Simultaneous Feature Aggregating and Hashing for Large-scale Image Search

Thanh-Toan Do[†⋆] Dang-Khoa Le Tan[⋆] Trung T. Pham[†] Ngai-Man Cheung[⋆]
[†]The University of Adelaide, [⋆]Singapore University of Technology and Design
{thanh-toan.do, trung.pham}@adelaide.edu.au, {letandang_khoa, ngaiman_cheung}@sutd.edu.sg

Abstract

In most state-of-the-art hashing-based visual search systems, local image descriptors of an image are first aggregated as a single feature vector. This feature vector is then subjected to a hashing function that produces a binary hash code. In previous work, the aggregating and the hashing processes are designed independently. In this paper, we propose a novel framework where feature aggregating and hashing are designed simultaneously and optimized jointly. Specifically, our joint optimization produces aggregated representations that can be better reconstructed by some binary codes. This leads to more discriminative binary hash codes and improved retrieval accuracy. In addition, we also propose a fast version of the recently-proposed Binary Autoencoder to be used in our proposed framework. We perform extensive retrieval experiments on several benchmark datasets with both SIFT and convolutional features. Our results suggest that the proposed framework achieves significant improvements over the state of the art.

1. Introduction

We are interested in the problem of large-scale image search in which finding a compact image representation is one of the crucial problems. State-of-the-art image search systems [22, 2, 20, 1, 9] include three main steps in computing the image representation: local feature extraction, embedding, and aggregating. The local feature extraction step extracts a set of local features, e.g. SIFT [31], representing the image. The embedding step improves the discriminativeness of the local features by mapping these features into a high-dimensional space [20, 22, 9]. The aggregating (pooling) step converts the set of mapped high dimensional vectors into a single vector representation which usually has the dimensionality of several thousands [20, 22, 9]. In particular, the aggregating step is very important. First, the aggregating step reduces the storage requirement which is one of main concerns in large-scale image search. Second, the aggregated representation vectors can be directly compared using standard metrics such as Euclidean distance.

Although the aggregated representation reduces the storage and allows simple distance-based comparison, it is not efficient enough for large-scale database which requires very compact representation and fast searching. An attractive approach for achieving these requirements is binary hashing. Specifically, binary hashing encodes the image representation into a compact binary hash code. Existing binary hashing methods can be categorized as data-independent and data-dependent schemes [43, 44, 13]. Data-dependent hashing methods use available training data for learning hash functions and they achieve better retrieval results than data-independent methods. The training can be unsupervised [45, 12, 14, 15, 7, 8] or supervised [34, 25, 30, 28]. In particular, unsupervised hashing does not require any label information. Hence, it is suitable for large-scale image search in which the label information is usually unavailable. Therefore, our work focuses on the unsupervised hashing for large-scale image search.

In this work, we propose a novel framework where feature aggregating and hashing are designed simultaneously and optimized jointly. Traditionally, the aggregating/hashing processes are designed independently and separately [11, 23, 16]: First, some aggregation is applied on the local (embedded) features, resulting in a single aggregated representation for each image. Then, the set of aggregated representations is used for learning a hash function which encodes the aggregated representations into compact binary codes. For example, the recent Generalized Max Pooling [33] seeks a representation that can achieve some desirable aggregation property, i.e., equalizing the similarity between the representation and individual local features. This aggregation process does not take into account any aspect of the subsequent hashing, and the resulted representations may not be suitable for hashing: in the context of unsupervised hashing, the aggregated representation may be difficult to be reconstructed by binary codes. On the contrary, in our proposed simultaneous aggregating/hashing framework, we aim to compute aggregated representations that not only can achieve some desired aggregation property (equalized sim-

Table 1. Notations and their corresponding meanings.

Notation	Meaning
$\mathbf{X}$	$\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^{m} \in \mathbb{R}^{D\times m}$: set of m training samples; each column of $\mathbf{X}$ corresponds to one sample
$\mathbf{Z}$	$\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^{m} \in \{-1,+1\}^{L\times m}$: binary code matrix
L	Number of bits to encode a sample
$\mathbf{W}_1, \mathbf{c}_1$	$\mathbf{W}_1 \in \mathbb{R}^{L\times D}, \mathbf{c}_1 \in \mathbb{R}^{L\times 1}$: weight and bias of encoder
$\mathbf{W}_2, \mathbf{c}_2$	$\mathbf{W}_2 \in \mathbb{R}^{D\times L}, \mathbf{c}_2 \in \mathbb{R}^{D\times 1}$: weight and bias of decoder
$\mathcal{V}$	$\mathcal{V} = \{\mathbf{V}_i\}_{i=1}^{m}; \mathbf{V}_i \in \mathbb{R}^{D\times n_i}$ is set of local (embedded) representations of image i; n_i is number of local descriptors of image i
Φ	$\Phi = \{\varphi_i\}_{i=1}^{m} \in \mathbb{R}^{D\times m}$: set of m aggregated vectors; φ_i corresponds to aggregated vector of image i
$\mathbf{1}$	column vector with all $1s$ elements
$\mathbf{I}$	identity matrix

ilarity) but also can be better reconstructed by some binary codes. As the aggregation is more reconstructible, the binary codes can retain more discriminative information, resulting in improved retrieval performance (in unsupervised hashing).

Our specific contributions are: (i) To accelerate simultaneous learning of aggregating and hashing, we first propose a relaxed version of the state-of-the-art unsupervised hashing Binary Autoencoder [6] to be used in our framework. Instead of solving a NP-hard problem with the hard binary constraint on the outputs of the encoder, we propose to solve the problem with relaxation of the binary constraint, i.e., minimizing the binary quantization loss. In order to minimize this loss, we propose to solve the problem with alternating optimization. This proposed hashing method is not only faster in training but also competitive in retrieval accuracy when comparing to Binary Autoencoder [6]. (ii) Our main contribution is a simultaneous feature aggregating/hashing learning approach which takes the local (embedded) features[1] as inputs and learn the aggregation and hashing function simultaneously. We propose alternating learning of the aggregated features and the hash function. (iii) The solid experiments on several image retrieval benchmark datasets show the proposed simultaneous learning significantly outperforms other recent unsupervised hashing methods.

The remaining of this paper is organized as follows. Section 2 presents related works. Section 3 introduces the relaxed version of Binary Autoencoder [6]. Section 4 introduces the simultaneous feature aggregating and hashing. Section 5 presents experimental results. Section 6 concludes the paper.

2. Related work

We summarize the notations in Table 1. Two main components of the proposed simultaneous learning are aggregating and hashing. For aggregating, we rely on the state-of-the-art Generalized Max Pooling [33]. For hashing, we propose a relaxed version of Binary Autoencoder [6]. This section presents a brief overview of Generalized Max Pooling [33] and Binary Autoencoder [6].

Generalized Max Pooling (GMP) [33] Max-pooling [46, 5] is a common aggregation method which aggregates a set of local (embedded) vectors of the image to a single vector. However, classical max-pooling approach can only be applied to BoW or sparse coding features. Recently, in [22] and [33] the authors introduced a generalization of max-pooling (i.e., Generalized Max Pooling (GMP) [33])[2] which can be applied to general features such as VLAD [21], Temb [22], Fisher vector [36]. The main idea of GMP is to equalize the similarity between each local embedded vector and the aggregated representation. In [22, 9], the authors showed that GMP achieves better retrieval accuracy than sum-pooling. Given $\mathbf{V} \in \mathbb{R}^{D\times n}$, the set of n embedded vectors of an image (each embedded vector has dimensionality D), GMP finds the aggregated representation φ which equalizes the similarity (i.e. the dot-product) between each column of $\mathbf{V}$ and φ by solving the following optimization

$$\min_{\varphi} \left(\left\| \mathbf{V}^T \varphi - \mathbf{1} \right\|^2 + \mu \left\| \varphi \right\|^2 \right) \tag{1}$$

(1) is a ridge regression problem which solution is

$$\varphi = \left(\mathbf{V}\mathbf{V}^T + \mu \mathbf{I} \right)^{-1} \mathbf{V}\mathbf{1} \tag{2}$$

Binary Autoencoder (BA)[6] In [6], in order to compute the binary code, the authors minimize the following optimization

$$\min_{\mathbf{h},\mathbf{f},\mathbf{Z}} \sum_{i=1}^{m} \left(\left\| \mathbf{x}_i - \mathbf{f}(\mathbf{z}_i) \right\|^2 + \mu \left\| \mathbf{z}_i - \mathbf{h}(\mathbf{x}_i) \right\|^2 \right) \tag{3}$$

$$\text{s.t. } \mathbf{z}_i \in \{-1, 1\}^L, i = 1, ..., m \tag{4}$$

where $\mathbf{h} = sgn(\mathbf{W}_1 \mathbf{x} + \mathbf{c}_1)$ and $\mathbf{f}$ are encoder and decoder, respectively. By having sgn, the encoder will output binary codes. In the training of BA, the authors compute each variable $\mathbf{f}, \mathbf{h}, \mathbf{Z}$ at a time while holding the other fixed. The authors show that the BA outperforms state-of-the-art unsupervised hashing methods. However, the disadvantage of BA is time-consuming training which is mainly caused by the computing of $\mathbf{h}$ and $\mathbf{Z}$. As $\mathbf{h}$ involves sgn, it cannot be solved analytically. Hence, when computing $\mathbf{h}$, the authors cast the problem as the learning of L separated linear SVM classifiers, i.e., for each $l = 1, ..., L$, they fit a linear SVM to $(\mathbf{X}, \mathbf{Z}_{l,.})$. When computing $\mathbf{Z}$, the authors solve for each sample $\mathbf{x}_i$ independently. Solving $\mathbf{z}_i$ in (3) for each sample

[1] In this work, the embedding is always applied when SIFT features are used.

[2] In [22], the authors named their method as *democratic aggregation*. It actually shares similar idea to *generalized max pooling* [33]

under binary constraint (4) is NP-hard. To handle this, the authors first solve the problem with the relaxed constraint $\mathbf{z}_i \in [-1, 1]$, resulting a continuous solution. They then apply the following procedure several times for getting $\mathbf{z}_i$: for each bit from 1 to L, they evaluate the objective function when the bit equals -1 or 1 with all remaining elements fixed and pick the best value for that bit. The asymptotic complexity for computing $\mathbf{Z}$ over all samples is $\mathcal{O}(mL^3)$.

In the following, we introduce our efficient Relaxed Binary Autoencoder algorithm (Section 3) which will be used in our novel simultaneous feature aggregating and hashing framework (Section 4).

3. Relaxed Binary Autoencoder (RBA)

3.1. Formulation

In order to achieve binary codes, we propose to solve the following constrained optimization

$$\min_{\{\mathbf{W}_i,\mathbf{c}_i\}_{i=1}^2} J = \frac{1}{2}\left\|\mathbf{X} - \left(\mathbf{W}_2(\mathbf{W}_1\mathbf{X} + \mathbf{c}_1\mathbf{1}^T) + \mathbf{c}_2\mathbf{1}^T\right)\right\|^2 + \frac{\beta}{2}\left(\|\mathbf{W}_1\|^2 + \|\mathbf{W}_2\|^2\right) \tag{5}$$

$$\text{s.t. } \mathbf{W}_1\mathbf{X} + \mathbf{c}_1\mathbf{1}^T \in \{-1, 1\}^{L\times m} \tag{6}$$

The constraint (6) makes sure the output of the encoder is binary. The first term of (5) makes sure the binary codes give a good reconstruction of the input, so it encourages (dis)similar inputs map to (dis)similar binary codes. The second term is a regularization that tends to decrease the magnitude of the weights, so it helps to prevent overfitting.

Solving (5) under (6) is difficult due to the binary constraint. In order to overcome this challenge, we propose to solve the relaxed version of the binary constraint, i.e., minimizing the binary quantization loss of the encoder. The proposed method is named as *Relaxed Binary Autoencoder* (RBA). Specifically, we introduce a new auxiliary variable $\mathbf{B}$ and solve the following the optimization

$$\min_{\{\mathbf{W}_i,\mathbf{c}_i\}_{i=1}^2,\mathbf{B}} J = \frac{1}{2}\left\|\mathbf{X} - \left(\mathbf{W}_2\mathbf{B} + \mathbf{c}_2\mathbf{1}^T\right)\right\|^2 + \frac{\lambda}{2}\left\|\mathbf{B} - (\mathbf{W}_1\mathbf{X} + \mathbf{c}_1\mathbf{1}^T)\right\|^2 + \frac{\beta}{2}\left(\|\mathbf{W}_1\|^2 + \|\mathbf{W}_2\|^2\right) \tag{7}$$

$$\text{s.t. } \mathbf{B} \in \{-1, 1\}^{L\times m} \tag{8}$$

The benefit of the auxiliary variable $\mathbf{B}$ is that we can decompose the difficult constrained optimization problem (5) into simpler sub-problems. We use alternating optimization on these sub-problems as will be discussed in detail.

An important difference between the proposed RBA and the original BA is that our encoder does not involve sgn function. The second term of (7) forces the output of encoder close to binary values, i.e., it minimizes the binary quantization loss, while the first term still ensures good reconstruction loss. By setting the penalty parameter λ sufficiently large, we penalize the binary constraint violation severely, thereby forcing the solution of (7) closer to the feasible region of the original problem (5).

3.2. Optimization

In order to solve for $\mathbf{W}_1, \mathbf{c}_1, \mathbf{W}_2, \mathbf{c}_2, \mathbf{B}$ in (7) under constraint (8), we solve each variable at a time while holding the other fixed.

$(\mathbf{W}, \mathbf{c})$-step: When fixing $\mathbf{c}_1, \mathbf{c}_2$ and $\mathbf{B}$, we have the closed forms for $\mathbf{W}_1, \mathbf{W}_2$ as follows

$$\mathbf{W}_1 = \lambda\left(\mathbf{B} - \mathbf{c}_1\mathbf{1}^T\right)\mathbf{X}^T\left(\lambda\mathbf{X}\mathbf{X}^T + \beta\mathbf{I}\right)^{-1} \tag{9}$$

$$\mathbf{W}_2 = \left(\mathbf{X} - \mathbf{c}_2\mathbf{1}^T\right)\mathbf{B}^T\left(\mathbf{B}\mathbf{B}^T + \beta\mathbf{I}\right)^{-1} \tag{10}$$

When fixing $\mathbf{W}_1, \mathbf{W}_2$ and $\mathbf{B}$, we have the closed forms for $\mathbf{c}_1, \mathbf{c}_2$ as follows

$$\mathbf{c}_1 = \frac{1}{m}\left(\mathbf{B} - \mathbf{W}_1\mathbf{X}\right)\mathbf{1} \tag{11}$$

$$\mathbf{c}_2 = \frac{1}{m}\left(\mathbf{X} - \mathbf{W}_2\mathbf{B}\right)\mathbf{1} \tag{12}$$

Note that in (9), the term $\mathbf{X}^T\left(\lambda\mathbf{X}\mathbf{X}^T + \beta\mathbf{I}\right)^{-1}$ is a constant matrix and it is computed only one time.

B-step: When fixing the weight and the bias, we can rewrite (7) as

$$\left\|\widetilde{\mathbf{X}} - \mathbf{W}_2\mathbf{B}\right\|^2 + \lambda\left\|\mathbf{H} - \mathbf{B}\right\|^2 \tag{13}$$

$$\text{s.t. } \mathbf{B} \in \{-1, 1\}^{L\times m} \tag{14}$$

where $\widetilde{\mathbf{X}} = \mathbf{X} - \mathbf{c}_2\mathbf{1}^T$ and $\mathbf{H} = \mathbf{W}_1\mathbf{X} + \mathbf{c}_1\mathbf{1}^T$.

Inspired by the recent progress of discrete optimization [40], we use coordinate descent approach for solving $\mathbf{B}$, i.e., we solve one row of $\mathbf{B}$ each time while fixing all other rows. Specifically, let $\mathbf{Q} = \mathbf{W}_2^T\widetilde{\mathbf{X}} + \lambda\mathbf{H}$; for $k = 1, ..., L$, let $\mathbf{w}_k$ be k^{th} column of $\mathbf{W}_2$; $\overline{\mathbf{W}}_2$ be matrix $\mathbf{W}_2$ excluding $\mathbf{w}_k$; $\mathbf{q}_k$ be k^{th} column of $\mathbf{Q}^T$; $\mathbf{b}_k^T$ be k^{th} row of $\mathbf{B}$; $\overline{\mathbf{B}}$ be matrix $\mathbf{B}$ excluding $\mathbf{b}_k^T$. We have the closed-form solution for $\mathbf{b}_k^T$ as

$$\mathbf{b}_k^T = sgn\left(\mathbf{q}_k^T - \mathbf{w}_k^T\overline{\mathbf{W}}_2\overline{\mathbf{B}}\right) \tag{15}$$

The proposed RBA is summarized in Algorithm 1. In the Algorithm 1, $\mathbf{B}^{(t)}$, $\mathbf{W}_1^{(t)}, \mathbf{c}_1^{(t)}, \mathbf{W}_2^{(t)}, \mathbf{c}_2^{(t)}$ are values at t^{th} iteration. After learning $(\mathbf{W}_1, \mathbf{c}_1, \mathbf{W}_2, \mathbf{c}_2)$, given a new vector $\mathbf{x}$, we pass $\mathbf{x}$ to the encoder, i.e., $h = \mathbf{W}_1\mathbf{x} + \mathbf{c}_1$, and round the values of h to $\{-1, 1\}$, resulting binary codes.

Comparison to Binary Autoencoder (BA) [6]: There are two main advances of the proposed RBA (7) over BA (3). First, our encoder does not involve the sgn function. Hence, during the iterative optimization, instead of using SVM for learning the encoder as in BA, we have an analytic solution ((9) and (11)) for the encoder. Second, when solving for $\mathbf{B}$, instead of solving each sample at a time as in BA, we solve all samples at the same time by adapting the recent advance discrete optimization technique [40]. The asymptotic complexity for computing one row of $\mathbf{B}$, i.e. (15),

Algorithm 1 Relaxed Binary Autoencoder (RBA)

```
Input:
    $\mathbf{X}$: training data; $L$: code length; $T_1$: maximum iteration number; parameters $\lambda, \beta$
Output:
    Parameters $\mathbf{W}_1, \mathbf{c}_1, \mathbf{W}_2, \mathbf{c}_2$

Initialize $\mathbf{B}^{(0)} \in \{1,1\}^{L\times m}$ using ITQ [12]
Initialize $\mathbf{c}_1^{(0)} = \mathbf{0}, \mathbf{c}_2^{(0)} = \mathbf{0}$
for $t = 1 \to T_1$ do
    Fix $\mathbf{B}^{(t-1)}, \mathbf{c}_1^{(t-1)}, \mathbf{c}_2^{(t-1)}$, solve $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)}$ by (9) and (10).
    Fix $\mathbf{B}^{(t-1)}, \mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)}$, solve $\mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}$ by (11) and (12).
    Fix $\mathbf{W}_1^{(t)}, \mathbf{W}_2^{(t)}, \mathbf{c}_1^{(t)}, \mathbf{c}_2^{(t)}$, solve $\mathbf{B}^{(t)}$ by B-step.
end for
Return $\mathbf{W}_1^{(T_1)}, \mathbf{W}_2^{(T_1)}, \mathbf{c}_1^{(T_1)}, \mathbf{c}_2^{(T_1)}$
```

is $\mathcal{O}(mL)$. Hence the asymptotic complexity for computing $\mathbf{B}$ is only $\mathcal{O}(mL^2)$ which is less than $\mathcal{O}(mL^3)$ of BA. These two advances makes the training of RBA is faster than BA.

3.3. Evaluation of Relaxed Binary Autoencoder (RBA)

This section evaluates the proposed RBA and compares it to the following state-of-the-art unsupervised hashing methods: Iterative Quantization (ITQ) [12], Binary Autoencoder (BA) [6], Spherical Hashing (SPH) [15], K-means Hashing (KMH) [14]. For all compared methods, we use the implementations and the suggested parameters provided by the authors. The values of λ, β and the number of iteration T_1 in the Algorithm 1 are empirically set by cross validation as $10^{-2}, 1$ and 10, respectively. The BA [6] and the proposed RBA required an initialization for the binary code. To make a fair comparison, we follow [6], i.e., using ITQ [12] for the initialization.

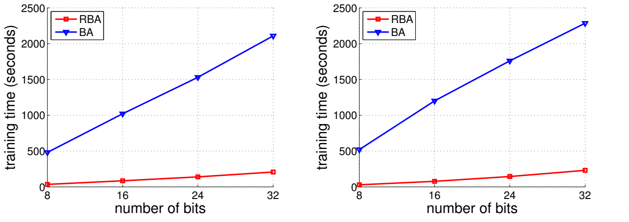
3.3.1 Dataset and evaluation protocol

Dataset We conduct experiments on CIFAR10 [24], MNIST [27] and SIFT1M [19] datasets which are widely used in evaluating hashing methods [12, 6].

CIFAR10 dataset [24] consists of 60,000 images of 10 classes. The dataset is split into training and test sets, with $50,000$ and $10,000$ images, respectively. Each image is represented by 320 dimensional GIST feature [35].

MNIST dataset [27] consists of 70,000 handwritten digit images of 10 classes. The dataset is split into training and test sets, with $60,000$ and $10,000$ images, respectively. Each image is represented by a 784 dimensional gray-scale feature vector.

SIFT1M dataset [19] contains 128 dimensional SIFT vectors [31]. There are 1M vectors used as database for retrieval, 100K vectors for training, and 10K vectors for query.



(a) CIFAR10 (b) SIFT1M

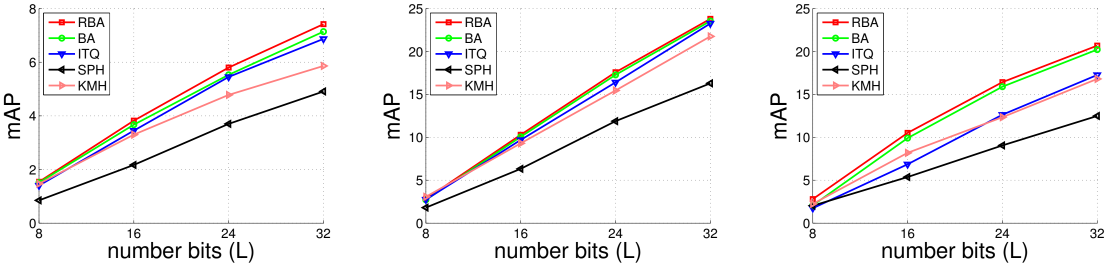
Figure 1. Training time of BA and RBA on CIFAR10 and SIFT1M

Evaluation protocol In order to create ground truth for queries, we follow [12, 6] in which the Euclidean nearest neighbors are used. The number of ground truths is set as in [6]. For each query in CIFAR10 and MNIST datasets, its 50 Euclidean nearest neighbors are used as ground truths; for each query in the large scale dataset SIFT1M, its 10,000 Euclidean nearest neighbors are used as ground truths. Follow the state of the art [12, 6], the performance of methods is measured by mAP. Note that as computing mAP is slow on the large scale dataset SIFT1M, we consider top 10,000 returned neighbors when computing mAP.

3.3.2 Experimental results

Training time of RBA and BA In this experiment, we empirically compare the training time of RBA and BA. The experiments are carried out on a processor core (Xeon E5-2600/2.60GHz). It is worth noting that the implementation of RBA is in Matlab, while BA optimizes the implementation by using mex-files at the encoder learning step. The comparative training time on CIFAR10 and SIFT1M datasets is showed in Figure 1. The results show that RBA is more than ten times faster training than BA for all code lengths on both datasets. The training time of BA is almost linear to the number of bits. This can be explained as follows: the most training time of BA is to solve the encoder and $\mathbf{Z}$. For both problems, they solve each bit separately (Section 2), i.e., for encoder, they learn L SVMs; for $\mathbf{Z}$, they check the optimum value of each bit sequentially.

Retrieval results Figure 2 shows the comparative mAP between methods. We find the following observations are consistent for all three datasets. At all code lengths, the proposed RBA outperforms or is competitive with the state-of-the-art BA. This result confirms the advance of our approach for computing encoder (i.e., closed-form) and **B-step** (i.e. using coordinate descent with closed-form for each row). The results in Figure 2 also confirm the superior performance of BA and RBA over other methods. The improvements are more clear on the large scale SIFT1M dataset.



(a) CIFAR10 (b) MNIST (c) SIFT1M

Figure 2. mAP comparison between RBA and state-of-the-art unsupervised hashing methods on CIFAR10, MNIST, and SIFT1M

4. Simultaneous Feature Aggregating and Hashing (SAH)

4.1. Formulation

Our goal is to simultaneously learn the aggregated vector representing an image and the hashing function, given the set of local image representations. For simultaneous learning, the learned aggregated vectors and the hash parameters should ensure desired properties of both aggregating and hashing. Specifically, *aggregating property*: (i) for each image i, the dot-product similarity between the aggregated vector φ_i and each local vector of $\mathbf{V}_i$ should be a constant; *hashing properties*: (ii) the outputs of the encoder are binary and (iii) the binary codes should preserve the similarity between image representations. In order to achieve these properties, we formulate the simultaneous learning as the following optimization

$$\min_{\mathbf{W}_1,\mathbf{c}_1,\mathbf{W}_2,\mathbf{c}_2,\Phi} \frac{1}{2}\left\|\Phi - \left(\mathbf{W}_2(\mathbf{W}_1\Phi + \mathbf{c}_1\mathbf{1}^T) + \mathbf{c}_2\mathbf{1}^T\right)\right\|^2$$
$$+\frac{\beta}{2}\left(\|\mathbf{W}_1\|^2 + \|\mathbf{W}_2\|^2\right) + \frac{\gamma}{2}\sum_{i=1}^{m}\left(\left\|\mathbf{V}_i^T\varphi_i - \mathbf{1}\right\|^2 + \mu\|\varphi_i\|^2\right) \quad (16)$$

$$\text{s.t. } \mathbf{W}_1\Phi + \mathbf{c}_1\mathbf{1}^T \in \{-1,1\}^{L\times m} \quad (17)$$

The first term of (16) ensures a good reconstruction of Φ, hence it encourages the similarity preserving (the property iii). The binary constraint (17) ensures the binary outputs of encoder (the property ii). Finally, the third term ensures the learned aggregated representation equals the similarities between φ_i and different columns of $\mathbf{V}_i$ by forcing their inner product to be 1 (the property i).

4.2. Optimization

In order to solve (16) under constraint (17), we propose to iteratively optimize it by alternatingly optimizing w.r.t. hashing parameters $(\mathbf{W},\mathbf{c})$ and aggregated representation Φ while holding the other fixed.

Φ-step: When fixing $\mathbf{W}_1,\mathbf{c}_1,\mathbf{W}_2,\mathbf{c}_2$ and solving for Φ, we can solve over each φ_i independently. Specifically, for each sample $i = 1,...,m$, we solve the following relaxed problem by skipping the binary constraint

$$\min_{\varphi_i} \frac{1}{2}\|\varphi_i - (\mathbf{W}_2(\mathbf{W}_1\varphi_i + \mathbf{c}_1) + \mathbf{c}_2)\|^2$$
$$+\frac{\gamma}{2}\left(\left\|\mathbf{V}_i^T\varphi_i - \mathbf{1}\right\|^2 + \mu\|\varphi_i\|^2\right) \quad (18)$$

By solving (18), we find φ_i which satisfies the properties (i) and (ii), i.e., φ_i not only ensures the aggregating property but also minimize the reconstruction error w.r.t. the fixed hashing parameters. (18) is actually a l_2 regularized least squares problem, hence we achieve the analytic solution as

$$\varphi_i = \left((\mathbf{I} - \mathbf{W}_2\mathbf{W}_1)^T(\mathbf{I} - \mathbf{W}_2\mathbf{W}_1) + \gamma\mathbf{V}_i\mathbf{V}_i^T + \gamma\mu\mathbf{I}\right)^{-1}$$
$$\times\left(\gamma\mathbf{V}_i\mathbf{1} + (\mathbf{I} - \mathbf{W}_2\mathbf{W}_1)^T(\mathbf{W}_2\mathbf{c}_1 + \mathbf{c}_2)\right) \quad (19)$$

The asymptotic complexity for computing (19) is $\mathcal{O}(\max(D^3, D^2 n_i))$ which is similar to the asymptotic complexity for computing (2).

$(\mathbf{W},\mathbf{c})$-step: When fixing Φ and solving for $(\mathbf{W}_1,\mathbf{c}_1,\mathbf{W}_2,\mathbf{c}_2)$, (16) under the constraint (17) is equivalent to the following optimization

$$\min_{\{\mathbf{W}_i,\mathbf{c}_i\}_{i=1}^2} \frac{1}{2}\left\|\Phi - \left(\mathbf{W}_2(\mathbf{W}_1\mathbf{X} + \mathbf{c}_1\mathbf{1}^T) + \mathbf{c}_2\mathbf{1}^T\right)\right\|^2$$
$$+\frac{\beta}{2}\left(\|\mathbf{W}_1\|^2 + \|\mathbf{W}_2\|^2\right) \quad (20)$$
$$\text{s.t. } \mathbf{W}_1\Phi + \mathbf{c}_1\mathbf{1}^T \in \{-1,1\}^{L\times m} \quad (21)$$

By solving (20) under the constraint (21), we find hash parameters which satisfy the properties (ii) and (iii), i.e., they not only ensure the binary outputs of the encoder but also minimize the reconstruction error w.r.t. the fixed aggregated representation Φ. (20) and (21) have same forms as (5) and (6), so we solve this optimization with the proposed Relaxed Binary Autoencoder (Section 3). We use the Algorithm 1 for solving $(\mathbf{W}_1,\mathbf{c}_1,\mathbf{W}_2,\mathbf{c}_2)$ in which Φ is used as the training data.

The proposed simultaneous feature aggregating and hashing is presented in the Algorithm 2. In the Algorithm 2, $\Phi^{(t)}$, $\mathbf{W}_1^{(t)}$, $\mathbf{c}_1^{(t)}$, $\mathbf{W}_2^{(t)}$, $\mathbf{c}_2^{(t)}$ are values at t^{th} iteration. After learning $\mathbf{W}_1,\mathbf{c}_1,\mathbf{W}_2,\mathbf{c}_2$, given set of local features of a new image, we first compute its aggregated representation φ using (19). We then pass φ to the encoder to compute the binary codes.

Algorithm 2 Simultaneous feature Aggregating and Hashing (SAH)

```
Input:
    $\mathcal{V} = \{\mathbf{V}_i\}_{i=1}^m$: training data; $L$: code length; $T, T_1$: maximum iteration numbers for SAH and RBA (Algorithm 1), respectively; parameters $\lambda, \beta, \gamma, \mu$.
Output:
    Parameters $\mathbf{W}_1, \mathbf{c}_1, \mathbf{W}_2, \mathbf{c}_2$

Initialize $\Phi^{(0)} = \{\varphi_i\}_{i=1}^m$ with Generalized Max Pooling (2)
for $t = 1 \rightarrow T$ do
    Fix $\Phi^{(t-1)}$, solve $(\mathbf{W}_1^{(t)}, \mathbf{c}_1^{(t)}, \mathbf{W}_2^{(t)}, \mathbf{c}_2^{(t)})$ using Algorithm 1
    Fix $(\mathbf{W}_1^{(t)}, \mathbf{c}_1^{(t)}, \mathbf{W}_2^{(t)}, \mathbf{c}_2^{(t)})$, solve $\Phi^{(t)}$ using $\Phi$-step.
end for
Return $\mathbf{W}_1^{(T)}, \mathbf{W}_2^{(T)}, \mathbf{c}_1^{(T)}, \mathbf{c}_2^{(T)}$
```

5. Evaluation of Simultaneous Feature Aggregating and Hashing (SAH)

This section evaluates and compares the proposed SAH to the following state-of-the-art unsupervised hashing methods: Iterative Quantization (ITQ) [12], Binary Autoencoder (BA) [6] and the proposed RBA, Spherical Hashing (SPH) [15], K-means Hashing (KMH) [14]. For all compared methods, we use the implementations and the suggested parameters provided by the authors. The values of λ, β, γ, and μ are set by cross validation as 10^{-2}, 10^{-1}, 10, and 10^2, respectively.

5.1. Dataset

We conduct experiments on Holidays [18] and Oxford5k [37] datasets which are widely used in evaluating image retrieval systems [22, 2, 20, 9].

Holidays The Holidays dataset [18] consists of 1,491 images of different locations and objects, 500 of them being used as queries. Follow [22, 9], when evaluating, we remove the query from the ranked list. For the training dataset, we follow [22, 9], i.e., using 10k images from the independent dataset Flickr60k provided with Holidays.

Holidays+Flickr100k In order to evaluate the proposed method on large scale, we merge Holidays dataset with 100k images downloaded from Flickr [17], forming the Holidays+Flickr100k dataset. This dataset uses the same training dataset with Holidays.

Oxford5k The Oxford5k dataset [37] consists of 5,063 images of buildings and 55 query images corresponding to 11 distinct buildings in Oxford. We follow standard protocol [22, 2]: the bounding boxes of the region of interest are cropped and then used as the queries. As standardly done in the literature, for the learning, we use the Paris6k dataset [38].

The ground truth of queries have been provided with the datasets [18, 37]. Follow the state of the art [12, 6], we evaluate the performance of methods with mAP.

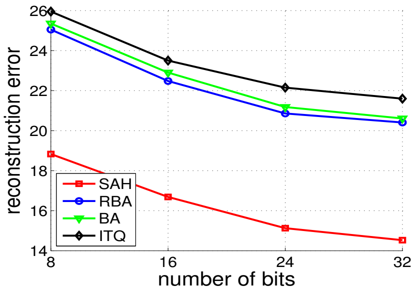


Figure 3. Reconstruction error comparison of different methods on Oxford5k dataset

5.2. Experiments with SIFT features

Follow state-of-the-art image retrieval systems [22, 20, 9], to describe images, we extract SIFT local descriptors [31] on Hessian-affine regions [32]. RootSIFT variant [1] is used in all our experiments. Furthermore, instead of directly using SIFT local features, as a common practice, we enhance their discriminative power by embedding them into high dimensional space (i.e., 1024 dimensions) with the state-of-the-art triangulation embedding [22]. As results, the set of triangulation embedded vectors $\mathcal{V} = \{\mathbf{V}_i\}_{i=1}^m$ is used as the input for the proposed SAH. In order to make a fair comparison to other methods, we aggregate the triangulation embedded vectors with GMP [33] and use the resulted vectors as input for compared hashing methods.

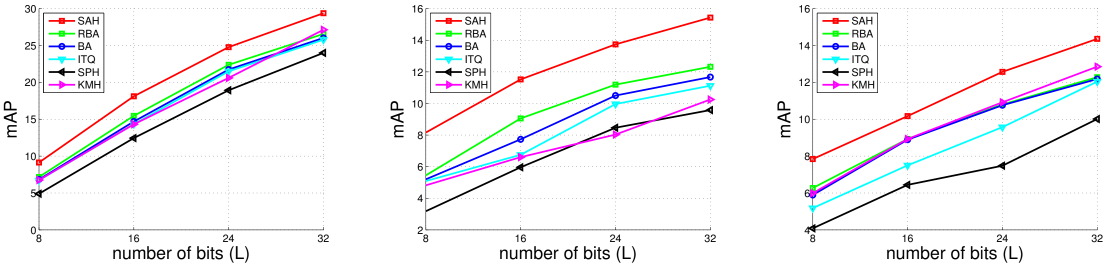
Reconstruction comparison In this experiment, we evaluate the reconstruction capacity of binary codes produced by different methods: ITQ [12], BA [6], RBA, and SAH. We compute the average reconstruction error on the Oxford5k dataset.

For ITQ, BA, and RBA, given the binary codes $\mathbf{Z}$ of the testing data (Oxford5k), the reconstructed testing data is computed by $\mathbf{X}_{res} = \mathbf{W}_2\mathbf{Z} + \mathbf{c}_2\mathbf{1}^T$, where $(\mathbf{W}_2, \mathbf{c}_2)$ is decoder. Note that the decoder is available in the design of BA/RBA and is learned in learning process. For ITQ, there is no decoder in its design, hence we follow [6], i.e., we compute the optimal linear decoder $(\mathbf{W}_2, \mathbf{c}_2)$ using the binary codes of the training data (Paris6k).

For SAH, given the binary codes $\mathbf{Z}$, we use the learned encoder and decoder to compute the aggregated representations Φ by using (19). The reconstruction of Φ is computed by using the decoder as $\Phi_{res} = \mathbf{W}_2\mathbf{Z} + \mathbf{c}_2\mathbf{1}^T$.

Figure 3 shows that BA and RBA are comparable while SAH dominates all other methods in term of reconstruction error. This confirms the benefit of the jointly learning of aggregating and hashing in the proposed SAH.

Retrieval results Figure 4 shows the comparative mAP between compared methods. We find the following observations are consistent on three datasets. The proposed RBA is competitive or slightly outperforms BA [6], especially on



(a) Holidays (b) Oxford5k (c) Holidays+Flickr100k

Figure 4. mAP comparison between SAH and state-of-the-art unsupervised hashing methods when using SIFT features on Holidays, Oxford5k, and Holidays+Flickr100k

Oxford5k dataset. The proposed SAH improves other methods by a fair margin. The improvement is more clear on Holidays and Oxford5k, e.g., SAH outperforms the most competitor RBA 2%-3% mAP at all code lengths.

5.3. Experiments with CNN feature maps

Recently, in [42, 4, 3] the authors showed that the activations from the convolutional layers of a convolutional neural network (CNN) can be interpreted as local features describing image regions. Motivated by those works, in this section we perform the experiments in which activations of a convolutional layer from a pre-trained CNN are used as an alternative to SIFT features. It is worth noting that our work is the first one that evaluates hashing on the image representation aggregated from convolutional features. Specifically, we extract the activations of the 5^{th} convolutional layer (the last convolutional layer) of the pre-trained VGG network [41]. Given an image, the activations form a 3D tensor of $W \times H \times C$, where $C = 512$ which is number of feature maps and $W = H = 37$ which is spatial size of the last convolutional layer. By using this setting, we can consider that each image is represented by $1,369$ local feature vectors with dimensionality 512. In [4], the authors showed that the convolutional features are discriminative, hence the embedding step is not needed for these features. Therefore, we directly use the convolutional features as the input for the proposed SAH. In order to make a fair comparison between SAH and other hashing methods, we aggregate the convolutional features with GMP [33] and use the resulted vectors as the input for compared hashing methods.

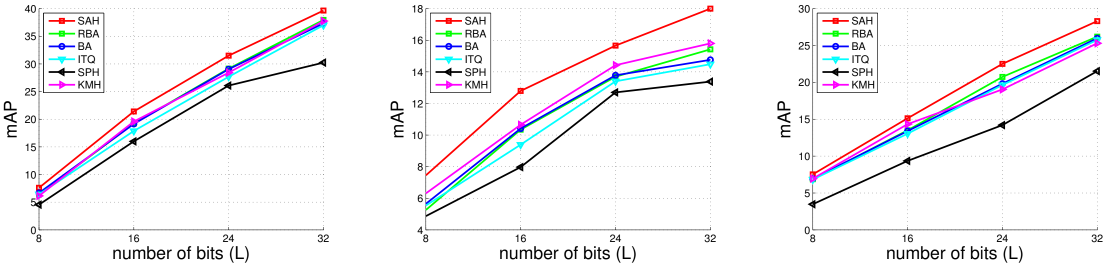
Retrieval results Figure 5 shows the comparative mAP between methods. We can see BA [6], KMH [14] and RBA achieve comparative results. It is clearly showed that the proposed SAH outperforms other methods by a fair margin. The improvements are more clear with longer code, e.g., SAH outperforms BA [6] 2%-3% mAP at $L = 32$ on three datasets. It is worth noting from Figure 5 and Figure 4 that at low code length, i.e., $L = 8$, SIFT features and convolutional features give comparable results. However, when increasing the code length, the convolutional features significantly improves over the SIFT features, especially on Holidays and Holidays+Flickr100k datasets. For example, for SAH on Holidays+Flickr100k, the convolutional features improves mAP over the SIFT features about 5%, 10%, 14% for $L = 16, 24$ and 32, respectively.

5.4. Comparison with fully-connected features

In [39], the authors showed that for image retrieval problem, using fully-connected features produced by a CNN outperforms most hand-crafted features such as VLAD [20], Fisher [36]. In this section, we compare the proposed SAH with state-of-the-art unsupervised hashing methods which take the fully-connected features (e.g. outputs of the 7^{th} fully-connected layer from the pre-trained VGG network [41]) as inputs. It is worth noting that there are few recent hashing methods which are based on end-to-end CNN, i.e., they jointly learn image representation and binary codes [26, 48, 47]. However, those works are for supervised hashing and they are incomparable to this work which focuses on unsupervised hashing. For our proposed SAH, we take the convolutional features of the same pre-trained VGG network as inputs to demonstrate the benefit of the jointly learning of aggregating and hashing.

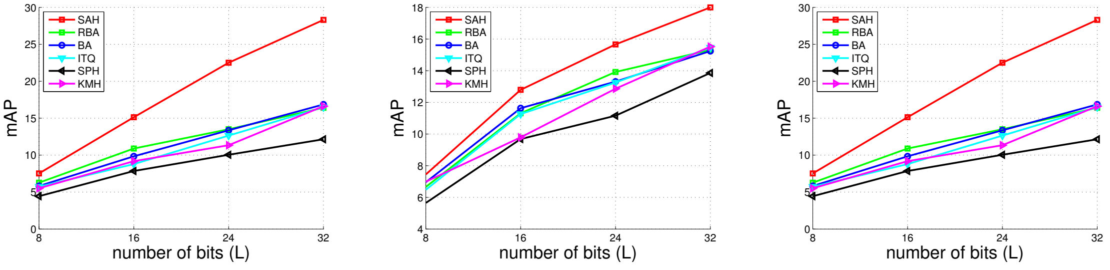
Retrieval results Figure 6 presents comparative mAP between methods. At low code length, i.e. $L = 8$, SAH is competitive to other methods. However, when increasing the code length, SAH outperforms compared methods a large margin. The significant improvements are shown on Holidays and Holidays+Flickr100k datasets, e.g., at $L = 32$, the improvements of SAH over BA [6] are 8% and 11.4% on Holidays and Holidays+Flickr100k, respectively.

Comparison with DeepBit [29] Recently, in [29], the authors proposed an end-to-end CNN-based unsupervised hashing approach. To the best of our knowledge, this is the only work using end-to-end CNN for unsupervised hashing. Starting with the pre-trained VGG network [41], they replaced the softmax layer of VGG with their binary layer and enforced several criteria on the binary codes learned at the



(a) Holidays (b) Oxford5k (c) Holidays+Flickr100k

Figure 5. mAP comparison between SAH and state-of-the-art unsupervised hashing methods when using convolutional features on Holidays, Oxford5k, and Holidays+Flickr100k



(a) Holidays (b) Oxford5k (c) Holidays+Flickr100k

Figure 6. mAP comparison between SAH and state-of-the-art unsupervised hashing methods using fully-connected features on Holidays, Oxford5k, and Holidays+Flickr100k

Table 2. Comparison between DeepBit [29] and other unsupervised hashing methods on CIFAR10. The results in the first four rows are cited from [29], which we have also reproduced.

Method	16 bits	32 bits	64 bits
ITQ [12]	15.67	16.20	16.64
KMH [14]	13.59	13.93	14.46
SPH [15]	13.98	14.58	15.38
DeepBit [29]	19.43	24.86	27.73
ITQ-CNN	38.52	41.39	44.17
KMH-CNN	36.02	38.18	40.11
SPH-CNN	30.19	35.63	39.23
SAH	**41.75**	**45.56**	**47.36**

binary layer, i.e., binary codes should: minimize the quantization loss with the output of the last VGG's fully connected layer, be distributed evenly, be invariant to rotation. Their network is fine-tuned using 50k training samples of CIFAR10. Note that as their approach is unsupervised, no label information is used during fine-tuning. Their comparative mAP of the top $1,000$ returned images (with the class labels ground truth) on the testing set of CIFAR10 is cited in the top part of Table 2. Note that their reported results of ITQ, KMH, SPH come from [10] in which GIST features are used. Therefore, we also evaluate those three hashing methods on the features extracted from the activations of the last fully connected layer of the same pre-trained VGG [41] (without fine-tuning). These results, i.e. ITQ-CNN, KMH-CNN, SPH-CNN, are presented in the bottom part of Table 2. It clearly shows that ITQ-CNN, KMH-CNN, SPH-CNN have significant improvements (using fully-connected instead of GIST). In order to evaluate the proposed SAH, we extract the activations of the last convolutional layer of the same pre-trained VGG and use them as input. The results of SAH presented in the last row in Table 2 show that at the same code length, SAH significantly outperforms the recent end-to-end work DeepBit [29], i.e., the mAP improvements are 22.3%, 20.7%, 19.6% at $L = 16$, 32 and 64, respectively. Furthermore, SAH also outperforms ITQ-CNN, KMH-CNN, SPH-CNN with a fair margin.

6. Conclusion

In this paper, we first introduce Relaxed Binary Autoencoder (RBA) hashing method in which instead of solving the hard binary constraint, we minimize the binary quantization loss. Compare to Binary Autoencoder, the proposed RBA achieves not only faster training but also competitive retrieval results. We then propose a novel unsupervised hashing approach called SAH by integrating feature aggregating and hash function learning into a joint optimization framework. Extensive experiments on benchmark datasets with SIFT, convolutional, and fully-connected features demonstrate that the proposed SAH method outperforms state-of-the-art unsupervised hashing methods.

References

[1] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[2] R. Arandjelovic and A. Zisserman. All about VLAD. In *CVPR*, 2013.

[3] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson. From generic to specific deep representations for visual recognition. In *CVPRW*, 2015.

[4] A. Babenko and V. S. Lempitsky. Aggregating local deep features for image retrieval. In *ICCV*, 2015.

[5] Y. Boureau, J. Ponce, and Y. LeCun. A theoretical analysis of feature pooling in visual recognition. In *ICML*, 2010.

[6] M. A. Carreira-Perpinan and R. Raziperchikolaei. Hashing with binary autoencoders. In *CVPR*, 2015.

[7] T.-T. Do, A.-D. Doan, and N.-M. Cheung. Learning to hash with binary deep neural network. In *ECCV*, 2016.

[8] T.-T. Do, A.-D. Doan, D.-T. Nguyen, and N.-M. Cheung. Binary hashing with semidefinite relaxation and augmented lagrangian. In *ECCV*, 2016.

[9] T.-T. Do, Q. Tran, and N.-M. Cheung. FAemb: a function approximation-based embedding method for image retrieval. In *CVPR*, 2015.

[10] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *CVPR*, 2015.

[11] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, 2013.

[12] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[13] K. Grauman and R. Fergus. Learning binary hash codes for large-scale image search. *Machine Learning for Computer Vision*, 2013.

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

[15] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing. In *CVPR*, 2012.

[16] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon. Spherical hashing: Binary code embedding with hyperspheres. *TPAMI*, pages 2304–2316, 2015.

[17] H. Jégou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[18] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, pages 316–336, 2010.

[19] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, pages 117–128, 2011.

[20] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[21] H. Jégou, F. Perronnin, M. Douze, J. Sánchez, P. Pérez, and C. Schmid. Aggregating local images descriptors into compact codes. *TPAMI*, 2012.

[22] H. Jégou and A. Zisserman. Triangulation embedding and democratic aggregation for image search. In *CVPR*, 2014.

[23] S. Kim and S. Choi. Bilinear random projections for locality-sensitive binary codes. In *CVPR*, 2015.

[24] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

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

[26] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *CVPR*, 2015.

[27] Y. Lecun and C. Cortes. The MNIST database of handwritten digits. `http://yann.lecun.com/exdb/mnist/`.

[28] 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.

[29] K. Lin, J. Lu, C.-S. Chen, and J. Zhou. Learning compact binary descriptors with unsupervised deep neural networks. In *CVPR*, 2016.

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

[31] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, pages 91–110, 2004.

[32] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *IJCV*, pages 63–86, 2004.

[33] N. Murray and F. Perronnin. Generalized max pooling. In *CVPR*, 2014.

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

[35] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, pages 145–175, 2001.

[36] F. Perronnin and C. R. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[37] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[38] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*, 2008.

[39] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: An astounding baseline for recognition. In *CVPRW*, 2014.

[40] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *CVPR*, 2015.

[41] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, 2014.

[42] G. Tolias, R. Sicre, and H. Jégou. Particular object retrieval with integral max-pooling of CNN activations. In *ICLR*, 2016.

[43] J. Wang, W. Liu, S. Kumar, and S. Chang. Learning to hash for indexing big data - A survey. *CoRR*, 2015.

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

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

[46] J. Yang, K. Yu, Y. Gong, and T. S. Huang. Linear spatial pyramid matching using sparse coding for image classification. In *CVPR*, 2009.

[47] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Transactions on Image Processing*, pages 4766–4779, 2015.

[48] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*, 2015.