BINARY CODE LEARNING WITH SEMANTIC RANKING BASED SUPERVISION

Viet-Anh Nguyen[1] *and Minh N. Do*[2]

[1]Advanced Digital Sciences Center (ADSC), Singapore
[2]University of Illinois at Urbana-Champaign, IL, USA

ABSTRACT

Recent years have witnessed the increasing popularity of binary hashing for efficient similarity search in large-scale vision problems. This paper presents a novel Supervised Ranking-Based Hashing (SRH) method for efficient binary code learning to better capture the semantic nearest neighbors and improve the search performance. In particular, a family of hash functions is designed to preserve the semantic data structure in the original high-dimensional space by utilizing the semantic ranking order information induced by any specific query. The proposed hashing framework is obtained by jointly minimizing the empirical error over the ranking violation in the binary code space together with the quantization loss between the original data and the binary codes. Furthermore, an effective regularizer for maximizing the even binary code distribution is also taken into account in the optimization to generate more efficient and compact binary codes. Experimental results have demonstrated the proposed method outperforms the state-of-the-art.

Index Terms— semantic hashing, binary code learning, image retrieval, approximate nearest neighbor search

1. INTRODUCTION

Approximate nearest neighbor (ANN) search has become increasingly important in large-scale vision problems (e.g., image retrieval, object recognition, pose estimation, etc.) in order to search for good enough nearest neighbors instead of the exact ones in a sub-linear or even constant search time complexity. One popular approach for ANN search is the hashing-based approach that has attracted more attention recently. The basic idea behind hashing is to construct a set of similarity preserving hash functions to map similar data points of the original high-dimensional space into nearby compact binary codes. As a result, similarity search can be efficiently computed on the basis of the Hamming distance with millions of distances per second and result in a substantial search time reduction even in an exhaustive search manner. In addition, compact binary codes also enable storage efficiency for the explosive growth of large-scale databases containing millions or even billions of data points together with the increased dimensionality of data feature descriptors.

The most popular and pioneered hashing techniques are the Locality Sensitive Hashing (LSH) family [1] and its extension Shift-Invariant Kernels Locality Sensitive Hashing (SKLSH) [2], which generate random projection bases from a Gaussian distribution as a set of hash functions to approximate the cosine similarity. Being data-independent, LSH and its extension require multiple hash tables with long binary codes to achieve an acceptable performance that may not scale well with the growth of data. To generate more compact binary codes, many data-dependent methods have been proposed to learn the hash functions using the training data, which can be broadly divided into *unsupervised* and *supervised* methods. Most popular unsupervised methods are Spectral Hashing (SH) [3], PCA-Hash (PCAH) [4], and Iterative Quantization (ITQ) [5]. Without using any labeled data, these methods mostly aim to preserve the data structure of a given metric (e.g., Euclidean) and do not guarantee good nearest neighbors due to the semantic gap. To capture the semantic nearest neighbors and improve the search performance, a number of supervised data-dependent methods have also been proposed by leveraging on the labeled data. The most representative methods include Binary Reconstruction Embedding (BRE) [6], Semi-Supervised Hashing (SSH) [7], Supervised Discriminative Hashing (SDH) [8], and Kernel Supervised Hashing (KSH) [9]. However, having mostly focused on the similarity matrix preservation, these methods may be sub-optimal due to the lack of fully utilizing ranking information.

To address such a shortcoming, a few hashing methods have been proposed to use the ranking information and preserve the order correlation between the original and binary code spaces. They include Minimal Loss Hashing (MLH) [10], Order Preserving Hashing (OPH) [11], Hamming Distance Metric Learning (HDML) [12]. Essentially, these methods formulate the optimization criteria in terms of penalty for either global order misalignment or local ranking loss and aim at optimizing a certain upper-bound surrogate loss function. Such a loss function is often complex and difficult to solve. Furthermore, like most of the existing methods, they typically adopt a naive two-stage paradigm to obtain the binary codes by learning a continuous embedding and then binarizing it. Despite its simplicity, it is likely to result in a sub-optimal performance due to the distort of the learned rank-preserving embedding introduced by the binary quantization.

Recently, a few methods have leveraged on deep convolutional neural network (CNN) to jointly learn the image representation and hash codes [13, 14, 15]. However, it is worth mentioning that the performance gains of these methods may mostly come from the powerful feature representations learned by the deep neural network. It has been observed that in a fair comparison setup the conventional supervised hashing methods such as KSH or ITQ may also achieve comparable hashing performance using these features without much concern about the training time and hashing speed as in the case of the deep neural network based approach.

In this paper, we propose a novel Supervised Ranking-Based Hashing (SRH) method to better capture the semantic similar neighbors in the binary space. Specifically, a family of hash functions is designed to preserve the semantic data structure of the original high-dimensional space (e.g., handcrafted or CNN-based features) by utilizing the semantic ranking order information induced by any specific query. We formulate and minimize the objective function

This study is supported by the research grant for the Human-Centered Cyber-physical Systems Programme at the Advanced Digital Sciences Center from Singapores Agency for Science, Technology and Research (A*STAR).

that consists of two main components: supervised empirical fitness and unsupervised information regularization. The empirical fitness is derived over any ranking violation in the binary code space based on the ground-truth ranking order information. Though sharing some similarity with HDML [12], our method differs significantly in the proposed final objective function and optimization. Unlike HDML [12] and other two-stage paradigm approaches, the proposed hashing framework is obtained by jointly optimizing the empirical fitness and effective regularization to reduce the distort of the learned rank-preserving embedding during binarization and generate more efficient and compact binary codes. Particularly, unsupervised regularizers are introduced to minimize the quantization loss between the original data points and the binary codes and maximize the even distribution of the binary codes. Furthermore, using a similarity function of bi-linear form in the Hamming space and simple but effective relaxation makes our final objective function easy to be manipulated and optimized. Extensive experimental results have shown the superior performance of the proposed hashing method compared to the state-of-the-art methods.

2. PROPOSED METHOD

2.1. Notations and Definitions

Given a dataset of N data points $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N \in \mathbb{R}^D$ that forms the data matrix $\mathbf{X} \in \mathbb{R}^{D\times N}$, where each column is a data point. The principle of binary hashing is to use a set of hash functions $\mathbf{H} = \{h_k : \mathbb{R}^D \rightarrow \mathbb{H}\}_{k=1}^K$ to map each $\mathbf{x}_i$ to $\mathbf{H}(\mathbf{x}_i) \in \mathbb{H}^K$ that is the K-dimensional Hamming space $\mathbb{H}^K \in \{-1, 1\}^K$. The binary code of $\mathbf{x}_i$ is denoted as $\mathbf{y}_i = \mathbf{H}(\mathbf{x}_i)$ that forms the i-th column of the binary code matrix $\mathbf{Y} \in \{-1, 1\}^{K\times N}$. In this work, we use the affine embedding as a hash function. Specifically, given a projection vector $\mathbf{w}_k \in \mathbb{R}^D$, the k-th hash function is defined as

$$h_k(\mathbf{x}_i) = \mathrm{sgn}(\mathbf{w}_k^T\mathbf{x}_i + b_k) \tag{1}$$

where

$$\mathrm{sgn}(v) = \begin{cases} 1 & \text{if } v \geq 0 \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

and b_k is the mean of the projected data. Without loss of generality, we could assume the data are zero-centered and simply set b_k to 0. Let $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_K] \in \mathbb{R}^{D\times K}$, we can write $\mathbf{y}_i = \mathrm{sgn}(\mathbf{W}^T\mathbf{x}_i)$. Thus, the entire binary coding process can be represented in matrix form as $\mathbf{Y} = \mathrm{sgn}(\mathbf{W}^T\mathbf{X})$, where $\mathrm{sgn}(\mathbf{W}^T\mathbf{X})$ is the matrix of signs of individual elements.

2.2. Formulation

Let $\mathcal{S}(\mathbf{x}_i, \mathbf{x}_j) \in \mathbb{R}$ be a pairwise similarity measure of data points $\mathbf{x}_i$ and $\mathbf{x}_j \in \mathcal{X}$, which indicates how strongly $\mathbf{x}_i$ and $\mathbf{x}_j$ are semantically related to each other. For example, such a similarity measure can be derived from labeled information where strongly similar data points typically have more commonly shared semantic labels. For any specific query point $\mathbf{x}_q$, denote the ground-truth ranking order based on $\mathcal{S}$ over $\mathcal{X}$ in the original space as

$$r(\mathbf{x}_q, \mathcal{X}, \mathcal{S}) = (r_1, \ldots, r_i, \ldots, r_N) \tag{3}$$

wher $r_i \in [1, N]$ and $\mathcal{S}(\mathbf{x}_q, \mathbf{x}_i) > \mathcal{S}(\mathbf{x}_q, \mathbf{x}_j)$ if $r_i < r_j$. The basic idea of ranking-based hashing is to preserve such a ranking relation in the embedded binary space where its ranking is derived based on the Hamming distance between the binary codes of the query and database points. Mathematically, it is desired to preserve the following condition $\mathcal{D}_h(\mathbf{y}_q, \mathbf{y}_i) < \mathcal{D}_h(\mathbf{y}_q, \mathbf{y}_j)$ if $r_i < r_j$, where $\mathcal{D}_h(\cdot, \cdot)$ is the Hamming distance measure defined as

$$\mathcal{D}_h(\mathbf{y}_q, \mathbf{y}_i) = |\{k|h_k(\mathbf{x}_q) \neq h_k(\mathbf{x}_i)\}, 1 \leq k \leq K| \tag{4}$$

For the ease of hash function learning, we represent the ground-truth ranking order into a set of rank triplets of $(\mathbf{x}_q, \mathbf{x}_i, \mathbf{x}_j)$, in which $\mathbf{x}_i$ has a higher rank than $\mathbf{x}_j$ in response to the query $\mathbf{x}_q$. These triplets indicate the ranking comparison information such that the similarity between $\mathbf{x}_q$ and $\mathbf{x}_i$ is greater than that between $\mathbf{x}_q$ and $\mathbf{x}_j$. Such a representation also allows the flexibility in case we do not have the full access to all values of $\mathcal{S}(\mathbf{x}_i, \mathbf{x}_j)$ and the incomplete ranking order, if any. Thus, the desired hash functions can be efficiently learned by preserving the relative ranking information deduced from these triplets instead. More specifically, our learning paradigm aims to minimize the empirical error over the ranking violation of these triplets in the binary space while jointly optimizing the other desirable properties of the binary codes (e.g., quantization loss and compactness).

2.3. Empirical Fitness

To learn the hash functions, we formulate the empirical fitness to penalize the violation of the relative ranking in the triplets within the large-margin learning framework. Instead of directly optimizing on the Hamming distance with nontrivial and complex mathematical formula, we utilize the code inner products of the binary codes. It has been shown in [9] that the code inner product is in one-to-one correspondence to the Hamming distance as

$$\mathbf{y}_i^T\mathbf{y}_j = K - 2\mathcal{D}_h(\mathbf{y}_i, \mathbf{y}_j) \tag{5}$$

The code inner product essentially represents a similarity measure $\mathcal{S}_h(\cdot, \cdot)$ of binary codes in the Hamming space. To preserve the ranking information of the triplet $(\mathbf{x}_q, \mathbf{x}_i, \mathbf{x}_j)$ in the original space, the goal is to find a family of hash functions to satisfy

$$\mathcal{S}_h(\mathbf{y}_q, \mathbf{y}_i) > \mathcal{S}_h(\mathbf{y}_q, \mathbf{y}_j) + \rho \tag{6}$$

where ρ is a safety margin. We employ a typical hinge loss function used in SVM [16] to define the empirical loss for the triplet as

$$l(\mathbf{x}_q, \mathbf{x}_i, \mathbf{x}_j) = \max\{0, \rho - \mathcal{S}_h(\mathbf{y}_q, \mathbf{y}_i) + \mathcal{S}_h(\mathbf{y}_q, \mathbf{y}_j)\} \tag{7}$$

The basic idea is to penalize any ranking violation of a triplet with a safe margin in the binary code space. Meanwhile, a high precision score (i.e., more relevant database points returned for a given specific query) is generally desirable in practice. As a result, it is preferable to preserve the ranking orders of the top ranked items in the original space. That is a higher ranking violation penalty should be imposed on the triplets containing the top ranked items and vice verse. Here, we follow [17] to define a rank-to-loss weight for the item of rank k as ϑ_k and choose $\vartheta_k = 1/k$, which yields state-of-the-art precision results in learning to rank. The empirical fitness is formulated as the accumulated weighted ranking loss over all possible triplets in the training set as

$$J_{\mathrm{e}}(\mathbf{W}) = \sum_{(\mathbf{x}_q, \mathbf{x}_i, \mathbf{x}_j)} \kappa(\mathbf{x}_i, \mathbf{x}_j) l(\mathbf{x}_q, \mathbf{x}_i, \mathbf{x}_j) \tag{8}$$

where

$$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \sum_{k=r_i}^{r_j} \vartheta_k \tag{9}$$

Since $J_{\mathrm{e}}(\mathbf{W})$ is non-differentiable due to the non-linear $\mathrm{sgn}(\cdot)$, minimizing such an objective function is an NP-hard problem. Similar to the spectral hashing method [3], we adopt the spectral relaxation by relaxing the sign of projection with its signed magnitude. Compared to the complex surrogate loss function with loss-augmented inference in [12], the spectral relaxation makes it more efficient to optimize the final objective function when jointly considering different crucial regularizers as shown later. Thus, we can rewrite the empirical term as follows

$$J_{\mathrm{e}}(\mathbf{W}) = \sum_{(\mathbf{x}_q,\mathbf{x}_i,\mathbf{x}_j)} \kappa(\mathbf{x}_i,\mathbf{x}_j) \max\Big\{0, \rho - \mathbf{x}_q^T\mathbf{W}\mathbf{W}^T\mathbf{x}_i + \mathbf{x}_q^T\mathbf{W}\mathbf{W}^T\mathbf{x}_j\Big\} \quad (10)$$

2.4. Regularizers

To retain generalization and minimize the distort of the continuous rank-preserving embedding, we use several regularizers that reflect the key properties of hash codes. The most desirable criteria is to minimize the quantization loss in the training set (i.e., the distortion of the learned embedding introduced by binarization), which can be imposed by the additional loss term

$$\begin{aligned} J_{\mathrm{q}}(\mathbf{W}) &= \sum_i ||\mathbf{y}_i - \mathbf{W}^T\mathbf{x}_i||^2 \\ &= ||\mathbf{Y} - \mathbf{W}^T\mathbf{X}||_F^2 \end{aligned} \quad (11)$$

For hash code compactness, it requires each bit to fire 50% of the time, which can be mathematically represented by $\sum_i \mathbf{y}_i = \mathbf{0}$ and be enforced by adding the bit balancing regularizer as

$$J_{\mathrm{b}}(\mathbf{W}) = ||\sum_i \mathbf{y}_i||^2 = ||\mathbf{Y}\mathbf{1}||_F^2 \quad (12)$$

where $\mathbf{1} \in \mathbb{R}^N$ is the column vector of all 1 elements. Adopting spectral relaxation, the bit balancing constraint regularizer can be written as

$$\begin{aligned} J_{\mathrm{b}}(\mathbf{W}) &= \mathbf{tr}\{\mathbf{Y}\mathbf{1}\mathbf{1}^T\mathbf{Y}^T\} \\ &\cong \mathbf{tr}\{\mathbf{W}^T\mathbf{X}\mathbf{Q}\mathbf{X}^T\mathbf{W}\} \end{aligned} \quad (13)$$

where $\mathbf{Q} = \mathbf{1}\mathbf{1}^T \in \mathbb{R}^{N\times N}$. Combining the empirical and regularization terms, the overall objective function is given as

$$\begin{aligned} J(\mathbf{W}) &= J_{\mathrm{e}}(\mathbf{W}) + \lambda J_{\mathrm{q}}(\mathbf{W}) + \beta J_{\mathrm{b}}(\mathbf{W}) \\ &= \sum_{(\mathbf{x}_q,\mathbf{x}_i,\mathbf{x}_j)} \kappa(\mathbf{x}_i,\mathbf{x}_j) \max\Big\{0, \rho - \mathbf{x}_q^T\mathbf{W}\mathbf{W}^T\mathbf{x}_i \\ &+ \mathbf{x}_q^T\mathbf{W}\mathbf{W}^T\mathbf{x}_j\Big\} + \lambda||\mathbf{Y} - \mathbf{W}^T\mathbf{X}||_F^2 + \beta\mathbf{tr}\{\mathbf{W}^T\mathbf{X}\mathbf{Q}\mathbf{X}^T\mathbf{W}\} \end{aligned} \quad (14)$$

where λ and β are the regularization parameters.

2.5. Optimization

It is rather challenging to find a closed form solution for the above function. Thus, we optimize the objective function iteratively over the sampled rank triplets of the training set in batches to update $\mathbf{W}$ and $\mathbf{Y}$ alternatively when the other variable is fixed. Although the final objective function is not convex with respect to $\mathbf{W}$ due to the empirical term, it is observed that optimization in terms of $\mathbf{W}$ does not appear to suffer from very poor local minima. These two alternating steps in each iteration are as follows.

Fix $\mathbf{W}$ and update $\mathbf{Y}$. With a fixed $\mathbf{W}$, minimizing $J(\mathbf{W})$ is equivalent to minimizing the following term

$$\begin{aligned} ||\mathbf{Y} - \mathbf{W}^T\mathbf{X}||_F^2 &= ||\mathbf{Y}||_F^2 + ||\mathbf{W}^T\mathbf{X}||_F^2 - 2\mathbf{tr}\{\mathbf{Y}\mathbf{X}^T\mathbf{W}\} \\ &= NK + ||\mathbf{W}^T\mathbf{X}||_F^2 - 2\mathbf{tr}\{\mathbf{Y}\mathbf{X}^T\mathbf{W}\} \end{aligned} \quad (15)$$

It has been shown in [5] that the solution of the binary code matrix $\mathbf{Y}$ for minimizing Eq. (15) can be given by

$$\mathbf{Y} = \mathrm{sgn}(\mathbf{W}^T\mathbf{X}) \quad (16)$$

As a result, we use such an assignment to update $\mathbf{Y}$ in our paper.

Fix $\mathbf{Y}$ and update $\mathbf{W}$. We adopt here the stochastic gradient descent approach to update $\mathbf{W}$. For a given ranking triplet $(\mathbf{x}_q, \mathbf{x}_i, \mathbf{x}_j)$, if

$$\rho - \mathbf{x}_q^T\mathbf{W}\mathbf{W}^T\mathbf{x}_i + \mathbf{x}_q^T\mathbf{W}\mathbf{W}^T\mathbf{x}_j < 0 \quad (17)$$

the hinge loss of the current triplet has a negative argument and makes no contribution to the overall empirical loss. Otherwise, given the fact that the binary code matrix $\mathbf{Y}$ and the training data $\mathbf{X}$ are fixed, we can compute the following gradient for the update rule as

$$\begin{aligned} \frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} &= \sum_{(\mathbf{x}_q,\mathbf{x}_i,\mathbf{x}_j)} \kappa(\mathbf{x}_i,\mathbf{x}_j)\Big(\mathbf{x}_q(\mathbf{x}_j - \mathbf{x}_i)^T + (\mathbf{x}_j - \mathbf{x}_i)\mathbf{x}_q^T\Big)\mathbf{W} \\ &+ 2\lambda\mathbf{X}(\mathbf{X}^T\mathbf{W} - \mathbf{Y}^T) + 2\beta\mathbf{X}\mathbf{Q}\mathbf{X}^T\mathbf{W} \end{aligned} \quad (18)$$

Thus, we can update $\mathbf{W}$ as

$$\mathbf{W} = \mathbf{W} - \eta\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}} \quad (19)$$

where the step size η was chosen by cross validation. Algorithm 1 summarizes the detailed procedure of the proposed SRH method.

Algorithm 1: Supervised Ranking-Based Hashing

INPUT: Training set $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^N$, parameters λ, β, η iteration number T, and binary code length K

OUTPUT: Projection matrix $\mathbf{W}$

Step 1: Initialization

Initialize $\mathbf{W}$ randomly from zero-mean normal distribution

Step 2: Optimization

FOR $t = 1, 2, \ldots, T$

- Sample a batch of rank triplets $(\mathbf{x}_q, \mathbf{x}_i, \mathbf{x}_j)$ such that $\mathcal{S}(\mathbf{x}_q, \mathbf{x}_i) > \mathcal{S}(\mathbf{x}_q, \mathbf{x}_j)$
- Fix $\mathbf{W}$ and update $\mathbf{Y}$ using Eq. (16)
- Fix $\mathbf{Y}$ and update $\mathbf{W} = \mathbf{W} - \eta\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$ where $\frac{\partial J(\mathbf{W})}{\partial \mathbf{W}}$ is computed by using Eq. (18)

3. EXPERIMENTAL RESULTS

We evaluated the proposed hashing method by comparing with a few state-of-the-art methods in both unsupervised and supervised categories. They include LSH [1], SH [3], SKLSH [2], PCAH [4], ITQ [5], supervised BRE [6], HDML [12], SSH [7], and KSH [9]. For implementation, we used the source codes provided by the authors and followed their instructions to tune the parameters. For supervised methods, we extracted a small set of labeled samples in each dataset for training purpose and used for all supervised methods. The

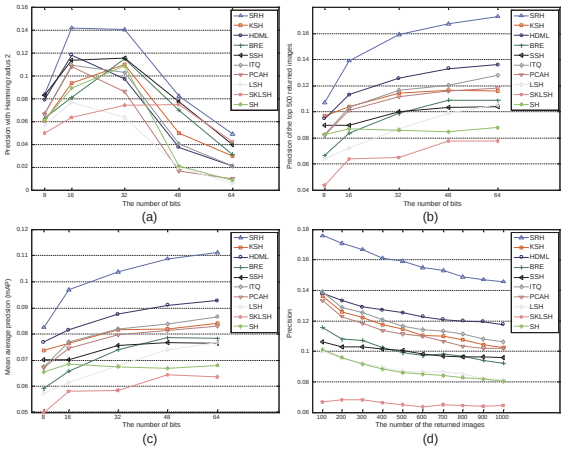

Fig. 1. Comparative evaluation on the CIFAR-100 dataset. (a) Precision within Hamming radius 2 using hash lookup, (b) Precision of the top 500 returned images using Hamming ranking, (c) Mean average precision (mAP), (d) Mean precision curves of Hamming ranking with 32 bits.

regularization parameters were roughly tuned using the cross validation technique.

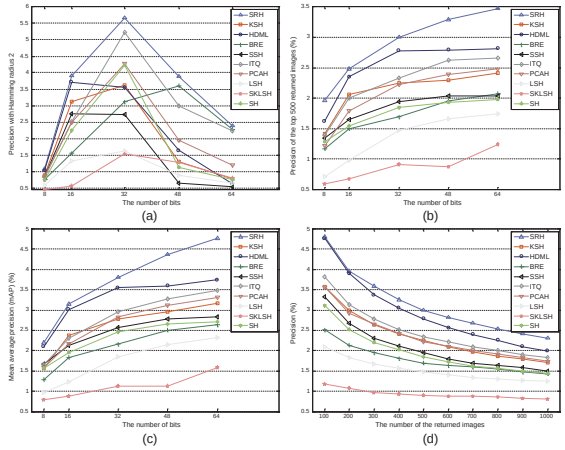

Fig. 2. Comparative evaluation on the Caltech256 dataset. (a) Precision within Hamming radius 2 using hash lookup, (b) Precision of the top 500 returned images using Hamming ranking, (c) Mean average precision (mAP), (d) Mean precision curves of Hamming ranking with 32 bits.

3.1. Dataset and protocols

The experiments were conducted using two benchmark datasets: CIFAR100[1] and Caltech256[2]. The CIFAR100 dataset consists of 60,000 images of size 32×32 pixels, in which each image is labeled in two semantic levels of coarse (20 classes) and fine classes (5 classes in each coarse category). The semantic similarity measure is derived based on the number of shared class labels. For the data points with the same similarity measure to the query, we assigned the same ranking level and derived a weak ground-truth ranking order of three similarity levels. We used 512-dimensional GIST descriptors to represent the images. The Caltech256 dataset contains 30,607 images grouped in 256 categories with more than 80 images per category. With single label images, the ranking order essentially contains two similarity levels of relevant and irrelevant images to a specific query. We used a 1024-dimensional bag of words (BoW) descriptor based on dense SIFT features to represent each image.

For evaluation, we randomly sampled in each dataset 1,000 images as queries, 2,000 images for the supervised training purpose, and the remaining images for the database construction. We used the common class labels as ground truth to evaluate the returned images. Two search protocols were used: 1) Hash lookup within a Hamming radius, 2) Hamming ranking of the top returned images. We reported the widely used measures including the precision within Hamming radius or by Hamming ranking, mean precision curve, and mean average precision (mAP) using 8 to 64 hash bits. It was also observed that the mAP saturates after a certain number of iterations. For efficiency, we used a small subset of the training set for validation to trace the saturation of mAP at regular intervals instead of sampling all possible $O(N^3)$ rank triplets.

3.2. Results

The performance comparisons of different methods using CIFAR-100 and Caltech256 datasets are shown in Figs. 1 and 2, respectively. The results show the proposed method provided a significantly better performance compared to that of the existing methods, especially in terms of the precision measure for both hash lookup and Hamming ranking search protocols, indicating more accurate performance of the returned results. It is also observed the unsupervised data-independent methods such as LSH and SKLSH generally could not provide the good performance compared to the other supervised methods due to the fact that without making use of any labeled data, these unsupervised methods could not capture well the semantic similarity. While sharing somewhat similarity in terms of leveraging on the large-margin framework, the performance of HDML was mostly worse than that of our method. This certainly indicates that using a bi-linear similarity function in the Hamming space and simple spectral relaxing not only makes it easy to optimize the objective function but also achieves a good performance. In addition, such a superior performance is also likely due to the fact that the quantization loss and bit balancing regularizations are jointly optimized in our framework to maximize the desirable properties of the learned binary codes. For further evaluation, Figs. 1(d) and 2(d) show the average precision curves obtained by varying the number of the returned images. It can be seen the results are totally aligned with the trends observed in the other figures for the two benchmark datasets.

4. CONCLUSIONS

We have presented in this paper a novel Supervised Ranking-Based Hashing (SRH) method by leveraging on the ranking order information to preserve the semantic data structure in the original space. A family of hash functions has been learned by jointly minimizing the empirical error over the ranking violation in the binary code space together with effective regularization to maximize other desirable properties of the binary codes. Specifically, the quantization loss between the original data and the binary codes together with the even binary code distribution have also been taken into account in the optimization to generate more efficient and compact codes. Experimental results have shown the proposed method outperforms the state-of-the-art.

[1]http://www.cs.toronto.edu/ kriz/cifar.html
[2]http://authors.library.caltech.edu/7694/

5. REFERENCES

[1] A. Gionis *et al*, "Similarity search in high dimensions via hashing," in *Proc. VLDB*, 1999.

[2] M. Raginsky and S. Lazebnik, "Locality sensitive binary codes from shift-invariant kernels," in *Proc. NIPS*, 2009.

[3] Y. Weiss *et al*, "Spectral hashing," in *Proc. NIPS*, 2008.

[4] J. Wang *et al*, "Sequential projection learning for hashing with compact codes," in *Proc. ICML*, 2010.

[5] Y. Gong *et al*, "Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval," *TPAMI*, 2012.

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

[7] J. Wang *et al*, "Semi-supervised hashing for large scale search," *TPAMI*, 2012.

[8] V. A. Nguyen *et al*, "Supervised discriminative hashing for compact binary codes," in *Proc. ACM Multimedia*, 2014.

[9] W. Liu *et al*, "Supervised hashing with kernels," in *Proc. CVPR*, 2012.

[10] M. Norouzi and D. Fleet, "Minimal loss hashing for compact binary codes," in *Proc. Int. Conf. Mach. Learn.*, 2011.

[11] J. Wang *et al*, "Order preserving hashing for approximate nearest neighbor search," in *Proc. ACM MM*, 2013.

[12] M. Norouzi *et al*, "Hamming distance metric learning," in *Proc. NIPS*, 2012.

[13] R. Xia *et al*, "Supervised hashing for image retrieval via image representation learning," in *Proc. AAAI Conference on Artificial Intelligence*, 2014.

[14] H. Lai, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE CVPR*, 2015.

[15] J. Wang *et al*, "Learning fine-grained image similarity with deep ranking," in *Proc. IEEE CVPR*, 2014.

[16] B. Scholkopf *et al*, "Learning with kernels: support vector machines, regularization, optimization, and beyond," *MIT Press, Cambridge*, 2002.

[17] J. Weston *et al*, "Large scale image annotation: learning to rank with joint word-image embeddings," *Machine Learning*, 2010.