Two Birds, One Stone: Jointly Learning Binary Code for Large-scale Face Image Retrieval and Attributes Prediction

Yan Li[1,2], Ruiping Wang[1], Haomiao Liu[1,2], Huajie Jiang[1,3], Shiguang Shan[1], Xilin Chen[1]
[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences (CAS),
Institute of Computing Technology, CAS, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
[3]School of Information Science and Technology, ShanghaiTech University, Shanghai, 200031, China
{yan.li, haomiao.liu, huajie.jiang}@vipl.ict.ac.cn, {wangruiping, sgshan, xlchen}@ict.ac.cn

Abstract

We address the challenging large-scale content-based face image retrieval problem, intended as searching images based on the presence of specific subject, given one face image of him/her. To this end, one natural demand is a supervised binary code learning method. While the learned codes might be discriminating, people often have a further expectation that whether some semantic message (e.g., visual attributes) can be read from the human-incomprehensible codes. For this purpose, we propose a novel binary code learning framework by jointly encoding identity discriminability and a number of facial attributes into unified binary code. In this way, the learned binary codes can be applied to not only fine-grained face image retrieval, but also facial attributes prediction, which is the very innovation of this work, just like killing two birds with one stone. To evaluate the effectiveness of the proposed method, extensive experiments are conducted on a new purified large-scale web celebrity database, named CFW 60K, with abundant manual identity and attributes annotation, and experimental results exhibit the superiority of our method over state-of-the-art.

1. Introduction

Content-based face image retrieval is concerned with automatic computer retrieval of face images of a given subject from a large-scale database with discriminative features. It emerges as a promising research direction with increasing demands, especially in the era of Internet multimedia involving huge body of face images. Imagine that people always wish to retrieve as many photos of a specific celebrity as possible, given one photo containing him/her as query; and it is also very helpful for police to search and locate a criminal suspect among a huge number of images captured by city surveillance system with a photo of him/her taken at the crime scene via CCTV. Both scenarios are extremely challenging, and the challenges mainly lie in two aspects: a) faces, especially the ones captured in the wild unconstrained environments always contain lots of variations, such as illumination, head pose, facial expression, occlusion, and sometimes heavy make-up, thus forming a phenomenon that the within-class variance is even larger than the between-class variance; b) the scale of database is in most cases so large that compact face representation becomes an urgent demand in order to reduce the storage cost and matching complexity.

For the first aspect, a number of methods have been proposed [29, 6, 1, 18] for conducting discriminant face representation. In [29] it utilizes different color spaces and standard PCA to extract discriminant facial representation for retrieval; in [6] a robust face retrieval approach is presented using structural and spatial point correspondence where the directional corner points (DCPs) are generated for efficient face coding; in [1] face image retrieval is conducted under a sparse coding based semi-supervised learning framework; in [18] it chooses Gabor-LBP histogram as facial representation followed by a sparse representation classifier for retrieval. While these works have achieved certain success mostly in well-controlled databases, they all have their face representation to be high-dimensional real-valued, which will inevitably limit their scalability to large-scale realistic retrieval scenario, which typically requires not only accurate but also compact representation for fast search. YES, binary code is a natural solution.

For the second aspect of the challenges, various binary code learning (a.k.a, hash learning) methods, have been proposed [7, 3, 13, 25, 37, 8, 11, 34, 26, 21], because binary codes are quite efficient to match with sub-linear time complexity, and are able to index a huge size of images with very short code length [22]. The basic idea of hashing is to learn similarity preserving binary codes for data representation, i.e., each data point will be hashed into a compact

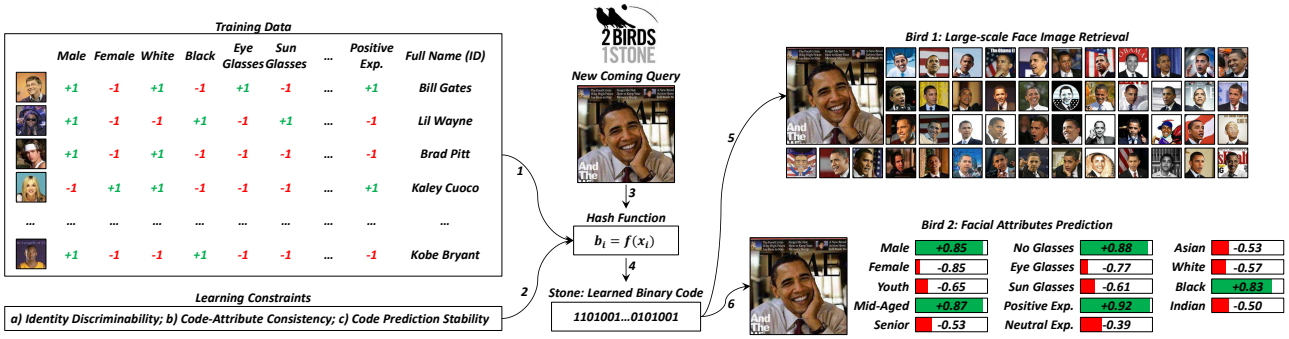

Figure 1: Schematic plot of the whole framework of the proposed method. Different from other binary code learning methods, we innovatively embed semantic facial attributes information into the code learning process to generate multipurpose binary codes, which can be used not only for face image retrieval, but also for facial attributes prediction.

binary string, and similar data points in the original feature space should be hashed into close points in the hash code space (always Hamming space). Existing hashing methods can be roughly divided into two types, i.e., *data independent* and *data dependent*. Representative data independent hashing methods include the pioneering Locality-Sensitive Hashing (LSH) [7] and its derivatives [3, 13], and Shift-Invariant Kernel Hashing (SIKH) [25]. Nevertheless, due to their inherent property that the original metrics are asymptotically preserved in the target Hamming space with increasing code length, LSH related methods usually achieve satisfactory performance at the expense of long codes. To overcome such disadvantage, another type, i.e., data dependent methods, have come up with their hash functions being learned from training data. Representative methods include Spectral Hashing (SH) [37], and ITerative Quantization (ITQ) [8], *etc*. Most recently, an increasing number of methods attempt to further boost performance by integrating discriminant supervised information into the hash functions learning, and then form a sub-branch of data dependent hashing methods, i.e., supervised hashing methods. Representative methods include Semi-Supervised Hashing (SSH) [34], Kernel-based Supervised Hashing (KSH) [21], Discriminative Binary Code (DBC) [26], and Supervised ITerative Quantization (SITQ) [8].

Discriminative binary code successfully solved the problem of large-scale data matching and storing with its compactness merit. Let's look a bit closer at the binary code. Each bit with value $+1$ or -1 can be naturally deemed as a discriminant "attribute" classifier, which indicates the presence or absence of such "attribute". While these "attributes" do have a certain level of discriminability, they generally carry little (if not nothing) semantic meaning that can be understood by human beings. Actually, attributes as a powerful mid-level representation have been investigated in a variety of computer vision tasks, including recognition, classification, image description and retrieval in recent years [5, 15, 17, 4, 16, 30, 14, 24, 28, 27, 38]. Firstly proposed in the computer vision community in [5], visual attributes can be automatically assigned to objects, scenes as text labels using standard machine learning techniques. After this pioneering work, later works have looked at visual attributes in the context of recognition [17, 4, 16, 28], zero-shot learning [17, 24, 38], automatic image description generating [14, 24], and image retrieval [15, 30, 27]. Besides, visual attributes also find successful applications in face recognition, verification, and retrieval [15, 16, 30, 27]. The most representative work is [16], where 65 facial attributes are defined (e.g., *male*, *female*, *sunglasses*, *wavy hair*, *mustache*), and the outputs of binary attribute classifiers are used as face feature representation. The reason for the success of visual attributes lies in that as a mid-level representation, they contain human-comprehensible semantic meaning that can effectively bridge the huge gap between low-level visual features and high-level vision tasks. Unfortunately, binary codes learned for retrieval do not have such helpful property yet. Therefore, we cannot help asking: is it possible to incorporate some semantic information into the binary code learning process, i.e., encoding a number of human-comprehensible facial attributes into the binary codes, e.g., gender, race, age, facial expression? If the answer is "yes", we can not only use the learned binary codes to retrieve face images of certain subjects, but also decode facial attributes information from the codes to carry out more high-level tasks, e.g., attribute prediction, or even retrieving face images with certain attributes.

Motivated by the analysis above, in this paper we make the first attempt to integrate the discriminative binary code learning and facial attributes encoding into a unified framework by jointly optimizing the code discriminability and code-attribute consistency, which turn out to be mutually helpful. By doing so, the learned codes can thus be used for two tasks at the same time, i.e., large-scale face image retrieval and facial attributes prediction, just like killing two birds with one stone. Fig. 1 shows the schematic plot of the proposed method. To evaluate the effectiveness of the proposed method, we conduct two groups of experiments on a new purified large-scale web celebrity database named

CFW-60K to respectively test the above two functionalities, i.e., face image retrieval and facial attributes prediction. Benefited from the inherent complementarity of identity discriminability and code-attribute consistency, we are glad to find that our method achieves comparable or even superior performance against the competing methods which are specifically tailored to either of the two functionalities.

2. Proposed Method

To learn the desirable binary codes mentioned before, we propose that three constraints need to be taken into consideration: a) ***identity discriminability***, i.e., the target Hamming space should be first discriminant, where the Hamming distance between images of the same category should be minimized, meanwhile images from distinct categories should better have quite different binary codes; b) ***code-attribute consistency***, i.e., the semantic attributes decoded from the binary codes should be consistent with the ground-truth probabilities of such attributes appearing in the corresponding images; c) ***code prediction stability***, i.e., images looking similar in the feature space should be mapped to similar hash codes within a short Hamming distance. Next, we will discuss each of them in detail.

2.1. Problem Description

Assume that we have N training images organized as $X = [x_1, x_2, ..., x_N] \in \mathbb{R}^{d \times N}$, where $x_i \in \mathbb{R}^d$ is the i^{th} image with a d-dimensional representation. For each training image x_i, we also have its ground-truth class label $l_i \in \{1, 2, ..., P\}$ indicating his/her identity, where P is the total number of categories. In addition to the identity labels, we also have the ground-truth annotation of T attributes for all N training images in a form of image-attribute annotation matrix, denoted by $A \in \{-1, 0, +1\}^{N \times T}$, where a_{it} indicates the presence, absence, or uncertainty of the t^{th} attribute in the i^{th} image with value $+1$, -1, or 0, respectively. Our target here is to obtain a linear hash function $f(x_i) : \mathbb{R}^d \rightarrow \{-1, +1\}^{K \times 1}$, which maps the image $x_i \in \mathbb{R}^d$ from the original feature to the binary code $b_i \in \{-1, +1\}^{K \times 1}$, where K denotes the length of binary code. Without loss of generality, in this paper we define the linear hash function $f(x_i)$ as follows:

$$f(x_i) = \mathrm{sgn}(W^T x_i) = b_i, \tag{1}$$

where $W \in \mathbb{R}^{d \times K}$ is the projection matrix, and sgn is the signum function. To measure the distance between binary codes b_i and b_j, here we use the classic normalized Hamming distance:

$$dis_H(b_i, b_j) = \frac{1}{4K} \|b_i - b_j\|^2. \tag{2}$$

2.2. Identity Discriminability

The identity discriminability of binary codes in Hamming space is expected most for reliable retrieval. To this end, we decompose the discriminability constraint into two components, i.e., intra-category compactness and inter-category separability. That is, images from the same category should have similar codes, and images from different categories should have better separability in the target Hamming space. Then we define the scatter measures of within-category S_w and between-category S_b as:

$$S_w = \sum_{i=1}^{N} \sum_{j=1}^{N} I_{ij}^w \|b_i - b_j\|^2, \tag{3}$$

$$S_b = \sum_{i=1}^{N} \sum_{j=1}^{N} I_{ij}^b \|b_i - b_j\|^2, \tag{4}$$

where I^w, I^b indicate the within- and between-category relationships of training images defined as follows.

$$I_{ij}^w = \begin{cases} 1 & \text{if } l_i = l_j \\ 0 & \text{otherwise} \end{cases}, \quad I_{ij}^b = \begin{cases} 1 & \text{if } l_i \neq l_j \\ 0 & \text{otherwise} \end{cases}. \tag{5}$$

Thus, to implement a strong discriminability, we minimize the following energy function E_{dis} formulated as

$$E_{dis} = S_w - \alpha S_b, \tag{6}$$

where α serves as the trade-off parameter for balancing the scale of S_w and S_b.

2.3. Code-Attribute Consistency

In this subsection, we discuss how to encode semantic visual attributes into the binary codes. Technically, we implement this encoding by exploring the correlation between visual attributes and the hash bits via the classic formulation of collaborative filtering [9, 23, 12, 33, 36] as follows:

$$E_{cons} = \sum_{t=1}^{T} \sum_{i=1}^{N} \|a_{it} - {b_i}^T u_t\|^2 + \lambda \sum_{t=1}^{T} \|u_t\|^2, \tag{7}$$

where $u_t \in \mathbb{R}^{K \times 1}$ indicates the correlation between the t^{th} visual attribute and the K-dimensional binary code (i.e., K hash bits), λ serves as a trade-off parameter on regularizer to avoid overfitting. The matrix form of u_t can be written as $U = [u_1, u_2, ..., u_T] \in \mathbb{R}^{K \times T}$, which plays the role of a bridge to connect the two views, i.e., visual attributes and hash code bits. Moreover, with this bridge, semantic attributes information can be reconstructed (predicted) from the K-bit binary code, by projecting the code from Hamming space to the T-dimensional attribute space as:

$$\hat{a}_i = {b_i}^T U, \qquad i \in \{1, 2, ..., N\}, \tag{8}$$

where $\hat{a}_i = [\hat{a}_{i1}, \hat{a}_{i2}, ..., \hat{a}_{iT}] \in \mathbb{R}^{1 \times T}$ is the predicted output of the T attributes for the i^{th} image.

Let's talk more about the image-attribute annotation matrix $A \in \{-1, 0, +1\}^{N \times T}$. Slightly different from the previous approaches [9, 23, 12, 33, 36] where an additional

confidence matrix is usually introduced to assign different weights for the elements in A to reflect the confidence of user-item annotation (usually A only contains two values, i.e., +1 and -1, which respectively indicate positive and negative/missing, and dominant weights are assigned to positive annotations), in this paper we extend the two-valued A to three-valued, i.e., +1, 0, -1, and equally treat every annotation without the need of the confidence matrix. The reason for introducing the uncertainty 0 is that we want to explicitly indicate ambiguity annotation and expect the learned attribute predictors to be highly consistent with human being perception on facial attributes, e.g., a well learned gender predictor should also "feel helpless" as human beings do when it comes to the face image of an infant (since it's really not an easy job for human beings to estimate the gender of an infant).

2.4. Code Prediction Stability

Code prediction stability (sometimes also called as generality, learnability, or predictability) is another crucial constraint in binary code learning. Intuitively, it is the concern about similarity preserving, i.e., visually similar images in the original feature space should be better mapped to similar hash codes within a short Hamming distance. Each hash function (binary code bit) can be viewed as a split in the original feature space, and we want the most stable splits. Specifically, a split is stable when it has large margins from samples (images) around it. Think about such a disappointing situation where a split crosses an area with dense samples, many actually neighboring samples will be inevitably assigned different hash values. Inspired by [26], we subtly utilize the classical technique, i.e., Support Vector Machine (SVM) [2], which is endowed with the max-margin property in hyperplane learning, to learn the linear hash function in Eqn. (1) and thus make the mapping more stable on unseen test samples. As a consequence, we describe the constraint of code prediction stability as an energy function E_{stab}:

$$E_{stab} = \sum_{k=1}^{K} \left\| w^k \right\|^2 + C \sum_{k=1}^{K} \sum_{i=1}^{N} \max(1 - b_i^k ({w^k}^T x_i), 0), \quad (9)$$

where $w^k \in \mathbb{R}^d, k \in \{1, 2, ..., K\}$ (corresponding to the k^{th} column of the projection matrix W in Eqn. (1)) denotes the k^{th} hyperplane (i.e., hash function), and C balances the empirical training error and the hyperplane margin.

2.5. Objective Function

After the analysis above, we can reach the final objective function by combining Eqn. (6), Eqn. (7), and Eqn. (9) to simultaneously consider the three learning constraints:

$$\min_{B,U,W} \quad E_{dis} + \beta E_{cons} + \gamma E_{stab}, \quad (10)$$

where B, U, W are the matrix forms of b_i, u_t, and w^k, i.e., $B = [b_1, b_2, ..., b_N] \in \{-1, +1\}^{K \times N}$, $U = [u_1, u_2, ..., u_T] \in \mathbb{R}^{K \times T}$, and $W = [w^1, w^2, ..., w^K] \in \mathbb{R}^{d \times K}$, and β, γ are the trade-off parameters to balance the roles of each term. By substituting Eqn. (6), Eqn. (7), and Eqn. (9), the objective function can be rewritten as follows:

$$\begin{aligned} \min_{B,U,W} \; & \sum_{i=1}^{N} \sum_{j=1}^{N} I_{ij}^w \|b_i - b_j\|^2 - \alpha \sum_{i=1}^{N} \sum_{j=1}^{N} I_{ij}^b \|b_i - b_j\|^2 \\ & + \beta \sum_{t=1}^{T} \sum_{i=1}^{N} \|a_{it} - b_i^T u_t\|^2 + \beta \lambda \sum_{t=1}^{T} \|u_t\|^2 \\ & + \gamma \sum_{k=1}^{K} \left\| w^k \right\|^2 + \gamma C \sum_{k=1}^{K} \sum_{i=1}^{N} \max(1 - b_i^k ({w^k}^T x_i), 0). \end{aligned} \quad (11)$$

Directly minimizing Eqn. (11) is intractable as it is a non-convex integer programming problem, therefore a block coordinate descent method is proposed in Section 3.

3. Optimization Algorithm

3.1. Iterative Optimization

Since the objective function Eqn. (11) is non-convex, it is infeasible to find a global analytical solution. In practice, we propose to utilize block coordinate descent method [31] to independently optimize each individual component to iteratively update B, U, and W. Technically, we first decompose the objective function into two sub-optimization problems as follows:

$$\begin{aligned} SOP_1: \min_{B,U} \; & \sum_{i=1}^{N} \sum_{j=1}^{N} I_{ij}^w \|b_i - b_j\|^2 - \alpha \sum_{i=1}^{N} \sum_{j=1}^{N} I_{ij}^b \|b_i - b_j\|^2 \\ & + \beta \sum_{t=1}^{T} \sum_{i=1}^{N} \|a_{it} - b_i^T u_t\|^2 + \eta \sum_{t=1}^{T} \|u_t\|^2, \end{aligned} \quad (12)$$

$$SOP_2: \min_{W} \sum_{k=1}^{K} \left\| w^k \right\|^2 + C \sum_{k=1}^{K} \sum_{i=1}^{N} \max(1 - b_i^k ({w^k}^T x_i), 0), \quad (13)$$

where η in Eqn. (12) is utilized to replace the parameter $\beta\lambda$ in Eqn. (11) for simplifying the formulation.

The proposed algorithm is conducted by iteratively optimizing SOP_1 and SOP_2 to update B, U, and W. The pseudo-code of the optimization can be found in Algorithm 1. Now, we give a detailed description. **First**, we initialize B by using PCA, which makes B have an orthogonal property, followed by the signum function (line 1∼line 2). **Second**, we use the initial B to update U by optimizing SOP_1 in Eqn. (12) (line 3). For SOP_1, we further simplify it by rewriting in a compact matrix form as:

$$\begin{aligned} SOP_1: \min_{B,U} \; & \mathrm{Tr}(BL^wB^T) - \alpha \mathrm{Tr}(BL^bB^T) \\ & + \beta \|A - B^T U\|_F^2 + \eta \|U\|_F^2, \end{aligned} \quad (14)$$

where L^w, L^b are the graph Laplacian matrices computed by $L^w = D^w - I^w$, $L^b = D^b - I^b$ (I^w and I^b are defined in

Eqn. (5), and they can be viewed as the adjacency matrices; and D^w and D^b are $N \times N$ diagonal degree matrices whose entries are given by $D^w_{ii} = \sum_{j=1}^{N} I^w_{ij}$ and $D^b_{ii} = \sum_{j=1}^{N} I^b_{ij}$), and $\|\cdot\|_F$ is the Frobenius norm of matrix. By taking the derivative of SOP_1 with respect to U, and setting the gradient to 0, we can obtain the closed form solution of U as follows:

$$\frac{\partial SOP_1}{\partial U} = 2\beta B(B^T U - A) + 2\eta U, \quad (15)$$

$$\tilde{U} = (BB^T + \lambda I)^{-1} BA, \quad (16)$$

where I is the $K \times K$ identity matrix. **Third**, we fix $\tilde{U}$ to update B by optimizing SOP_1 in Eqn. (12) (line 4). However, directly minimizing SOP_1 to optimize the binary-valued B as an integer programming problem has been proven to be NP-hard. Therefore, we convert the current optimization problem into a relaxed version by using the magnitude instead of the signum function as suggested in [21, 35], i.e., optimizing B by regarding it as a real-valued rather than binary-valued matrix with an additional Frobenius norm constraint (please find more details in supplementary material). The gradient of SOP_1 with respect to B is calculated as follows:

$$\frac{\partial SOP_1}{\partial B} = 2\beta U(U^T B - A^T) + 2BL^w - 2\alpha BL^b. \quad (17)$$

With this obtained gradient, a limited memory quasi-Newton method L-BFGS [20] is applied to minimize SOP_1. Please note that it is in this step, the identity discriminability is implemented. **Fourth**, we binarize the new updated real-valued B via signum function, and turn to SOP_2 for optimizing the hash functions W (line 5$\sim$line 6). As mentioned in Section 2.4, we directly use SVM to implement the optimization of SOP_2. More specifically, for the k^{th} bit, we use $b^k \in \{-1,+1\}^{1\times N}$ as training labels to train the k^{th} SVM hyperplane, i.e., the k^{th} hash function w^k. **Fifth**, we use the trained K SVMs to update B via Eqn. (1) (line 7). The whole optimization is looped from the second to fifth step by iteratively update B, U and W, and in practice we find that usually two or three iterations can achieve the convergence (please see supplementary material for more about convergency).

3.2. Discussion

Method Property: The idea of exploring binary code related semantic information can be traced back to [26] and [28]. Specifically, [26] tried to interpret the learned discriminative hash bits via 2-D visualization, however, it appears a bit farfetched to name an attribute implicitly only depending on visualization. [28] tried to augment semantic attributes with non-semantic ones, while the augmentation manner is straightforward via directly concatenating the two parts. Different from the previous works, our method

Algorithm 1 Optimization

Input: Training images $X = [x_1, x_2, ..., x_N] \in \mathbb{R}^{d\times N}$, image-attribute annotation matrix $A \in \{-1, 0, +1\}^{N\times T}$, and ground-truth labels $l_i \in \{1, 2, ..., P\}$, where $i \in \{1, 2, ..., N\}$.
Output: $B \in \{-1, 1\}^{K\times N}$, $U \in \mathbb{R}^{K\times T}$, $W \in \mathbb{R}^{d\times K}$.
Initialization:
1. Initialize B: $B \leftarrow$ Projection of X on first K components of $PCA(X)$;
2. Binarize B: $B \leftarrow \text{sgn}(B)$;
Repeat until convergence
3. Fix B to optimize U with Eqn. (16);
4. Fix U to optimize B with Eqn. (17);
5. Binarize B: $B \leftarrow \text{sgn}(B)$;
6. Train K SVMs to update W with Eqn. (13);
7. Update B with Eqn. (1);
End

for the first time attempts to integrate the binary code learning and attribute encoding into a whole framework, where the two tasks are twisted together and optimized iteratively (please refer to the experiment section for complementarity evaluation). In some sense, the proposed method can also be viewed as an extension of [26] with a complementary task by explicitly embedding semantic attributes into the code learning procedure, and it is expected such attempt would provide some insightful thoughts that it is theoretically feasible and of great significance to encode different messages into binary code, not only limited to identity discriminability.

Parameters Sensitivity: From our main objective function in Eqn. (10/11), a few parameters are observed to have potential influence on the performance of the learned binary code. Nevertheless, most of them have clear physical meanings and thus can be easily tuned. In particular, since the three components of Eqn. (10) are optimized separately in an iterative manner (Alg.1), β and γ mainly play the role of balancing each component, and can be simply set to equally weight those components. For the component E_{dis}, α can be simply set to normalize the pair numbers in S_w and S_b, and for the component E_{stab}, C is set to the default value 1 as standard SVM. Therefore, the only substantial parameter is the regularization coefficient λ for the component E_{cons}, which needs some effort to tune via a grid search with cross-validation and is selected as $\lambda = 6 \times 10^{-4}$ in our empirical study.

4. Experiment

To justify the two functionalities of the learned binary code, i.e., large-scale face image retrieval and facial attributes prediction. We conduct two groups of experiments respectively for each task, and all the experiments are conducted on a new purified large-scale web celebrity database, named by us CFW 60K.

4.1. Large-scale CFW 60K

Celebrity Faces on the Web (CFW) [39] is a large-scale database of celebrity face images collected from the Internet, and the released version contains about 200,000 faces of 1,500+ subjects. Compared with the popular LFW [10], each subject in CFW has more distinct images (average 100+ images for each subject), and these images include relatively more complex and real variations (some example images are shown in Fig. 1, Fig. 3, and Fig. 5, and please find more in supplementary material). However, the originally provided identity labels of CFW are generated automatically, which would inevitably involve a number of incorrect annotations. To fix such problem, for every face image we invite three annotators to check whether the claimed label is correct, and only images with three positive confirmations are preserved. Besides, the three annotators are also required to annotate five facial landmarks for each face (i.e., geometric centers of two eyes, tip of nose, and two corners of mouth). Finally, the purified CFW contains 153,461 faces of 1,520 subjects, and we name this version as CFW-p.

In this paper, we select 500 most famous subjects containing 60,000 images from CFW-p to form the CFW 60K database[1]. Moreover, we further select 10,000 images (20 images $\times$ 500 subjects) from CFW 60K to annotate 14 facial attributes including gender, race, age, eye accessory, and facial expression (please see Table 2 for details) with the help of seven annotators. Please note that the final attributes annotations are presented with three values, i.e., $+1$, -1, and 0, for respectively indicating the presence, absence, and uncertainty of a certain attribute.

4.2. Experimental Setting

As we categorize this work under the fields of binary code learning and semantic attributes embedding, we won't specially focus on the choice of front-end face representation. Therefore, without loss of generality, classic Gabor feature [19] is used to represent a face image in this paper. More specifically, for each image, we crop the face to a fixed size of 80×64, and block-wise Gabor feature with 5 scales and 8 orientations is extracted to form a 4000-dimensional feature.

4.3. Evaluation on Face Image Retrieval

Data Partition: Among the 10,000 images (with 14 facial attributes annotation), we take 5,000 images of them (10 images $\times$ 500 subjects) to form the training set. For testing, we use the 50,000 images (without facial attributes annotation) as database, and choose 2,500 images from them (5 images $\times$ 500 subjects) as query set.

Comparative Methods: Since our method in general is a binary code (hash) learning method, to evaluate its retrieval performance, we select several representative hashing methods for comparison in this part, including Locality-Sensitive Hashing (LSH) [7], Random Rotation (RR) [8], ITerative Quantization (ITQ) [8], Spectral Hashing (SH) [37], Semi-Supervised Hashing (SSH) [34], Discriminative Binary Code (DBC) [26], Kernel-based Supervised Hashing (KSH) [21], and Supervised ITerative Quantization (SITQ) [8]. For fair comparison, we use the same 4000-dimensional Gabor feature for all the comparative methods, and the important parameters of each method are empirically tuned according to the recommendations in the original references as well as the source codes provided by the original authors.

[1]The matlab implementation and database can be downloaded from http://vipl.ict.ac.cn/resources/codes.

Table 1: Comparison with state-of-the-art hash learning methods with different code lengths in mAP on CFW 60K.

Method	8 bits	16 bits	32 bits	64 bits	128 bits	256 bits
LSH [7]	0.0041	0.0162	0.0223	0.0295	0.0453	0.0707
RR [8]	0.0047	0.0181	0.0280	0.0406	0.0621	0.0822
ITQ [8]	0.0053	0.0172	0.0281	0.0424	0.0598	0.0792
SH [37]	0.0064	0.0170	0.0309	0.0410	0.0576	0.0851
SSH [34]	**0.0114**	**0.0284**	0.0435	0.0627	0.0871	0.1048
DBC [26]	0.0067	0.0216	0.0348	0.0506	0.0774	0.1133
KSH [21]	0.0088	0.0238	0.0422	0.0641	0.0917	0.1247
SITQ [8]	0.0055	0.0209	0.0341	0.0605	0.1151	0.1846
Ours	0.0089	0.0241	**0.0466**	**0.0771**	**0.1277**	**0.1979**

Measurements: For quantitative evaluation, we use the standard definition of mean Average Precision (mAP) [32] and the precision recall curve calculated among the range of whole database as measurements.

Results and Analysis: Table 1 and Fig. 2 show the comparison results in mAP and precision recall curve with different code lengths. It is obvious to find that: a) supervised hashing methods achieve higher retrieval performance than those unsupervised and semi-supervised methods, which can be mostly attributed to the full use of supervised information. Moreover, among those supervised hashing methods, SITQ is shown to have an edge over the others. One possible explanation is that the CCA preprocessing in SITQ perfectly embeds the two views (i.e., original features and identity labels) into a discriminative common space which is conducive to the subsequent code learning; b) SSH exhibits superior performance for small number of bits (i.e., 8, 16 bits) because there is enough vari-

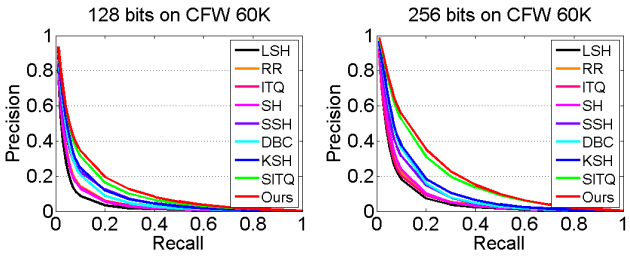

Figure 2: Comparison with state-of-the-art binary code learning methods with different code lengths in precision recall curve on CFW 60K.

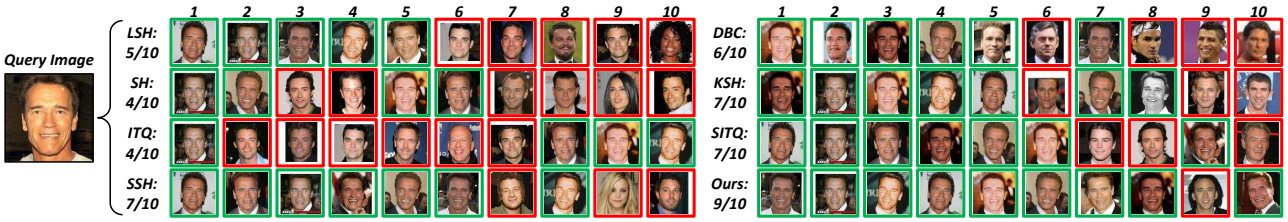

Figure 3: A real retrieval case on CFW 60K with 256 bits binary codes, where the query is a face image of *Arnold Schwarzenegger*. For space limitation, only the top ten feedbacks of each method are shown here (please find more cases in supplementary material).

Table 2: Comparison with classic attribute classifiers (Kumar *et al.* [16]) in classification accuracy. For our method, 256 bits are used in this comparison.

Attribute	Kumar *et al.*	Ours	Attribute	Kumar *et al.*	Ours
Male	0.9178	0.9400	*Mid-Aged*	0.6772	0.6712
Female	0.9178	0.9400	*Senior*	0.9170	0.9168
Asian	0.9460	0.9458	*No Glasses*	0.9480	0.9796
White	0.8198	0.8384	*Eye Glasses*	0.9640	0.9822
Black	0.9140	0.9228	*Sun Glasses*	0.9834	0.9832
Indian	0.9482	0.9276	*Positive Exp.*	0.6696	0.6342
Youth	0.6618	0.7276	*Neutral Exp.*	0.6732	0.6380

ance in the top few orthogonal directions computed in SSH (similar phenomenon can be found in the original literature of SSH); c) we are glad to find that although the proposed method simultaneously deals with the constraints of identity discriminability and code-attribute consistency (there is no doubt that this is more challenging than the optimization with only one constraint), it achieves comparable or even better performance than the other hashing methods; d) content-based large-scale face image retrieval is indeed an extremely challenging problem, and even the best hashing method can only achieve an mAP below 0.2 on CFW 60K with the current evaluation protocol. For this, two aspects should both be explored, i.e., more powerful front-end face representation (e.g., deep learning based features) and better discriminative binary code learning method for the back-end indexing (encoding as much discriminative information as possible into the binary codes). We also provide a real retrieval case on CFW 60K to exhibit our superiority in Fig. 3, and please find more cases in supplementary material. At last, we also compare the proposed method with two classic real-valued representation based face retrieval methods (both methods are carefully implemented by us), i.e., [29] (different color spaces+PCA with 4000-dimensional final representation: 0.1761) and [18] (Gabor-LBP+sparse coding with 5000-dimensional final representation: 0.2218). Although such comparison is unfair, i.e., 256 bits v.s. 4000/5000-dimensional, ours still achieves comparable performance (0.1979).

4.4. Evaluation on Facial Attributes Prediction

Data Partition: In this experiment, among the $10,000$ images (with 14 facial attributes annotation), we take the same $5,000$ images as Section 4.3 for training, and take the rest $5,000$ images for testing.

Comparative Method: To evaluate the performance on facial attributes prediction, we choose the classic method proposed in [16] for comparison, i.e., training SVM attribute classifier with RBF kernel for each attribute. We choose it because it has been proven to be not only simple but also powerful and practical [15, 16].

Measurement: In this experiment, we use the classification accuracy (the same to [16]) as measurement.

Results and Analysis: Table 2 shows the comparison result in classification accuracy on 14 facial attributes. Although only 256 bits are used, our method achieves comparable performance with the baseline method which directly utilizes the original 4000-dimensional real-valued Gabor feature. To better exhibit the performance of our method, four real cases are shown in Fig. 5. The reason behind such promising results is that the optimization of attribute predictor in our method (i.e., the U in Eqn. (8)) is conducted by simultaneously considering all the attributes (the baseline method treats each attribute individually), and the correlation between attributes can thus be embedded into the learning of attribute predictor. To further investigate the correlation between attributes and hash bits, we visualize the learned attribute predictor U in Fig. 4 (for space limitation, only 64 bits are shown). The attribute prediction can be viewed as mapping the binary codes from Hamming space to the attribute space, where U serves as the projection matrix. So the correlation coefficient between two columns of U naturally encodes the inherent correlation between the two corresponding attributes (please see supplementary material). As for the computational complexity, we fix the face representation (i.e., 4000-dimensional Gabor feature), the platform (Matlab v7.12 on a PC with Intel Core i7 processor of 3.40GHz), and the training size (i.e., 5,000 training instances) for both methods to ensure fair comparison. By

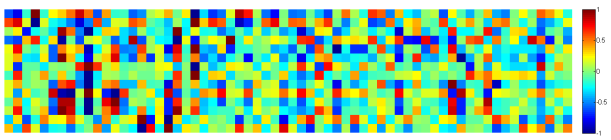

Figure 4: Visualization of the correlation between facial attributes (14 rows) and hash bits (64 columns), i.e., the proposed attributes predictor U in Eqn. (8). For better view, U^T is shown.

taking the average of ten times running, our method takes about 100s for training (256 bits), while the training process of the baseline method [16] takes about 54 minutes.

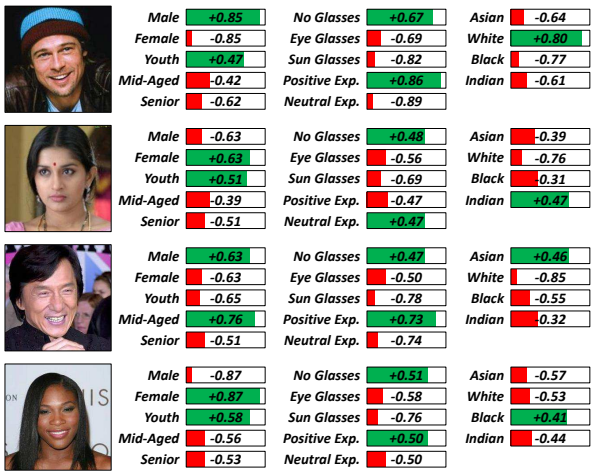

Figure 5: Four real cases of facial attributes prediction on CFW 60K with 256 bits binary codes. Each attribute bar ranges from the leftmost -1 to the rightmost $+1$, which respectively indicates the absence and presence of the current attribute.

4.5. Thinking: Complementary or Not?

Although the proposed method has achieved a certain level of success on both face image retrieval and facial attributes prediction, it is still not that clear till now whether the two tasks are complementary. Can they be performed separately, or is it really necessary to integrate these two into a whole learning framework? To answer such questions, we further conduct a comparison experiment, i.e., optimizing the two learning tasks separately and sequentially. More specifically, we first learn the discriminative binary codes only with the identity discriminability constraint (E_{dis}), and then use them for face image retrieval. After that, we utilize the code-attribute consistency constraint (E_{cons}) to compute the closed-form attribute predictor U, and finally use it for facial attributes prediction. Experimental results can be found in Table 3.

It is quite obvious to find that without the other task, single task learning is not as satisfactory as joint learning. We believe the reason behind this lies in the natural complementarity between the two tasks. Let's give a more detailed analysis: most attributes involved are identity-preserving, e.g., gender, race, and these attributes will definitely help promote the learning of code discriminability. Although there exist some attributes that are not identity-preserving, e.g., age and eye accessory, they do have high correlation with corresponding celebrities. As for the few expression attributes, since most of the annotations in our CFW 60K database are negative, it can secure to a large extent that they will hardly confuse the optimization. Besides, from the perspective of optimization theory, non-convex optimization for single task inevitably has the risk of falling into local optima, and iteratively optimizing with another complementary task has the potential to approach more stable global optima. Therefore, it is of great significance to encode extra information apart from identity into the human-incomprehensible binary codes, and we believe that by selecting appropriate information, e.g., attributes, complementarity can be fully explored to benefit both learning tasks.

Table 3: Evaluation of the complementarity of the two learning tasks, where the retrieval performance is measured by mAP, and attributes prediction is measured by average accuracy of the 14 attributes. 256 bits of binary code are used here.

Learning Strategy	Face Retrieval	Attributes Prediction
Separate Learning	0.1364	0.8245
Joint Learning	0.1979	0.8605

5. Conclusion

To address the problem of large-scale face image retrieval, we proposed a novel multipurpose binary code learning method by dexterously integrating three constraints into a unified framework, i.e., identity discriminability, code-attribute consistency, and code prediction stability. The learned codes can be directly used for face image retrieval, and in addition, they can also be readily used for facial attributes prediction by mapping the codes from Hamming space to the attribute space. Along with the method, we also built a large-scale web celebrity database with identity and facial attributes annotation, i.e., CFW 60K. Experimental results demonstrate the advantage of our method over state-of-the-art hashing methods and classic attributes prediction method. We hope this work would provide a train of thought for researchers that it is of great significance to encode auxiliary information into discriminant binary codes. For future work, we are exploring to apply the proposed framework to more generalized fine-grained image retrieval, not limited to faces.

Acknowledgements

This work is partially supported by 973 Program under contract No. 2015CB351802, Natural Science Foundation of China under contracts Nos. 61390511, 61222211, 61379083, and Youth Innovation Promotion Association CAS No. 2015085.

References

[1] B.-C. Chen, Y.-H. Kuo, Y.-Y. Chen, K.-Y. Chu, and W. Hsu. Semi-supervised face image retrieval using sparse coding with identity constraint. In *Multimedia*, pages 1369–1372. ACM, 2011. 1

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995. 4

[3] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253–262. ACM, 2004. 1, 2

[4] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, pages 1778–1785. IEEE, 2009. 2

[5] V. Ferrari and A. Zisserman. Learning visual attributes. 2007. 2

[6] Y. Gao and Y. Qi. Robust visual similarity retrieval in single model face databases. *PR*, 38(7):1009–1020, 2005. 1

[7] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, volume 99, pages 518–529, 1999. 1, 2, 6

[8] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, pages 817–824. IEEE, 2011. 1, 2, 6

[9] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272. IEEE, 2008. 3

[10] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, 2007. 6

[11] W. Kong and W.-J. Li. Isotropic hashing. In *NIPS*, pages 1646–1654, 2012. 1

[12] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. 3

[13] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137. IEEE, 2009. 1, 2

[14] G. Kulkarni, V. Premraj, S. Dhar, S. Li, Y. Choi, A. C. Berg, and T. L. Berg. Baby talk: Understanding and generating simple image descriptions. In *CVPR*, pages 1601–1608. IEEE, 2011. 2

[15] N. Kumar, P. Belhumeur, and S. Nayar. Facetracer: A search engine for large collections of images with faces. In *ECCV*, pages 340–353. Springer, 2008. 2, 7

[16] N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *ICCV*, pages 365–372. IEEE, 2009. 2, 7, 8

[17] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, pages 951–958. IEEE, 2009. 2

[18] H. Lee, Y. Chung, J. Kim, and D. Park. Face image retrieval using sparse representation classifier with gabor-lbp histogram. In *Information Security Applications*, pages 273–280. Springer, 2011. 1, 7

[19] C. Liu and H. Wechsler. Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *TIP*, 11(4):467–476, 2002. 6

[20] D. C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989. 5

[21] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081. IEEE, 2012. 1, 2, 5, 6

[22] P. Lyman and H. Varian. How much information 2003? 2004. 1

[23] R. Pan, Y. Zhou, B. Cao, N. N. Liu, R. Lukose, M. Scholz, and Q. Yang. One-class collaborative filtering. In *ICDM*, pages 502–511. IEEE, 2008. 3

[24] D. Parikh and K. Grauman. Relative attributes. In *ICCV*, pages 503–510. IEEE, 2011. 2

[25] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *NIPS*, pages 1509–1517, 2009. 1, 2

[26] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV*, pages 876–889. Springer, 2012. 1, 2, 4, 5, 6

[27] W. J. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *CVPR*, pages 2933–2940. IEEE, 2012. 2

[28] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Augmented attribute representations. In *ECCV*, pages 242–255. Springer, 2012. 2, 5

[29] P. Shih and C. Liu. Comparative assessment of content-based face image retrieval in different color spaces. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(07):873–893, 2005. 1, 7

[30] B. Siddiquie, R. S. Feris, and L. S. Davis. Image ranking and retrieval based on multi-attribute queries. In *CVPR*, pages 801–808. IEEE, 2011. 2

[31] P. Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001. 4

[32] A. Turpin and F. Scholer. User performance versus precision measures for simple search tasks. In *SIGIR*, pages 11–18. ACM, 2006. 6

[33] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*, pages 448–456. ACM, 2011. 3

[34] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431. IEEE, 2010. 1, 2, 6

[35] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang. Learning hash codes with listwise supervision. In *ICCV*, pages 3032–3039. IEEE, 2013. 5

[36] Q. Wang, L. Si, and D. Zhang. Learning to hash with partial tags: Exploring correlation between tags and hashing bits for large scale image retrieval. In *ECCV*, pages 378–392. Springer, 2014. 3

[37] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008. 1, 2, 6

[38] F. X. Yu, L. Cao, R. S. Feris, J. R. Smith, and S.-F. Chang. Designing category-level attributes for discriminative visual recognition. In *CVPR*, pages 771–778. IEEE, 2013. 2

[39] X. Zhang, L. Zhang, X.-J. Wang, and H.-Y. Shum. Finding celebrities in billions of web images. *IEEE Trans. on Multimedia*, 14(4):995–1007, 2012. 6