Supervised Hashing with Pseudo Labels for Scalable Multimedia Retrieval

Jingkuan Song
University of Trento
jingkuan.song@unitn.it

LianLi Gao,
University of Electronic Science and Technology of China
lianli.gao@uestc.edu.cn

Yan Yan
University of Trento
yan@disi.unitn.it

Dongxiang Zhang
National University of Singapore
zhangdo@comp.nus.edu.sg

Nicu Sebe
University of Trento
sebe@disi.unitn.it

ABSTRACT

There is an increasing interest in using hash codes for efficient multimedia retrieval and data storage. The hash functions are learned in such a way that the hash codes can preserve essential properties of the original space or the label information. Then the Hamming distance of the hash codes can approximate the data similarity. Existing works have demonstrated the success of many supervised hashing models. However, labeling data is time and labor consuming, especially for scalable datasets. In order to utilize the supervised hashing models to improve the discriminative power of hash codes, we propose a Supervised Hashing with Pseudo Labels (SHPL) which uses the cluster centers of the training data to generate pseudo labels, based on which the hash codes can be generated using the criteria of supervised hashing. More specifically, we utilize linear discriminant analysis (LDA) with trace ratio criterion as a showcase for hash functions learning and during the optimization, we prove that the pseudo labels and the hash codes can be jointly learned and iteratively updated in an unified framework. The learned hash functions can harness the discriminant power of trace ratio criterion, and thus can achieve better performance. Experimental results on three large-scale unlabeled datasets (i.e., SIFT1M, GIST1M, and SIFT1B) demonstrate the superior performance of our SHPL over existing hashing methods.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: INFORMATION STORAGE AND RETRIEVAL—*Search process*

Keywords

Hashing; multimedia retrieval; pseudo labels

1. INTRODUCTION

Recently, learning-based hashing methods [10, 15, 5, 7, 11, 12] have become the mainstream for scalable ANN due to their compact binary representation and efficient Hamming distance calculation. Such approaches map data points to compact binary codes through a hash function. Hashing methods can be categorized as unsupervised and supervised. The unsupervised learning of the hash functions is usually based on the criterion of preserving important properties of the training data points. Typical approaches preserve the consistency property (i.e., the similarity of binary codes should be consistent with that of the original data points) [16, 5], the similarity alignment property (i.e., the Hamming distance of the binary code should approximate the Euclidean distance of the original data points) [6], the order preserving property (i.e., the order of a reference data item computed from the original space and the Hamming space should be aligned) [9], etc. A fundamental limitation of property-preserving hashing methods is that different methods try to preserve varied properties according to the applications, and thus the performance may degrade when a specifically designed hashing method is applied to another application. On the other hand, supervised hashing is designed to preserve some label-based similarity [7, 2, 13]. For example, Strecha *et al.* [13] developed a supervised hashing which maximizes the between-class Hamming distance and minimizes the within-class Hamming distance. [7, 2] proposed learning the hash codes which can approximate the pairwise label similarity. The performance for supervised hashing methods is usually significantly superior to unsupervised methods. However, the supervised information is scarce, especially for scalable datasets.

Inspired by this, in this paper, we propose a Supervised Hashing with Pseudo Labels (SHPL) which makes use the self-generated cluster centers of the training data as supervision information to improve the effectiveness of the hash codes. It is worth highlighting the following contributions:

- We propose a general framework to learn hashing code in a supervised way to improve the effectiveness of hashing methods by using some pseudo labels. We then prove that the pseudo supervision information and the hash codes can be jointly learned and iteratively updated in an unified framework.
- We further integrate the pseudo labels strategy to unsupervised hashing model, which can potentially improve the search accuracy of hash codes.

- We successfully show it to work on a huge dataset SIFT1B (1 billion data points). Experimental results on three popular datasets show superior performance compared to other unsupervised hashing methods.

2. SUPERVISED HASHING WITH PSEUDO LABELS

In this section, we introduce our algorithm. We first introduce the notations which will be used in the rest of the paper.

Suppose there is a dataset $X = [x_1, ..., x_N] \in \mathbb{R}^{M\times N}$, where N is the number of data points and M is the dimensionality of each x_i. Their hash codes are $Y = [y_1, ..., y_N] \in \mathbb{R}^{L\times N}$ where L is the code length for each y_i. For supervised hashing methods, different criteria are utilized to preserve the label information [13, 14, 8]. Based on these criteria to be preserved, different objective functions are given. Without loss of generality, we utilize LDA with trace ratio criterion (similar to LDAH [13]) as a showcase for hash functions learning and demonstrate how it is incorporated into our framework. Note that some other criterion like label-similarity preserving [14, 8] can also be applied to our framework.

The loss function for LDA trace ratio based hashing method is:

$$\ell_{SHPL}(Y) = \frac{tr(S_b)}{tr(S_w)}, \quad s.t.\ Y \in \{0,1\}^{N\times L} \tag{1}$$

where S_w represents within-class scatter matrix, and S_b is the between-class scatter matrix. They are defined as:

$$\begin{array}{l} S_w = \sum\limits_{k=1}^{C} \sum\limits_{x_i\in\Omega_k} (y_i - c_k)(y_i - c_k)^T \\ S_b = \sum\limits_{k=1}^{C} n_k (c_k - \overline{y})(c_k - \overline{y})^T \end{array} \tag{2}$$

where y_i is the corresponding hash code for a date point x_i, Ω_k indicates the data points in the k-th class and c_k is the mean of hash codes in the k-th class. $\overline{y}$ is the mean of all hash codes Y. We also define the total scatter matrix S_t as:

$$S_t = S_b + S_w = \sum_{i=1}^{N} (y_i - \overline{y})(y_i - \overline{y})^T \tag{3}$$

Suppose the data has been centralized, i.e., $\overline{x} = 0$. We denote the pseudo labels as F, and we define a cluster centroid matrix C to include the centroid vector of the hash codes in each class as $C = [c_1, ..., c_K]$. We use a linear hash function, i.e., $y_i = sgn\left(W^T x_i\right)$. This objective function is intractable and we follow [8, 16] to apply the spectral relaxation trick to drop the sign functions. Then $\overline{y} = 0$. Thus S_b, S_w and S_t can be rewritten as:

$$\begin{array}{l} S_w = \left(W^T X - CF^T\right)\left(W^T X - CF^T\right)^T \\ S_b = CF^T F C^T \\ S_t = W^T X X^T W \end{array} \tag{4}$$

Then the objective function becomes:

$$\ell_{SHPL}\left(W, W^T W = I\right) = \frac{tr\left(CF^T FC^T\right)}{tr\left(\left(W^T X - CF^T\right)\left(W^T X - CF^T\right)^T\right)} \tag{5}$$

Because $S_t = S_w + S_b$, optimizing (1) is equivalent to optimize $\frac{tr(S_b)}{tr(S_t)}$. The problem is that we do not know the class labels F. Then, the final objective function for our SHPL becomes:

$$\begin{array}{l} \ell_{SHPL}(W, F, C) = \frac{tr\left(CF^T FC^T\right)}{tr\left(W^T X X^T W\right)} \\ s.t. \left\{ \begin{array}{l} F \in \{0,1\}^{N\times K} \\ \|f_i\|_1 = 1 \\ W^T W = I \end{array} \right. \end{array} \tag{6}$$

where $\|\cdot\|_1$ is the l_1 norm. The second constraint $\|f_i\|_1 = 1$ requires that each x_i belongs to a single class.

2.1 Solution

There are three unknown variables in (6), namely W, C and F. Using the class indicator matrix F, we can represent each cluster centroid c_k as:

$$c_k = \frac{1}{size\left(\prod(k)\right)} \sum_{y_i\in\prod(k)} y_i \tag{7}$$

where $\prod(k)$ indicates the set of data points in class k. Then, C can be reformulated in a matrix form:

$$C = W^T X F \left(F^T F\right)^{-1} \tag{8}$$

Then, (6) becomes:

$$\begin{array}{l} \ell_{SHPL}(W, F) = \frac{tr\left(W^T X F\left(F^T F\right)^{-1} F^T X^T W\right)}{tr\left(W^T X X^T W\right)} \\ s.t. \left\{ \begin{array}{l} F \in \{0,1\}^{N\times K} \\ \|f_i\|_1 = 1 \\ W^T W = I \end{array} \right. \end{array} \tag{9}$$

We utilize coordinate descent to optimize (9). We firstly fix F and update W, and then fix update W by fixing F. They are updated iteratively until convergence. The solution is illustrated in Algorithm (1).

Update W: Given F, obviously solving problem (9) is to minimize the trace ratio LDA w.r.t. W:

$$\ell\left(W, W^T W = I\right) = \frac{tr\left(W^T X F\left(F^T F\right)^{-1} F^T X^T W\right)}{tr\left(W^T X X^T W\right)} \tag{10}$$

which can be directly solved with the generalized eigenvalue decomposition (GEVD) method:

$$XF\left(F^T F\right)^{-1} F^T X^T w_l = \lambda_l X X^T w_l \tag{11}$$

where λ_l is the l-th largest eigenvalue of the GEVD with the corresponding eigenvector w_l, and w_l constitutes the l-th column vector of the matrix W.

When W is fixed, $tr(W^T X X^T W)$ is irrelevant to F. Thus, we need to maximize the following problem *w.r.t* F:

$$\begin{array}{l} \ell(F) = tr\left(W^T X F\left(F^T F\right)^{-1} F^T X^T W\right) \\ s.t. \left\{ \begin{array}{l} F \in \{0,1\}^{N\times K} \\ \|f_i\|_1 = 1 \end{array} \right. \end{array} \tag{12}$$

It is still difficult to solve due to the intractable constraints in (12). Because $tr(W^T X X^T W)$ is a constant now (W is fixed), maximizing between-class distance in problem (12) is equivalent to minimizing within-class distance. Problem (12) is equivalent to the following problem:

$$\ell(F) = \left\|W^T X - CF^T\right\|_F^2, s.t. \left\{ \begin{array}{l} F \in \{0,1\}^{N\times K} \\ \|f_i\|_1 = 1 \end{array} \right. \tag{13}$$

Problem (13) can be easily solved by alternating optimization, i.e., iteratively optimizing C when F is fixed and optimizing F when C is fixed.

Update C: Each cluster center c_k is the mean of all data points in the class k. C can be updated using (7).

Update F: Since each data point belongs to one class, x_i is assigned to its closest cluster center c_k. Therefore, f_i is a column vector with its k-th element being 1 and others being 0.

After we get W, the hash codes for x_i can be generated by $sgn(W^T x_i)$.

Algorithm 1 Solution for the SHPL

```
Input: Initialized $F$;
Output: $F$, $W$;
1: repeat
2:   Fix $F$, update $W$ according to Eqn.(11);
3:   repeat
4:     Fix $W$ and $F$, update $C$ according to classical K-means,
       each $c_k$ is the mean of the $y_i$ within the same class;
5:     Fix $W$ and $C$, update $F$;
6:   until convergence or max iteration is reached.
7: until convergence or max iteration is reached.
8: return $F$, $W$;
```

3. PSEUDO LABELS FOR UNSUPERVISED HASHING

The pseudo label strategy can also be integrated to unsupervised hashing models to improve the performance of hash codes. A typical unsupervised hashing preserving global similarity can be formulated as the following loss function:

$$\ell(Y) = \sum_{ij} w_{ij} \|y_i - y_j\|_H, \; s.t. \; Y^T Y = I, \; y_i \in \{0,1\}^L \tag{14}$$

where y_i is the hash code for the data point x_i, L is the hash code length, and W is the affinity matrix [1].

Suppose we also have pseudo labels f_i for each data point x_i. Suppose there are K classes and each data point belongs to a single class. Then we can have K cluster centers. It is reasonable to assume that data points from the same class should have similar hash codes. Based on these clustering centers, the class label f_i for each x_i can be updated as:

$$k_i^* = \arg \min_{k \in \{1,\ldots,K\}} \|\mathbf{x_i} - \mathbf{c}_k\|_2^2 \tag{15}$$

where c_k is the cluster center for x_i. The problem is that we do not have the cluster centers. Actually, by taking each bucket with the same hash codes as a class, we can build a connection between y_i and k_i as:

$$y_i = b(k_i), \; k_i = r(y_i) \tag{16}$$

where $r(.)$ converts a binary code y_i into an integer k_i and $b(.)$ converts an integer k_i to a binary code y_i.

The solution is omitted due to the space limit.

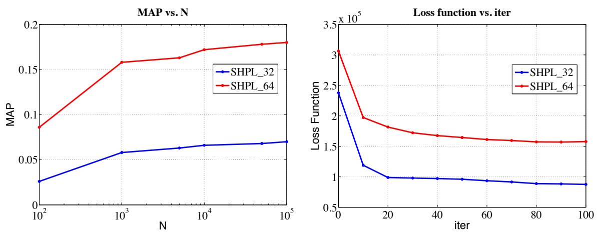
4. EXPERIMENTS

We evaluate our algorithm on the task of high-dimensional approximate nearest neighbor (ANN) search. Firstly, we study the influence of the parameters in our algorithm. Then, we compare our results with state-of-the-art algorithms on three standard datasets.

4.1 Settings

Experiments are conducted on three widely-used high-dimensional datasets: SIFT1M [14], GIST1M [14], and SIFT1B [14]. Each dataset comprises of disjoint one training set, one query set, and one base database (on which the search is performed). SIFT1M provides 10^5 training points, 10^4 query points and 10^6 database points with each point being a 128-dimensional SIFT descriptor. GIST1M provides 5×10^5 training points, 10^3 query points and 10^6 database points with each point being a 960-dimensional GIST feature. SIFT1B is composed of 10^8 training points, 10^4 query points and as large as 10^9 database points. Following [10], we use the first 10^6 training points on the SIFT1B datasets. The whole training set is used on SIFT1M and GIST1M.

ANN search is conducted to evaluate our proposed approaches, and two indicators are reported, namely Recall *vs.* K (top-K results) and Mean Average Precision (MAP).



(a) Performance variance with N (b) Number of iterations

Figure 1: Parameters study with code length 32 and 64 on SIFT1M

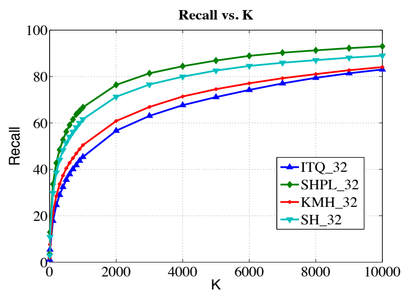
We compare our SHPL with other state-of-the-art hashing algorithms, such as spectral hashing (SH) [16], iterative quantization (ITQ) hashing [3] and K-means Hashing (KMH) [4]. Some other hashing methods (e.g, minimal loss hashing (MLH) [9], PCA hashing (PCAH) [14]) are not compared here because they are outperformed by these compared methods.

4.2 Parameters

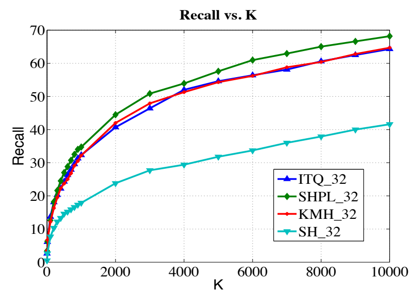
There are several parameters, e.g., the size of the training datasets N and the number of iteration $iter$, affecting the performance of our algorithm. In this subsection, we study the performance variance with different parameters. Due to the space limit, we only report the results on the SIFT1M dataset. The default settings for SIFT1M are: $iter = 50$ and $N = 10^5$. The size of training dataset N affects the training speed and model accuracy. We tune $N = 10^3, 5 \times 10^3, 10^4, 5 \times 10^4, 10^5$, and illustrate the performance changes in Fig. 1(a). There is an increasing trend with the rising of N. When N reaches 10^4, further increasing the training number will not improve the MAP significantly. The loss function in each iteration is shown in Fig. 1(b). The loss function drops dramatically in the first 20 iterations, and then keeps stable after 40 iterations. This indicates the efficiency of our solution.

4.3 Results

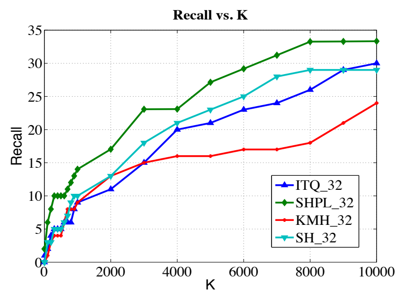
Fig. 2 shows the comparisons of different unsupervised hashing methods on the three unlabeled datasets. We have tested L=32 and L=64. We used the codes provided by the authors to compare different algorithms. For KMH, the bit number of each subspace is 8 for L=32 and $L = 64$. Larger bit number will increase the performance of KMH, but will cause out-of-memory for our computer. For the other algorithms, the default settings are used. From these figures, we have the following observations:



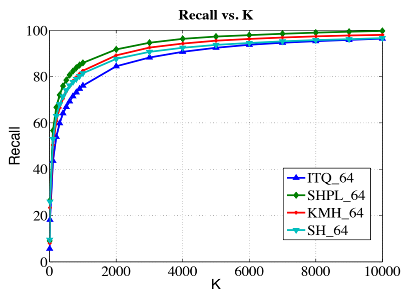
(a) SIFT1M_32



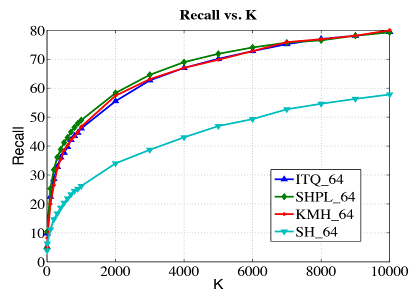
(b) GIST1M_32



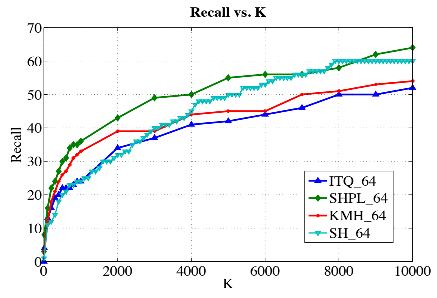
(c) SIFT1B_32



(d) SIFT1M_64



(e) GIST1M_64



(f) SIFT1B_64

Figure 2: The comparison of different unsupervised hashing methods with code length 32 **and** 64

- Our method consistently outperforms the other hashing methods in all datasets. In SIFT1M and SIFT1B datasets, the improvement of SHPL over the other methods is more significant, compared with that in GIST1M dataset. Also, the improvements gap between SHPL and other hashing methods is larger in the case of $L = 32$, than that of $L = 64$.

- With the increase of code length, the performance of different hashing methods is improved accordingly. More specifically, the recall improvements of KMH (20%-28%) and ITQ (20%-25%) are generally more significant than SH (10%-15%) on SIFT1M and SIFT1B dataset, while the improvements on GIST1M dataset are more consistent.

- SH performs surprisingly well in SIFT1M and SIFT1B datasets, but it is inferior in GIST1M dataset. KMH is competitive in most settings, especially when the code length is 64 bits.

5. CONCLUSION

In this work, we propose a Supervised Hashing with Pseudo Labels (SHPL), a general framework which uses the supervised hashing models to improve the effectiveness of hashing methods by the pseudo label strategy. It is shown that SHPL has stronger discriminative power and thus achieves better performance. The pseudo label strategy can also be integrated to unsupervised hashing models. Experiments on three large-scale datasets demonstrate that SHPL obtains superior accuracy over existing hashing methods.

6. ACKNOWLEDGMENTS

The work of Jingkuan Song, Yan Yan and Nicu Sebe has been partially supported by the EC project xLiMe. The work of Lianli Gao is partially supported by the Fundamental Research Funds for the Central Universities of China under grant No.ZYGX2014J063 and No.ZYGX2014Z007.

7. REFERENCES

[1] L. Gao, J. Song, F. Nie, Y. Yan, N. Sebe, and H. T. Shen. Optimal graph leaning with partial tags and multiple features for image and video annotation. In *CVPR*, 2015.

[2] T. Ge, K. He, and J. Sun. Graph cuts for supervised binary coding. In *ECCV*, 2014.

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

[4] K. He, F. Wen, and J. Sun. K-means hashing: An affinity-preserving quantization method for learning binary compact codes. In *CVPR*, 2013.

[5] G. Irie, Z. Li, X. Wu, and S. Chang. Locally linear hashing for extracting non-linear manifolds. In *CVPR*, 2014.

[6] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[7] 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*, 2014.

[8] W. Liu, J. Wang, R. Ji, Y. Jiang, and S. Chang. Supervised hashing with kernels. In *CVPR*, 2012.

[9] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *ICML*, 2011.

[10] M. Norouzi and D. J. Fleet. Cartesian k-means. In *CVPR*, 2013.

[11] J. Song, Y. Yang, Z. Huang, H. T. Shen, and R. Hong. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *ACM MM*, 2011.

[12] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *SIGMOD*, 2013.

[13] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua. Ldahash: Improved matching with smaller descriptors. *TPAMI*, 34(1):66–78, 2012.

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

[15] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *CoRR*, abs/1408.2927, 2014.

[16] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, 2008.