteration while other rows fixed. Let b be the l^{th} row of B, $\widetilde{B}$ be the rest of B omitting the b; let z be the l^{th} row of Z, $\widetilde{Z}$ be the rest of Z omitting the z; let q be the l^{th} row of Q, $\widetilde{Q}$ be the rest of the Q omitting the q. Thus the original optimization problem reduces to:

$$\begin{aligned} \min_{b} \quad & (z^T \widetilde{B}^T \widetilde{Z} - q^T)b \qquad (13) \\ \text{s.t.} \quad & b \in \{-1,1\}^r. \end{aligned}$$

Finally, we have the solution:

$$b = \text{sgn}(q - \widetilde{Z}^T \widetilde{B} z). \tag{14}$$

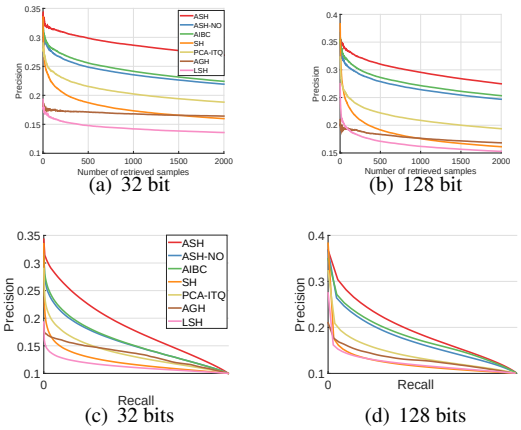
Fix B, W and update Z, R. In the same way, with B and W fixed, (6) can be written as

$$\min_{Z,R} \quad \left\|B^T Z - S\right\|^2 + \|B - CZ\|^2 + \beta\left\|Z - R^T A\right\|^2 \tag{15}$$

$$\text{s.t.} \quad Z \in \{-1,1\}^{r\times n}, R^T R = I. \tag{16}$$

Table 1. Results in terms of mean average precision (MAP) and mean precision of the top 500 retrieved neighbors (Precision@500) of the compared methods with 32, 64, 96 and 128 bits on the **CIFAR-10** dataset .

Method	MAP				Precision@500			
	32-bit	64-bit	96-bit	128-bit	32-bit	64-bit	96-bit	128-bit
LSH	0.1209	0.1261	0.1279	0.1371	0.1504	0.1678	0.1800	0.1914
AGH	0.1581	0.1500	0.1442	0.1411	0.2930	0.2955	0.3066	0.3072
SH	0.1252	0.1254	0.1250	0.1249	0.1873	0.1880	0.1892	0.1912
PCA-ITQ	0.1490	0.1447	0.1479	0.1520	0.2154	0.2040	0.2167	0.2222
AIBC	0.1724	0.1925	0.1937	0.2051	0.2581	0.2626	0.2754	0.2841
ASH-NO	0.1705	0.1943	0.1987	0.1967	0.2486	0.2754	0.2741	0.2788
ASH	**0.2143**	**0.2133**	**0.2157**	**0.2175**	**0.2986**	**0.3039**	**0.3100**	**0.3102**



(a) 32 bit (b) 128 bit

(c) 32 bits (d) 128 bits

Fig. 1. (Top) Precision@2000 retrieved samples and (Bottom) precision-recall curves on CIFAR-10 for 32 and 128 bits respectively.

We can obtain the orthogonal projection R and the binary code matrix Z in the similar way as solving (8). We iteratively solve (8) and (15) and the algorithm converges after several iterations. The proposed Asymmetric Sparse Hashing algorithm is summarized in **Algorithm 1**.

Algorithm 1 Asymmetric Sparse Hashing (ASH)

Input: Training data X and A ; similarity matrix S; binary code length L; parameters λ and β.

Output: hash function $h(x)$ and $z(a)$

1: **for** $i = 1; k \leq n; i++$ **do**
2: Compute C_i according to (3);
3: **end for**
4: Initialize R by PCA projection;
5: Initialize Z by $Z = sgn(R^T A)$;
6: **repeat**
7: update B and W by solving (8);
8: update Z and R solving (15);
9: **until** converge or reach maximum iterations

3. EXPERIMENTS

In this section, we extensively evaluate the proposed method with comparison to several state-of-the-art algorithms on the image retrieval task. We test our method on three image datasets:CIFAR-10, SUN397 and ImageNet. Since the proposed asymmetric sparse hashing (ASH) is unsupervised, we take into comparison several state-of-the-art unsupervised algorithms, including LSH [24], AGH [25], ITQ [15], SH [1] and AIBC [20]. For AGH, we use k-means to generate 1,000 cluster centers for anchor. We use the available codes and suggested parameters in the original papers of these approaches. Despite the original algorithm of ASH, we also evaluate ASH without orthogonality constraint (denoted by ASH-NO). For our methods, we set the data matrix X as the whole training data and generate A by the 10,000 randomly selected points from training set; we set the maximum iteration number as 10 for all the experiments; we empirically set the parameters $\lambda = 100$ and $\beta = 100$; the similarity matrix is defined as follows. First, we compute the Euclidean distance between each point in X and its k_{th} nearest neighbor, and then set the threshold as the average distance to round S to be 1 or 0. Then the semantic neighbors are decided if the euclidean distance is less than the threshold. We set k as 500 for SUN397 and CIFAR-10 and 1000 for ImageNet.

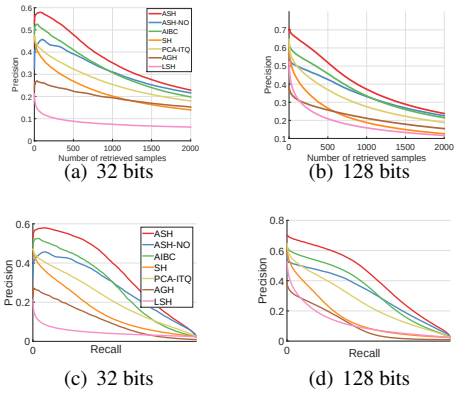
We use the following evaluation metric to measure the performance: hashing ranking performance means average precision (MAP) , mean precision of the top 500 retrieved samples (Precision@500) and hashing lookup precision of Hamming distance 2 (HD2 Precision). We also show the precision of top 2000 and precision-recall curves of three datasets.

Result on CIFAR-10. The CIFAR-10[1] is a labeled subset of 80-million tiny images collection which consists of 60000 32×32 color images in 10 classes. The comparative results on CIFAR-10 are reported in Table 1. As can be seen, our ASH achieves superior result with all code lengths in MAP, precision@500 on this dataset to all other methods. ASH consistently performs better than AIBC though the

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

Table 2. Results in terms of mean average precision (MAP) and mean precision of the top 500 retrieved neighbors (Precision@500) of the compared methods for 32, 64, 96 and 128 bit respectively on the **SUN397** .

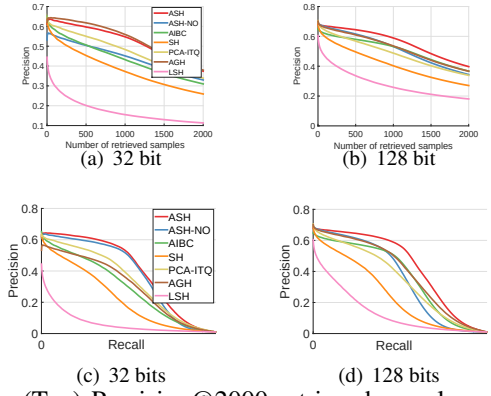
Method	MAP				Precision@500			
	32-bit	64-bit	96-bit	128-bit	32-bit	64-bit	96-bit	128-bit
LSH	0.0572	0.0940	0.1280	0.1486	0.0883	0.1432	0.1842	0.2086
AGH	0.3149	0.2919	0.2600	0.2372	0.3699	0.3620	0.3410	0.3620
SH	0.2048	0.2082	0.2000	0.1926	0.2687	0.2779	0.2733	0.2674
PCA-ITQ	0.2781	0.3073	0.3164	0.3183	0.3231	0.3546	0.3643	0.3660
AIBC	0.3601	0.3987	0.4178	0.4207	0.4098	0.4465	0.4593	0.4575
ASH-NO	0.3617	0.3680	0.3929	0.3954	0.3917	0.3968	0.4238	0.4238
ASH	**0.4466**	**0.4849**	**0.4961**	**0.5013**	**0.4782**	**0.5081**	**0.5162**	**0.5193**



(a) 32 bits (b) 128 bits

(c) 32 bits (d) 128 bits

Fig. 2. (Top) Precision@2000 retrieved samples and (Bottom) precision-recall curves on SUN397 with 32 and 128 bits respectively.



(a) 32 bit (b) 128 bit

(c) 32 bits (d) 128 bits

Fig. 3. (Top) Precision@2000 retrieved samples and (Bottom) precision-recall curves on ImageNet with 32 and 128 bits respectively.

advantage becomes smaller as the code size increases. We can clearly see that ASH outperforms ASH-NO in all situations, which demonstrates the advantage of the unrelated binary codes for retrieval. Among the compared methods, the data-independent LSH improves the performance as the code size increases, and it even surpasses the data-dependent method SH at long code size. This behavior may due to the theoretic convergence guarantee of LSH with the long hash codes. For precision@500, our method ASH has a small but consistent advantage over AIBC. The precision curves with top 2000 returned samples and precision-recall curves are reported in Figurer 1. The results with 32 and 128 bits confirm the performance gains of ASH in Table 1.

Result on SUN397. This dataset [26] contains 397 classes about 108K images. Each image is represented by a 1600-dimensional feature vector. Table 2 shows the results on SUN397 dataset. Consistent with the results on CIFAR-10, our method ASH clearly outperforms the compared methods in terms of both MAP and precision@500. For instance, with 128 bits, ASH obtains 50.13% MAP which are higher than the second best result (by AIBC) by 8.06%. The MAPs of AGH and KSH suffer from performance deterioration with the increasing code lengths. Among other methods, AIBC achieves best results, which implies that directly learning the binary codes without relaxations is preferable than the continuous solution to achieve approximate binary solution. Fig. 2 (top) shows the comparison in terms of precision of top 2000 samples, where we can see that ASH ranks first with 32 and 128 bits. Fig. 2 (bottom) represents the precision-recall curves for our method and compared methods, which corresponds to the trend in Table 2.

Result on ImageNet. As a subset of ImageNet [27], the large dataset ILSVRC 2012 contains over 1.2 million image of totally 1,000 categories. We use the provided training dataset as the retrieval dataset and 50,000 from the training set as the query set. We use the 4096-dimensional features extracted by the convolution neural networks (CNN) model. Table 3 compares all the methods in MAP and precision@500 on the ImageNet dataset. For MAP, we can see that both ASH and AGH achieve outstanding performance on this dataset. However, AGH suffers from performance deterioration at high code length. For instance, when code length is 128 bits, our method achieves 54.37% and is higher than AGH 45.65% by 8.72%. Similar results are observed in terms of precision@500. These results demonstrate the ability of ASH in preserving pairwise affinity and capturing local neighborhood structure.

Table 3. Results in terms of mean average precision (MAP) and mean precision of the top 500 retrieved neighbors (Precision@500) of the compared methods for 32, 64, 96 and 128 bit respectively on the **ImageNet** dataset.

Method	MAP				Precision@500			
	32-bit	64-bit	96-bit	128-bit	32-bit	64-bit	96-bit	128-bit
LSH	0.0592	0.1136	0.1555	0.1903	0.1209	0.2183	0.2835	0.3326
AGH	**0.3873**	0.4547	0.4999	0.4565	**0.4653**	0.5430	0.6216	0.6155
SH	0.2418	0.3066	0.3243	0.3310	0.3642	0.4513	0.4813	0.4956
PCA-ITQ	0.3014	0.3842	0.4234	0.4416	0.4115	0.5052	0.5479	0.5679
AIBC	0.3067	0.3575	0.3275	0.3607	0.3932	0.4270	0.4008	0.4402
ASH-NO	0.2978	0.3679	0.4509	0.4981	0.3847	0.4378	0.4509	0.5518
ASH	0.3748	**0.4874**	**0.5011**	**0.5437**	0.4637	**0.5920**	**0.6344**	**0.6526**

Table 4. Results in terms of Hamming distance 2 Precision of the compared methods on the **CIFAR-10**, **SUN397** and **ImageNet** dataset at 32 bit.

Method	LSH	AGH	SH	PCA-ITQ	AIBC	ASH-NO	ASH
CIFAR-10	0.1650	0.2160	0.2262	0.1632	0.1452	0.1624	**0.2350**
SUN397	0.0437	0.4852	0.2472	0.3813	0.4448	0.4055	**0.5177**
ImageNet	0.0666	0.5015	0.4029	0.4425	0.1827	**0.5447**	0.4814

4. CONCLUSION

In this paper, we presented a novel method for learning asymmetric hash functions, which was capable of preserving pairwise affinity and local structure in Hamming space jointly. An efficient algorithm was designed to optimize the model in alternating manners and finally we obtained high-quality binary code and hash function. Our experiments on large scale image datasets demonstrated the superiority of our method over state-of-the-art methods.

5. REFERENCES

[1] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: NIPS, 2008, pp. 1753–1760.

[2] J. Song, Y. Yang, X. Li, Z. Huang, Y. Yang, Robust hashing with local models for approximate similarity search, IEEE TCYB 44 (7) (2014) 1225–1236.

[3] J. Song, Y. Yang, Y. Yang, Z. Huang, H. T. Shen, Inter-media hashing for large-scale retrieval from heterogeneous data sources, in: SIGMOD, 2013, pp. 785–796.

[4] J. Song, Y. Yang, Z. Huang, H. T. Shen, R. Hong, Multiple feature hashing for real-time large scale near-duplicate video retrieval, in: ACM Multimedia, 2011, pp. 423–432.

[5] Y. Yang, Y. Luo, W. Chen, F. Shen, J. Shao, H. T. Shen, Zero-shot hashing via transferring supervised knowledge, in: ACM Multimedia, 2016, pp. 1286–1295.

[6] H. Zhang, N. Zhao, X. Shang, H.-B. Luan, T.-s. Chua, Discrete image hashing using large weakly annotated photo collections., in: AAAI, 2016, pp. 3669–3675.

[7] Y. Yang, Z.-J. Zha, Y. Gao, X. Zhu, T.-S. Chua, Exploiting web images for semantic video indexing via robust sample-specific loss, IEEE TMM 16 (6) (2014) 1677–1689.

[8] H. Zhang, M. Wang, R. Hong, T.-S. Chua, Play and rewind: Optimizing binary representations of videos by self-supervised temporal hashing, in: ACM Multimedia, 2016, pp. 781–790.

[9] L. Liu, M. Yu, F. Shen, L. Shao, Discretely coding semantic rank orders for image hashing, in: CVPR, 2017.

[10] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, T.-S. Chua, Discrete collaborative filtering, in: SIGIR, 2016, pp. 325–334.

[11] F. Shen, C. Shen, W. Liu, H. T. Shen, Supervised discrete hashing, in: CVPR, 2015, pp. 37–45.

[12] Y. Yang, H. Zhang, M. Zhang, F. Shen, X. Li, Visual coding in a semantic hierarchy, in: ACM Multimedia, 2015, pp. 59–68.

[13] L. Liu, F. Shen, Y. Shen, X. Liu, L. Shao, Deep sketch hashing: Fast free-hand sketch-based image retrieval, in: CVPR, 2017.

[14] J. Qin, L. Liu, L. Shao, F. Shen, B. Ni, J. Chen, Y. Wang, Zero-shot action recognition with error-correcting output codes, in: CVPR, 2017.

[15] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval, TPAMI 35 (12) (2013) 2916–2929.

[16] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang, H. T. Shen, Hashing on nonlinear manifolds, TIP 24 (6) (2015) 1839–1851.

[17] F. Shen, C. Shen, Q. Shi, A. Van Den Hengel, Z. Tang, Inductive hashing on manifolds, in: CVPR, 2013, pp. 1562–1569.

[18] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, in: NIPS, 2009, pp. 1042–1050.

[19] W. Liu, J. Wang, R. Ji, Y. Jiang, S. Chang, Supervised hashing with kernels, in: CVPR, 2012, pp. 2074–2081.

[20] F. Shen, W. Liu, S. Zhang, Y. Yang, H. T. Shen, Learning binary codes for maximum inner product search, in: ICCV, 2015, pp. 4148–4156.

[21] A. Shrivastava, P. Li, Asymmetric LSH (ALSH) for sublinear time maximum inner product search, in: NIPS, 2014, pp. 2321–2329.

[22] H. Lee, A. Battle, R. Raina, A. Y. Ng, Efficient sparse coding algorithms, in: NIPS, 2006, pp. 801–808.

[23] G. Irie, Z. Li, X.-M. Wu, S.-F. Chang, Locally linear hashing for extracting non-linear manifolds, in: CVPR, 2014, pp. 2115–2122.

[24] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: VLDB, 1999, pp. 518–529.

[25] W. Liu, J. Wang, S. Kumar, S. Chang, Hashing with graphs, in: ICML, 2011, pp. 1–8.

[26] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, A. Torralba, SUN database: Large-scale scene recognition from abbey to zoo, in: CVPR, 2010, pp. 3485–3492.

[27] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: A Large-Scale Hierarchical Image Database, in: CVPR, 2009, pp. 248–255.