Local Feature Binary Coding for Approximate Nearest Neighbor Search

Li Liu
li2.liu@northumbria.ac.uk
Mengyang Yu
m.y.yu@ieee.org
Ling Shao
ling.shao@ieee.org

Computer Vision and Artificial Intelligence Group,
Department of Computer Science and Digital Technologies,
Northumbria University, UK

Abstract

The potential value of hashing techniques has led to it becoming one of the most active research areas in computer vision and multimedia. However, most existing hashing methods for image search and retrieval are based on global representations, e.g., GIST, which lack the analysis of the intrinsic geometric property of local features and heavily limit the effectiveness of the hash code. In this paper, we propose a novel supervised hashing method called Local Feature Binary Coding (LFBC) for projecting local feature descriptors from a high-dimensional feature space to a lower-dimensional Hamming space via compact bilinear projections rather than a single large projection matrix. LFBC takes the matrix expression of local features as input and preserves the feature-to-feature and image-to-class structures simultaneously. Experimental results on challenging datasets including Caltech-256, SUN397 and NUS-WIDE demonstrate the superiority of LFBC compared with state-of-the-art hashing methods.

1 Introduction

Learning to hash [4, 14, 15, 16, 17, 20, 25, 27, 28] has received substantial attention due to its potential in various applications such as data mining, pattern recognition and information retrieval. In these topics, we usually need to utilize hashing methods to embed high-dimensional data points into a similarity-preserved Hamming space with low-dimensional compact binary string, which can lead large efficiency gains of the memory storage requirements and simultaneously increase the retrieval speed. Roughly, current hashing techniques can be divided into two groups, the unsupervised methods and the supervised methods.

A most well-known unsupervised method is Locality-Sensitive Hashing (LSH) [4] which can preserve similarity information and map data points close in a Euclidean space to similar codes. Beyond that, principled linear projections like PCA Hashing (PCAH) [25] has been developed for better quantization. Spectral Hashing (SpH) [28] was proposed to preserve the data locality relationship [26] to keep neighbors in the input space as neighbors in the Hamming space. Besides, Anchor Graphs Hashing (AGH) [17] and Compressed Hashing (CH) [15] have also been effectively applied for large-scale data retrieval tasks as well. All these hashing techniques mentioned above are regarded as unsupervised methods which

DOI: https://dx.doi.org/10.5244/C.29.173

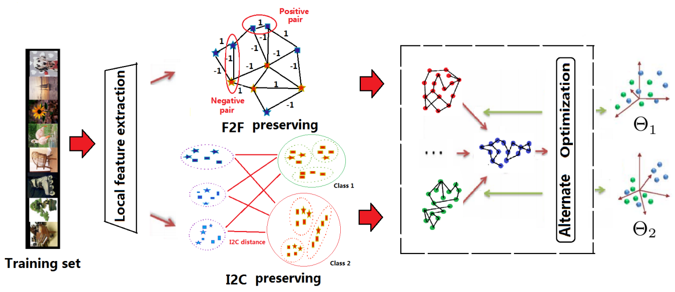


Figure 1: The illustration of the working flow of LFBC learning. The algorithm intends to preserve the pairwise F2F structure and the I2C distances and outputs the optimal bilinear projection matrices Θ_1 and Θ_2.

may lead to worse retrieval precision for the datasets with noise. To achieve better results, researchers have developed supervised hashing methods which could attain higher search accuracy, since the label information is involved in the learning phase. A simple supervised hashing method is Linear Discriminant Analysis Hashing (LDAH) [22] which can tackle supervision via easy optimization but still lacks adequate performance due to the use of orthogonal projection in hash functions. Beyond that, some more complicated methods have been proposed such as: Binary Reconstructive Embeddings (BRE) [14], Minimal Loss Hashing (MLH) [20] and Kernel Supervised Hashing (KSH) [18].

However, both conventional unsupervised and supervised hashing algorithms mentioned above are primarily designed for global representations, e.g., GIST [21]. For realistic visual retrieval tasks, however, these global hashing techniques cannot cope with different complications appearing in the images such as cluttering, scaling, occlusion and change of lighting conditions. To overcome this problem, local features such as SIFT [19] are usually adopted to represent images, which have proved to be more robust in challenging and noisy scenarios. Inspired by advantages of local representations, in this work, we intend to develop a local feature based hashing method for improving the retrieval results. If keypoints are well detected, local hash codes are able to avoid the limitations such as background variations, occlusions, and shifts in global representations.

In this paper, we propose an supervised hashing framework, i.e., Local Feature Binary Coding (LFBC), for visual similarity search, in which the feature-to-feature (F2F) and image-to-class (I2C) structures are successfully preserved and combined together. Specifically, the F2F structure considers the pairwise relationship between local features in the original feature space. While, from a higher-level aspect, I2C structure reflects the connection between images and their corresponding classes, which is derived from [2]. Besides, considering that our method is specifically designed for local feature based hashing, the original Hamming Ranking and Hamming Table cannot be directly applied on local features for visual indexing. Thus, in this paper, we also introduce an image indexing/searching scheme called Local Hashing Voting (LHV), which has been demonstrated to be efficient and accurate for image similarity search in our experiments. The outline of the proposed method is illustrated in Fig. 1.It is worthwhile to highlight several properties of the proposed method: (1) Different with global representation based hashing, LFBC directly learns hashing function from local features and simultaneously preserves pairwise F2F and I2C structure, which is proved to be more effective for accurate retrieval. (2) Since bilinear projection hashing function is adopted in our method, the complexity of the eigen-decomposition, which is the cubic form of the dimensionality, will be significantly reduced.

2 Related Work

In the literature, there are only a few works related to local feature based embedding and retrieval. A main work involving local feature coding was proposed in [10]. In particular, two schemes are introduced to improve the standard Bag-of-Words (BoW) model: (1) a Hamming embedding (HE) which provides binary signatures to refine visual words; (2) a weak geometric consistency constraint with the geometrical transformation. Both methods can significantly improve the final performance for retrieval tasks. Furthermore, a coupled Multi-Index (c-MI) framework was proposed for accurate image retrieval [32]. Beyond that, a selective match kernel approach [23] has also been developed to incorporate matching kernels sharing the best properties of HE and VLAD. Another related work based on [10] can be seen in [29], which introduces a color binary descriptor being calculated in either a global or a local form. However, all the above methods mainly focus on the retrieval techniques rather than the learning procedure of the binary coding for large-scale hashing. Besides, these methods are not fully linear, which limits their efficiency and applicability for large-scale datasets. To further improve these problems, in this paper, a bilinear projection based hashing method LFBC has been proposed on local features for improved image retrieval.

3 Local Feature Binary Coding

We are given N local features $\mathbf{x}_1, \cdots, \mathbf{x}_N \in \mathbb{R}^D$ from n images. For image i, we use $\mathcal{X}_i = \{\mathbf{x}_{i1}, \cdots, \mathbf{x}_{im_i}\}$ to represent its local feature set. Inspired by [6, 31], in our method, we aim to learn bilinear projections to effectively encode these local features, which can effectively reduce the complexity in optimization for large-scale tasks. Meanwhile, the learned projection matrices are much smaller and the projection speed is much faster compared with the conventional single projection. In particular, the bilinear projection matrices are multiplied on both sides of data. It can explore the matrix structure of features to enhance the effectiveness of projection. Firstly, we factor integer D as $D = D_1 \times D_2$. Then we reorganize vector $\mathbf{x}_i$ into matrix $X_i \in \mathbb{R}^{D_1 \times D_2}$ such that $\mathrm{vec}(X_i) = \mathbf{x}_i$, where $\mathrm{vec}(\cdot)$ represents the vectorization of a matrix. And we also have the inverse map of vectorization $\mathrm{vec}^{-1}(\mathbf{x}_i) = X_i$, since the vectorization is a one-to-one correspondence if D_1 and D_2 are given. To make the transformation more efficient, in this matrix form of local features, we define our hash function using two matrices $\Theta_1 \in \mathbb{R}^{D_1 \times d_1}$ and $\Theta_2 \in \mathbb{R}^{D_2 \times d_2}$:

$$H(X_i) = \mathrm{sgn}\left(\mathrm{vec}(\Theta_1^T X_i \Theta_2)\right) \in \{-1, +1\}^{d_1 d_2}. \tag{1}$$

It is noticeable that during the code learning stage, we use $\{-1, +1\}$ to encode local features and employ centralized data $\mathbf{x}_i - \frac{1}{N}\sum_{j=1}^{N} \mathbf{x}_j$ instead of $\mathbf{x}_i$, $\forall i$.

In fact, we notice that $\mathrm{vec}(\Theta_1^T X_i \Theta_2) = (\Theta_2^T \otimes \Theta_1^T)\mathrm{vec}(X_i) = (\Theta_2^T \otimes \Theta_1^T)\mathbf{x}_i$, where $\otimes$ is the Kronecker product, thus a bilinear projection is simply a special case of the single matrix projection Θ which can be decomposed as $\Theta = \Theta_2 \otimes \Theta_1$. Besides, it is easy to show that if Θ_1 and Θ_2 are orthogonal, i.e., $\Theta_1^T \Theta_1 = I_{d_1 \times d_1}$ and $\Theta_2^T \Theta_2 = I_{d_2 \times d_2}$, then Θ is orthogonal, as well. The bilinear projection leads to a more efficient eigen-decomposition on matrices with much smaller sizes $D_1 \times D_1$ and $D_2 \times D_2$ rather than $D_1 D_2 \times D_1 D_2$ for single projection.

Additionally, the space complexity for bilinear projections is $O(D_1^2 + D_2^2)$, while the single one needs $O((D_1 \times D_2)^2)$. Besides, since most of the local features are represented as concatenated histogram vectors, they can be intrinsically decomposed by two data structures. For instance, 128-dim SIFT is computed on 4×4 grids and for each grid a 8-bin histogram is calculated. In this way, a 128-dim SIFT is formed by concatenating $16 \times$ 8-bin histograms. Thus, for SIFT feature, we can naturally decomposed it via 16×8 in our bilinear codes learning.

Feature-to-feature (F2F) preserving: To obtain meaningful hash codes for local features, we first consider the geometric structure of the entire local feature set $\mathcal{F} = \{X_1, \cdots, X_N\}$. We are concerned about the individual relationship between local features in the original space, which should also be retained in the lower-dimensional space. Specifically, for similar (dissimilar) pairs, their distance is expected to be minimized (maximized) in the Hamming space. Since the class labels are unavailable for unsupervised method, we first use k-nearest neighbors (KNN) on $\mathcal{F}$ to obtain some weak label information. Then the pairwise label of (X_i, X_j) is defined as: $\ell_{ij} = +1$, if X_i and X_j are nearest neighbors; otherwise, $\ell_{ij} = -1$.

Since different pairs have different importance in the embedding, for pair (X_i, X_j), we assign a weight which is related to the pairwise distance with parameter σ:

$$W_{ij}^F = \exp\left(-\frac{\ell_{ij}+1}{2\sigma^2}\|X_i - X_j\|^2 + \frac{\ell_{ij}-1}{2\sigma^2\|X_i - X_j\|^2}\right),$$

where $\|\cdot\|$ is Frobenius norm. We can find that $W_{ij}^F \in (0,1)$ and for a positive pairwise label, W_{ij}^F is decreasing as the distance $\|X_i - X_j\|$ increases and vice versa. In other words, the positive pair is more important when they are close to each other, and the negative pair is more important when they are far away from each other. We denote $\mathcal{P} = \{(i,j) | X_i, X_j \in \mathcal{F}\}$. Therefore, preserving the F2F structure is to maximize

$$\sum_{(i,j)\in\mathcal{P}} W_{ij}^F \ell_{ij} \langle H(X_i), H(X_j) \rangle. \tag{2}$$

The above function reaches its maximum value when $W_{ij}^F \ell_{ij} H(X_i)$ and $H(X_j)$ are similarly sorted due to the rearrangement inequality [8].

Image-to-class (I2C) preserving: We are also concerned about a higher level connection, i.e., relationship between images and classes. Thus we consider a complete bipartite graph (a.k.a. bigraph) $G = (V_1, V_2, E)$ in which V_1 is the set of all images and V_2 is the set of all classes. The image-to-class (I2C) distance provides a feasible way to quantize the edges of E. Given the set of local features of an image $\mathcal{X}_i = \{X_{i1}, \cdots, X_{im_i}\}$, which contains all of local features of image i, the I2C distance between image i and class c is defined as: $D_{\mathcal{X}_i}^c = \sum_{X \in \mathcal{X}_i} \|X - \mathrm{NN}^c(X)\|^2$, where $\mathrm{NN}^c(X)$ is the nearest neighbor of the local feature X in class c.

However, searching the nearest neighbor in such a large scale space of local features of each class will still cost much time. Here, to reduce the complexity of searching, we employ K-means clustering algorithm on the set of local features of each class, i.e., $\bigcup_{C(\mathcal{X}_i)=c} \mathcal{X}_i$, $c = 1, \cdots, C$, where C is the number of classes and $C(\cdot) \in \{1, \cdots, C\}$ is the label information function that represents the class label of the input. And then we reduce the searching range of nearest neighbor to the cluster centers, i.e., for $c = 1, \cdots, C$, we let $\mathrm{NN}^c(X) \in$ Centroids $\{S_1, \cdots, S_K\}$ of $\bigcup_{C(\mathcal{X}_i)=c} \mathcal{X}_i$. Via I2C distances, we construct the bigraph $G = (V_1, V_2, E)$, where V_1 and V_2 represent the set of all the images and the set of all the classes, respectively. Then for each edge in E, we define its weight W_{ic}^{I2C}, named the I2C similarity between image i and class c in the original space, by the following Gaussian function

$$W_{ic}^{I2C} = \exp(-(D_{\mathcal{X}_i}^c)^2/2\sigma^2), \; i = 1, \cdots, n, \; c = 1, \cdots, C, \tag{3}$$

where σ is the Gaussian smooth parameter and n is the number of training samples. After applying LFBC, we have the I2C distance in Hamming space: $\widehat{D}_{\mathcal{X}_i}^c = \sum_{X \in \mathcal{X}_i} \|H(X) - \mathrm{NN}^c(H(X))\|^2$. In order to preserve the order of projected similarity in the projected space, a reasonable objective function for bigraph regularization is to minimize

$$\sum_{i=1}^{n} \sum_{c=1}^{C} \widehat{D}_{\mathcal{X}_i}^c \cdot W_{ic}^{I2C}. \tag{4}$$

One of the necessary conditions of the above function reaches the minimum value is that $\{\widehat{D}_{\mathcal{X}_i}^c\}$ is order-preserved due to the rearrangement inequality [8].

In addition, to make the projected space more compact, we set orthogonality constraints on the projection matrices Θ_1 and Θ_2, i.e., $\Theta_1^T \Theta_1 = I$ and $\Theta_2^T \Theta_2 = I$. Combined the F2F preserving part and the I2C preserving part with the orthogonality constraints, finally, we set our optimization problem as:

$$\underset{\Theta_1^T\Theta_1 = I, \; \Theta_2^T\Theta_2 = I}{\arg\max} \sum_{(i,j) \in \mathcal{P}} W_{ij}^F \ell_{ij} \langle \mathrm{sgn}(\widehat{X}_i), \mathrm{sgn}(\widehat{X}_j) \rangle - \gamma \sum_{i=1}^{n} \sum_{c=1}^{C} \widehat{D}_{\mathcal{X}_i}^c \cdot W_{ic}^{I2C}. \tag{5}$$

where γ is the regularization parameter.

Alternate Optimization via Relaxation: In this section, Motivated by [18, 28], to gain a solution of Eq. (5), we first relax the discrete sign function in optimization problem (5) to a real-valued continuous function by using its signed magnitude, i.e., $\mathrm{sgn}(x) \approx x$. In this case the F2F preserving part, becomes $\sum_{(i,j) \in \mathcal{P}} W_{ij}^F \ell_{ij} \mathrm{Tr}(\Theta_1^T X_i \Theta_2 \Theta_2^T X_j^T \Theta_1)$. For simplicity, we denote $\mathrm{NN}^c(X) = X^c$. Besides, we also make a statistical approximation on the computation of projected I2C distances due to the large amount of local features. That is, we exchange the operation of NN^c and $H(\cdot)$ for all $X \in \mathcal{X}_i$, i.e., $\sum_{X \in \mathcal{X}_i} \|H(X) - H(X)^c\|^2 \approx \sum_{X \in \mathcal{X}_i} \|H(X) - H(X^c)\|^2$. Thus, the projected I2C distance after applying matrices Θ_1 and Θ_2 becomes

$$\begin{aligned} \widehat{D}_{\mathcal{X}_i}^c &\approx \sum_{X \in \mathcal{X}_i} \|\Theta_1^T X \Theta_2 - \Theta_1^T X^c \Theta_2\|^2 = \sum_{X \in \mathcal{X}_i} \|\Theta_1^T (X - X^c) \Theta_2\|^2 \\ &= \sum_{k=1}^{m_i} \mathrm{Tr}(\Theta_1^T (X_{ik} - X_{ik}^c) \Theta_2 \Theta_2^T (X_{ik} - X_{ik}^c)^T \Theta_1) \\ &:= \sum_{k=1}^{m_i} \mathrm{Tr}(\Theta_1^T \Delta X_{ik}^c \Theta_2 \Theta_2^T (\Delta X_{ik}^c)^T \Theta_1), \end{aligned} \tag{6}$$

where $\Delta X_{ik}^c = X_{ik} - X_{ik}^c$, $k = 1, \cdots, m_i$.

Regarding the optimization scheme, two variables (Θ_1 and Θ_2) are defined in our objective function. For each single variable, the objective function is convex and analytic. However, for both variables, the objective function becomes a nonconvex optimization problem. To the best of our knowledge, there is no direct way to output the projections Θ_1 and Θ_2 simultaneously. Thus, we derive an alternate iteration algorithm [1] in this section. Specifically, for a fixed Θ_2, we can compute the optimal Θ_1 by solving a classical eigen-decomposition problem. And for the computation of Θ_2, we can then update Θ_2 by solving another eigen-decomposition problem with the computed Θ_1. First, let us denote the objective function in optimization problem (5) by $\mathcal{L}(\Theta_1, \Theta_2)$ and transform it to the following form by Eq. (6):

$$\begin{aligned}\mathcal{L}(\Theta_1,\Theta_2) = & \sum_{(i,j)\in\mathcal{P}} W_{ij}^F \ell_{ij} \operatorname{Tr}(\Theta_1^T X_i \Theta_2 \Theta_2^T X_j^T \Theta_1) \\ & -\gamma \sum_{i=1}^{n}\sum_{c=1}^{C}\sum_{k=1}^{m_i} W_{ic}^{I2C} \operatorname{Tr}(\Theta_1^T \Delta X_{ik}^c \Theta_2 \Theta_2^T (\Delta X_{ik}^c)^T \Theta_1),\end{aligned} \tag{7}$$

and

$$\begin{aligned}\mathcal{L}(\Theta_1,\Theta_2) = & \sum_{(i,j)\in\mathcal{P}} W_{ij}^F \ell_{ij} \operatorname{Tr}(\Theta_2^T X_j^T \Theta_1 \Theta_1^T X_i \Theta_2) \\ & -\gamma \sum_{i=1}^{n}\sum_{c=1}^{C}\sum_{k=1}^{m_i} W_{ic}^{I2C} \operatorname{Tr}(\Theta_2^T (\Delta X_{ik}^c)^T \Theta_1 \Theta_1^T \Delta X_{ik}^c \Theta_2),\end{aligned} \tag{8}$$

since $\operatorname{Tr}(AB) = \operatorname{Tr}(BA)$ if both products AB and BA exist. Then we denote

$$M_2(\Theta_2) = \sum_{(i,j)\in\mathcal{P}} W_{ij}^F \ell_{ij} X_i \Theta_2 \Theta_2^T X_j^T - \gamma \sum_{i=1}^{n}\sum_{c=1}^{C}\sum_{k=1}^{m_i} W_{ic}^{I2C} \Delta X_{ik}^c \Theta_2 \Theta_2^T (\Delta X_{ik}^c)^T, \tag{9}$$

and

$$M_1(\Theta_1) = \sum_{(i,j)\in\mathcal{P}} W_{ij}^F \ell_{ij} X_j^T \Theta_1 \Theta_1^T X_i - \gamma \sum_{i=1}^{n}\sum_{c=1}^{C}\sum_{k=1}^{m_i} W_{ic}^{I2C} (\Delta X_{ik}^c)^T \Theta_1 \Theta_1^T \Delta X_{ik}^c, \tag{10}$$

which are two matrix-valued functions with their codomains $\mathbb{R}^{D_1 \times D_1}$ and $\mathbb{R}^{D_2 \times D_2}$. In this way, we can rewrite function $\mathcal{L}$ as: $\mathcal{L}(\Theta_1,\Theta_2) = \operatorname{Tr}(\Theta_1^T M_2(\Theta_2)\Theta_1) = \operatorname{Tr}(\Theta_2^T M_1(\Theta_1)\Theta_2)$.

Although the number of the local features is relatively huge, the size of our final matrices M_1 and M_2 used for decomposition are small enough (D_1 and D_2 are always less than 100). This property mainly guarantees the efficiency and feasibility. Therefore, for $t = 0$, we randomly initialize $\Theta_2^{(t)}$; for the t-th step, we have the update rules:

$$\begin{aligned}\Theta_1^{(t)} & \leftarrow \text{the first } d_1 \text{ eigenvectors of } M_2(\Theta_2^{(t-1)}); \\ \Theta_2^{(t)} & \leftarrow \text{the first } d_2 \text{ eigenvectors of } M_1(\Theta_1^{(t)}).\end{aligned}$$

For any t, we have the inequality $\mathcal{L}(\Theta_1^{(t-1)},\Theta_2^{(t-1)}) \le \mathcal{L}(\Theta_1^{(t)},\Theta_2^{(t-1)}) \le \mathcal{L}(\Theta_1^{(t)},\Theta_2^{(t)})$. Thus $\mathcal{L}(\Theta_1^{(t)},\Theta_2^{(t)})$ is monotonic nondecreasing as $t \to \infty$. And continuous function $\mathcal{L}(\Theta_1,\Theta_2)$ is bounded in the closed district $\{(\Theta_1,\Theta_2) | \Theta_1^T\Theta_1 = I, \Theta_2^T\Theta_2 = I\}$. Then the above alternate iteration converges. In practice, we stop the iteration when $|\mathcal{L}(\Theta_1^{(t)},\Theta_2^{(t)}) - \mathcal{L}(\Theta_1^{(t-1)},\Theta_2^{(t-1)})|$ is less than a very small threshold (*Note:* LFBC can always coverage within 10 iterations).

4 Indexing via Local Hashing Voting

Once the bilinear projection matrices $\{\Theta_1, \Theta_2\}$ are obtained, we can easily embed the training data into binary hash codes by Eq. (1). Particularly, for a query local feature $\widehat{\mathbf{x}}$, its hash

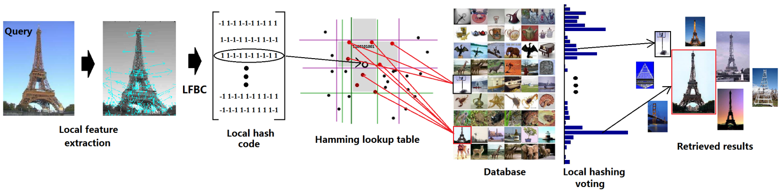


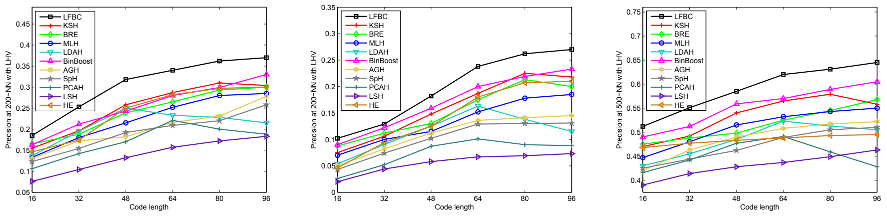
Figure 2: The illustration for the proposed LHV scheme. Given a query image, its local features are first extracted and embedded into hash codes via LFBC. Then, each hash code (e.g., "1 1 -1 -1 1 -1 1 -1 -1 1") corresponding to a local feature in the query image is then searched in the Hamming lookup table within the Hamming radius r and the corresponding images' indices are obtained. Finally, we vote and accumulate the times of each image's indices appearing in relevant buckets and rank them to return the retrieved results.

code is obtained by $H(\widehat{X}) = \mathrm{sgn}(\mathrm{vec}(\Theta_1^T(\widehat{X} - \frac{1}{N}\sum_{j=1} X_j)\Theta_2))$, where $\widehat{X}$ is the matrix form of $\widehat{\mathbf{x}}$. However, for our local feature hashing scenario, traditional linear search (e.g., Hamming distance ranking) with complexity $O(N)$ is not fast any more, since N denotes the total number (at least 3M local features for a large-scale database) of local features. To accomplish the local feature based visual retrieval, in this paper, we introduce a fast indexing scheme via Local Hashing Voting (LHV) as shown in Fig. 2. We first construct the Hamming lookup table (a.k.a. the hashing table) into our LHV scheme which build a series of bucket containing the indices of the documents with the same hash code. Given a query, we can find the bucket of corresponding hash codes in near constant time $O(1)$, and return all the data in the bucket as the retrieval results.

After construction of the Hamming lookup table over the training set, we store the corresponding indices for the hash codes of all local features. For instance, given a bucket with hash code $[1,1,-1,-1,1,-1,1,-1,-1,1]$, we store the indices of the images, which contain the same local feature hash code with this bucket. In this way, we search the hash code $H(\mathrm{vec}^{-1}(\mathbf{q}_i))$ for each local feature $\mathbf{q}_k \in \mathcal{Q}$ in the query image $\mathcal{Q} = \{\mathbf{q}_1, \cdots, \mathbf{q}_m\}$ over the Hamming lookup table within Hamming radius r and return the possible images' indices. It is noteworthy that the same bucket in the Hamming lookup table may store the indices from different images. Finally, we vote and accumulate the times of each image's indices appearing in relevant buckets and then rank them in decreasing order. The final retrieved samples are returned according to the relevant ranking generated by LHV.

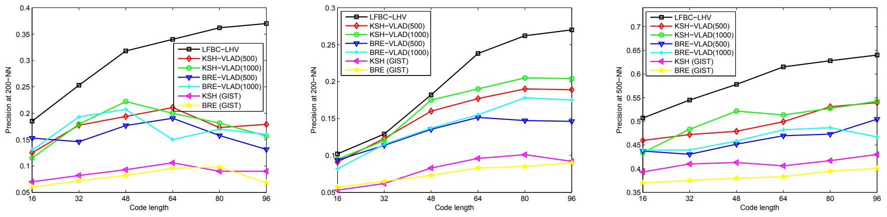
5 Experiments and Results

In this section, the proposed LFBC algorithm is evaluated for the image similarity search problem. Three different datasets are used in our experiments, i.e., Caltech-256 [7], SUN397 [30] and NUS-WIDE [3]. The **Caltech-256** dataset consists of 30607 images associated with 256 object categories. By following the experimental setting in [13], we randomly select 1000 images as the query set and the rest of dataset is regarded as the training set. The **SUN397** dataset contains 108,754 scene images in total from 397 well-sampled categories with at least 100 images per category. We randomly select 70 samples from each category to construct the training set and the rest of samples are the query set. Thus, there are total numbers of 27790 and 80964 in the training set and query set, respectively. The **NUS-WIDE** dataset consists of around 270,000 web images associated with 81 ground truth concept



(a) Caltech-256 (b) SUN397 (c) NUS-WIDE

Figure 3: Performance comparison with different numbers of bits.



(a) Caltech-256 (b) SUN397 (c) NUS-WIDE

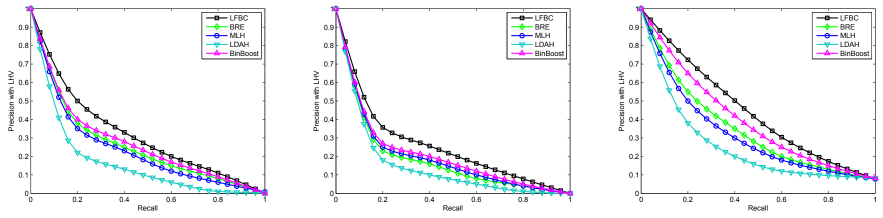
Figure 4: Performance comparison (MAP) of our LFBC and other global hashing schemes. Note: KSH and BRE are the top-performing supervised hashing methods.

classes. As in [17], we only use the most frequent 21 concept classes with a total number of 234,182 images and each of the used classes has abundant images ranging from $5,000$ to $30,000$. Unlike other datasets, each image in the NUS-WIDE dataset is assigned with multiple semantic labels (tags). In our work, two images belong to the same class, only if they share at least one common tag. We further sample uniformly 100 images from each of the selected 21 tags to form a query set of $2,100$ images with the rest serving as the training set. Furthermore, given an image, we would like to describe it with a set of local features extracted from it. In our experiments, we adopt 128-D SIFT[1] [19] as the local feature to describe the images and then learn to hash these local descriptors with all compared methods. Moreover, the labels assigned to the local features are consistent with the label of the image they come from. It is noteworthy that the training set of each dataset is used for not only training but also as a gallery database for similarity search in the querying phase.

In the querying phase, a returned point is regarded as a true neighbor if it lies in the top ranked 200, 200 and 500 points in LHV for Caltech-256, SUN397 and NUS-WIDE, respectively. Specifically in LHV, we just consider the local hash codes lying in the buckets that fall within a small Hamming radius $r = 2$ (following [18, 28]) in the Hamming lookup table which is constructed using the training set codes. We evaluate the retrieval results by the Mean Average Precision (MAP) and the precision-recall curve by changing the number of top ranked points in LHV. Our experiments are completed using Matlab 2013a on a server configured with a 12-core processor and 128G of RAM running the Linux OS.

Compared Methods and Settings: In our experiments, we compare the proposed method against nine general hashing algorithms, including five supervised methods: LDAH [22], BRE [14], MLH [20], KSH [18] and BinBoost descriptor [24], and six unsupervised methods: LSH [4], PCAH [25], SpH [28],Spherical hashing (SpherH) [9], Iterative quantization

[1]Assuredly our approach can also work with any other legitimate local features.



(a) Caltech-256 (b) SUN397 (c) NUS-WIDE

Figure 5: The comparison of precision-recall curves of the supervised algorithms on the three datasets with the code length of 96 bits.

(ITQ) [8] and AGH [17]. Besides, Hamming embedding (HE) [10], which is a nonlinear local feature based hashing method, is also included in our comparison. All the above methods (except BinBoost) are computed on the extracted SIFT local features to show their capability. It is noteworthy that we treat BinBoost as a binary feature learning method which learns to hash from the original local patches[2], however, the other methods construct the binary codes from extracted SIFT descriptors. We used the publicly available codes of BRE, MLH, LDAH, SpH, SpherH, ITQ and AGH, and implemented LSH, PCAH, KSH and BinBoost ourselves. All of the above methods are then evaluated on six different lengths of codes (16, 32, 48, 64, 80, 96). Under the same experimental setting, all the parameters used in the compared methods have been strictly chosen according to their original papers.

For our LFBC, we set $k = 15$ for pairwise data construction via KNN in F2F structure preserving. In this way, we assign the pairwise label $\ell_{ij} = +1$ if they are 15 nearest neighbors of each other in Euclidean space, for others, we assign the pairwise label $\ell_{ij} = -1$. Besides, we set $D_1 = 16$ and $D_2 = 8$ for the transformation of SIFT local features due to the natural composition of the $128dim$-SIFT features (can also be seen in Table 1). We further adopt $K = 300$ in the K-means clustering for I2C structure preserving. For LFBC, the optimal regularization parameter γ for each dataset is selected from one of $\{0.05, 0.1, \cdots, 0.5\}$ with step of 0.05, which yields the best performance by 10-fold cross-validation on training data.

Results Comparison: Fig. 3 shows the MAP curves of all compared algorithms on Caltech-256, SUN397 and NUS-WIDE datasets. All MAP values are calculated using the proposed LHV ranking algorithm under the same setting, since compared methods are directly used on local features. In general, the searching accuracies on the NUS-WIDE dataset are obviously higher than that on the other two datasets with more categories. Specifically, the supervised methods, KSH, BRE, MLH and BinBoost, always achieve the better performance than most of unsupervised methods, since the label information is involved in the learning phases. HE, as the only hashing method which is specifically designed for local features, can lead to competitive results compared with BRE in both Caltech-256 and SUN397 even though it is unsupervised. BinBoost has the best performance compared with all other existing methods and AGH achieves the best performance among all of the unsupervised methods. Furthermore, the results of KSH always climb up then go down when the length of code increases. The same tendency also appears with BRE, LDAH and PCAH. LSH consistently brings the worst actuaries on all these datasets. In general, our LFBC algorithm consistently outperforms all the compared methods in every length of code. This is because we consider the geometry structure of local features (F2F) and the global relationship from images

[2]Each patch is with the size of 4×4 located on the keypoints detected by SIFT.

Table 1: Result comparison (32 bits) via LFBC from different decompositions of SIFT features

Decomposition methods	Caltech-256	SUN397	NUS-WIDE
1×128	0.229	0.101	0.525
2×64	0.239	0.104	0.538
4×32	0.241	0.118	0.541
8×16	**0.253**	**0.129**	**0.551**

Table 2: Result comparison (32 bits) with/without F2F and I2C term in Eq. (5).

Methods	Caltech-256	SUN397	NUS-WIDE
Only F2F preserving	0.189	0.065	0.387
Only I2C preserving	0.227	0.104	0.432
F2F+I2C preserving (LFBC)	**0.253**	**0.129**	**0.551**

to class (I2C) simultaneously. Besides, in Table 1 we also illustrate the results computed via LFBC under different local feature decomposition. Since SIFT feature is intrinsically composed via 16 grid with 8-bin histograms, the best naturally bilinear decomposition i.e., $8 \times 16 = 128dim$, can achieve the better results than other decomposition ways. Table 2 shows the effectiveness of F2F and I2C term in 32 bits LFBC. It is observed only preserving F2F or I2C individually cannot achieve the best performance.

Besides, to make the comparison more convincing, we also compare with hashing on global representations. For all three datasets, we first use the K-means clustering to construct the codebooks with sizes of 500 and 1000 respectively and then encode SIFT features into global representations via vector of locally aggregated descriptors (VLAD) [11], which proves to be more discriminative than the original Bag-of-Words (BoW) representation. After that, two best performed hashing methods[3], i.e., KSH and BRE, are used to learn the hash codes on these global representations. Additionally, we also list the search performance via directly using the global feature GIST with KSH and BRE. In Fig. 4, one can see our local hashing method LFBC with LHV achieves better results than the compared global hashing schemes. Moreover, the precision-recall curves of all the compared methods on three datasets with the code length of 96 bits are presented in Fig. 5 as well. From all these figures, we can further discover that LFBC achieves better performance than other methods for both the Mean Average Precision (MAP) and Area Under the Curve (AUC) on all three datasets.

6 Conclusion

In this paper, we have presented a novel supervised framework, called Local Feature Binary Coding (LFBC), to learn highly discriminative binary codes on local descriptors for large-scale image similarity search. LFBC aims to seek orthogonal projection matrices for hashing, which can successfully preserve the pairwise similarity between different local features and simultaneously take image-to-class distances into consideration. We have systematically evaluated our methods on Caltech-256, SUN397, and NUS-WIDE datasets and show promising results compared with state-of-the-art hashing methods. In future work, it would be interesting to utilize our method on large-scale video dataset [12] for action retrieval tasks.

[3]BinBoost is not considered here, since it cannot be directly applied on extracted features.

References

[1] James C Bezdek and Richard J Hathaway. Some notes on alternating optimization. In *Advances in Soft Computing-AFSS*, pages 288–300. 2002.

[2] Oren Boiman, Eli Shechtman, and Michal Irani. In defense of nearest-neighbor based image classification. In *CVPR*, 2008.

[3] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yantao Zheng. Nus-wide: a real-world web image database from national university of singapore. In *ACM CIVR*, 2009.

[4] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999.

[5] Yunchao Gong and Svetlana Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[6] Yunchao Gong, Sanjiv Kumar, Henry A. Rowley, and Svetlana Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, 2013.

[7] Gregory Griffin, Alex Holub, and Pietro Perona. Caltech-256 object category dataset. 2007.

[8] Godfrey Harold Hardy, John Edensor Littlewood, and George Pólya. *Inequalities*. Cambridge university press, 1952.

[9] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. Spherical hashing. In *CVPR*, 2012.

[10] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[11] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[12] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[13] Saehoon Kim, Yoonseop Kang, and Seungjin Choi. Sequential spectral learning to hash with multiple representations. In *ECCV*. 2012.

[14] Brian Kulis and Trevor Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[15] Yue Lin, Rong Jin, Deng Cai, Shuicheng Yan, and Xuelong Li. Compressed hashing. In *CVPR*, 2013.

[16] Li Liu, Mengyang Yu, and Ling Shao. Multiview alignment hashing for efficient image search. *IEEE Transactions on Image Processing*, 24(3):956–966, 2015.

[17] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. Hashing with graphs. In *ICML*, 2011.

[18] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. Supervised hashing with kernels. In *CVPR*, 2012.

[19] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60 (2):91–110, 2004.

[20] Mohammad Norouzi and David M Blei. Minimal loss hashing for compact binary codes. In *ICML*, pages 353–360, 2011.

[21] Aude Oliva and Antonio Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJCV*, 42(3):145–175, 2001.

[22] Christoph Strecha, Alexander M Bronstein, Michael M Bronstein, and Pascal Fua. L-dahash: Improved matching with smaller descriptors. *T-PAMI*, 34(1):66–78, 2012.

[23] Giorgos Tolias, Yannis S. Avrithis, and Hervé Jégou. To aggregate or not to aggregate: Selective match kernels for image search. In *ICCV*, 2013.

[24] Tomasz Trzcinski, Mario Christoudias, Pascal Fua, and Vincent Lepetit. Boosting binary keypoint descriptors. In *CVPR*, 2013.

[25] Jun Wang, Sanjiv Kumar, and S Chang. Semi-supervised hashing for large scale search. *T-PAMI*, 34(12):2393–2406, 2012.

[26] Qi Wang, Pingkun Yan, Yuan Yuan, and Xuelong Li. Multi-spectral saliency detection. *Pattern Recognition Letters*, 34(1):34–41, 2013.

[27] Qi Wang, Guokang Zhu, and Yuan Yuan. Statistical quantization for similarity search. *Computer Vision and Image Understanding*, 124:22–30, 2014.

[28] Yair Weiss, Antonio Torralba, and Rob Fergus. Spectral hashing. In *NIPS*, 2008.

[29] Christian Wengert, Matthijs Douze, and Hervé Jégou. Bag-of-colors for improved image search. In *ACM Multimedia*, 2011.

[30] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010.

[31] Jieping Ye, Ravi Janardan, Qi Li, et al. Two-dimensional linear discriminant analysis. In *NIPS*, 2004.

[32] Liang Zheng, Shengjin Wang, Ziqiong Liu, and Qi Tian. Packing and padding: Coupled multi-index for accurate image retrieval. *CoRR*, abs/1402.2681, 2014.