Neurocomputing 213 (2016) 5–13

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

A Bayesian Hashing approach and its application to face recognition

Qi Dai [a], Jianguo Li [b], Jun Wang [c], Yurong Chen [b], Yu-Gang Jiang [a,*]

[a] School of Computer Science, Fudan University, Shanghai, China
[b] Intel Labs China, Beijing, China
[c] School of Computer Science and Software Engineering, East China Normal University, Shanghai, China

ARTICLE INFO

Article history:
Received 24 September 2015
Received in revised form
11 April 2016
Accepted 2 May 2016
Available online 29 June 2016

Keywords:
Bayesian Hashing
Hamming embedding
Face recognition

ABSTRACT

With the rapid development in the computer vision community, many recent studies show that high-dimensional feature representations can produce better accuracies in various image and video content recognition tasks. However, it also brings high costs for both computation and storage. In this paper, we introduce a novel method called *Bayesian Hashing*, which learns an optimal Hamming embedding to encode high-dimensional features to binary bits, and discuss its application to the challenging problem of face recognition. The learned hashing representation is modeled with a well-designed supervised Bayesian learning framework, which consists of three ingredients. First, we elaborately model local bit correlations using Naive Bayesian model (FERN), and boost FERNs to obtain a classifier for the hashing bit stream. Second, without incurring additional storage cost, we impose hashing bit-stream permutations to obtain a series of classifiers, which could achieve better performance. Third, we introduce the sequential forward floating search (SFFS) algorithm to perform model selection on multiple-permutation models, gaining further performance improvement. We carry out extensive evaluations and comparative studies, which demonstrate that the proposed approach gives superior performance on both accuracy and speed. State-of-the-art results are achieved on several well-known face recognition benchmarks.

1. Introduction

Recently many researchers have demonstrated that, compared to low-dimensional features, high-dimensional feature representations often perform better, especially in the area of image and video content analysis. The challenging face recognition serves as one of the typical tasks with this phenomenon [1,2]. Many efforts are done to reduce the costs on both storage and computation, with the developments of different feature compression (selection, filtering, and projection) techniques. In this paper, we take face recognition as an example to study the feature compression and associated classification problem.

Face recognition is a topic of active research in recent years, mostly due to its wide applications [3,4]. In particular, researchers have created several real-world benchmarks with challenging *uncontrolled* face images recently, including Labeled Faces in the Wild (LFW) and YouTube Faces (YTF), which have been popular in this area.

Recent face recognition techniques could be roughly categorized into two groups, i.e., low-level representation designs and learning-based methods. In particular, the former methods aim at designing robust hand-crafted feature representations. Hence, various feature representation approaches are proposed, such as Gabor [5,6], LBP [7], and HoG [8]. The latter methods utilize modern machine learning techniques to provide better feature representations or recognition strategies. For instance, with machine learning techniques, some methods could learn more discriminative representations from the low-level features or raw images, e.g. subspace based methods [9–12] and mid-level representation (a.k.a. attributes) learning [13,14]. Other methods try to build advanced learning models with higher discriminative power. The developed learning models can be distance metrics [15–18], classifiers [19] and Boosting [20]. In addition, some recent works rely on the popular deep learning techniques, which simultaneously perform feature learning and model training in an end-to-end learning framework [21–24].

To achieve better recognition accuracies, many existing approaches tend to employ high-dimensional feature representations or over-complete feature sets [25,26,2,1]. However, with such high-dimensional features, we need more storage space and computational time, which make the practical application infeasible. To solve this problem, one of the recent works suggests to learn low-rank representations from the high-dimensional features [27]. Another solution is to generate good feature representations by using feature selection or sparse representation

* Corresponding author.
E-mail addresses: daiqi@fudan.edu.cn (Q. Dai), jianguo.li@intel.com (J. Li), jwang@sei.ecnu.edu.cn (J. Wang), yurong.chen@intel.com (Y. Chen), ygj@fudan.edu.cn (Y.-G. Jiang).

http://dx.doi.org/10.1016/j.neucom.2016.05.097
0925-2312/

[28,29]. In this paper, different from performing the low-rank matrix recovery or feature selection, we use hashing technology to alleviate this issue. In particular, a novel Bayesian framework, which encodes the high-dimensional face features into compact binary codes, is proposed, yielding significantly reducing of computation and storage costs. The learned hashing representation is modeled with a well-designed supervised Bayesian learning framework, which consists of three steps. First, we elaborately model local bit correlations using Naive Bayesian model (FERN), and boost FERNs to obtain a classifier for the hashing bit streams. Second, to further exploit bit-level relationships among different binary codes, we impose hashing bit-stream permutations to obtain a series of classifiers with different permutations for achieving better discrimination power. Note this step does not incur additional storage cost. Third, we introduce the sequential forward floating search (SFFS) algorithm to perform model selection on multiple-permutation models, resulting in further performance improvement. Unlike most existing face recognition methods that use floating point feature representations, we use compact binary codes extracted by the proposed Bayesian Hashing framework. What is more, our method can still maintain state-of-the-art recognition performance.

The main contribution of this work is the Bayesian optimal Hamming embedding method, namely Bayesian Hashing, which can efficiently encode floating point features into compact binary codes. Also, after building the boosted FERNs classification models on hashing bit-streams and exploiting bit-level relationships with random permutation technique, we further show that performance improvement could be obtained by sequential model selection. Finally, extensive experiments are carried out on three popular face benchmark datasets, i.e., the FRGC, the LFW and the YTF. The results demonstrate that the proposed method produces superior performance on both accuracy and speed. This paper extends upon a recent conference version [30] with extended discussions and new evaluations on a popular dataset and more parameter settings.

In the following, we discuss related works in Section 2 and introduce the proposed Bayesian Hashing in Section 3. In Section 4, we discuss experimental settings and results. Finally, Section 5 concludes this paper.

2. Related works

Related works are discussed in two groups: (1) feature projection and metric learning, and (2) supervised hashing.

2.1. Feature projection and metric learning

Numerous studies have focused on the high-dimensional feature learning issue. Among these works, subspace methods are probably the most common choices to reduce redundancy of high-dimensional features. Suppose we are going to project a d-dimensional raw feature into a p-dimensional discriminant subspace, with the projection matrix to be of size $d \times p$. Generally, the original feature dimension d will be very high. For example, some traditional face features such as Gabor feature [6,31] are with tens of thousands of dimensions; in addition, the value of d could be more than several hundreds of thousands in some over-complete feature learning algorithms like [26,1,2]. It would be a huge cost to learn such a projection matrix, which makes the practical application infeasible. For example, Ref. [32] proposes a supervised regularization locality-preserving projection approach based on a supervised graph and a new regularization strategy, which is computationally expensive. In order to solve the scalability problem, various techniques are proposed, including the divide-and-conquer strategy [25], sparse projection matrix learning with ℓ_1 regularization [2], and low-rank representations learning [27]. These techniques could reduce the computation or storage cost to a certain degree.

Metric learning is also widely used in face recognition. It aims to learn a distance metric which maximally separates different subject classes. The most representative is the Mahalanobis distance metric, which is defined in a quadric form $d(\mathbf{v}_i, \mathbf{v}_j) = (\mathbf{v}_i - \mathbf{v}_j)^T \mathbf{M} (\mathbf{v}_i - \mathbf{v}_j)$, where $\mathbf{v}_i$ and $\mathbf{v}_j$ are features of two face images. The symmetric positive definite matrix $\mathbf{M} \in R^{d \times d}$ can be decomposed to $\mathbf{A}^T\mathbf{A}$, where $\mathbf{A}$ is a linear transformation matrix. The learned distance function $d(\mathbf{v}_i, \mathbf{v}_j)$ usually measure the actual distance in transformed space, and can be incorporated into objectives like the logistic discriminant function [15], and the Cosine function [17]. In [26], sparse block diagonal constraints are imposed on the metric matrix $\mathbf{M}$ for fast training. In [33], the authors proposed a binary representation with metric learning to reduce the dimensionality of high-dimensional Fisher vector (FV) features. In [34], the side information and the inherent neighborhood structures among examples with a set of similar pairwise constraints are utilized to learn a kernel coupled distance metric. In addition, some other distance metrics are also explored in recent works. Reference [35] uses Cayley–Klein distance metric, which is defined in terms of logarithm and projective cross-ratio function, as an alternative to the Mahalanobis metric. Reference [36] utilizes projection metric to map data from the original Grassmann manifold to another more discriminative one. Reference [37] performs Log-Euclidean metric learning on symmetric positive definite matrices manifold, and learns a tangent map to transform the matrix logarithms to a new tangent space. Despite the good results shown in these approaches, the time cost of metric learning is very huge for high-dimensional data. We propose to encode the high-dimensional features using a fast Bayesian Hashing method, which significantly reduces the computation and storage costs.

2.2. Supervised hashing

By mapping raw features to compact binary codes (in Hamming space) with the preservation of the similarity, hashing techniques could significantly reduce the computation and storage costs. Different from unsupervised hashing, which keeps the feature similarity, supervised hashing aims to preserve semantic similarity. More and more researchers have focused on this area. For example, LDA-Hash [38] maximizes the difference between label-same/label-different pairs in the linear discriminative projection space. Supervised Hash with Kernels (KSH) [39] applies kernelization to formulate hash functions and minimizes the loss function over hash codes. However, the kernel computation is very time consuming in both training and testing. CNN Hash [40] learns the hash functions in an end-to-end manner. Semantic correlation maximization Hash (SCM) [41] handles large scale data by using supervised multimodal hashing. FastHash [42] uses decision tree based hash functions and GraphCut based binary code inference, showing its applicability on large scale application. Supervised discrete hashing [43] formulates the hash coding problem with linear classifier training jointly, and decomposes the problem into three sub-problems. Discrete cyclic coordinate descent algorithm is proposed to solve the problem. Asymmetric inner product binary coding [44] learns two asymmetric coding functions such that the inner products between original data pairs are approximated by the produced codes. Dual complementary hashing [45] learns multiple complementary hash tables, where all hash functions inside each hash table are also complementary. Multi-component hashing [46] tries to find latent components which imply the similarity between different objects. Probabilistic attributed hashing [47] integrates attributes with low-level features, by building connections between them on latent binary codes. Different from most of the existing supervised hashing techniques which attempt

to preserve semantic similarity in the final Hamming space, our Bayesian Hashing directly aims at minimizing the Bayes error of the classification problem to generate compact hash codes.

Moreover, Ref. [48] proposes a method called BayesLSH. This work has a similar name to our approach. However, it is fundamentally different from ours. It only uses the Bayes theorem to estimate the similarity between two existing hash codes.

3. Bayesian hashing for face recognition

Since many practical face recognition systems use high-dimensional representations, they require significant computational and storage overhead. Our objective is to learn a compact representation from the original features with negligible loss of recognition performance. Being popular in large scale search and retrieval applications, hashing based binary embedding, namely Hamming embedding, has drawn intensive intentions in the past decade. In this paper, we present a Bayesian Hashing method to encode high-dimensional face features into compact and effective binary codes. Briefly, we formulate our objective by minimizing the Bayes error to obtain an optimal Hamming embedding. We further utilize the boosted FERNs to perform the task of face recognition with learned hash codes.

The proposed Bayesian hashing based face recognition system consists of three key components. First, an optimal Hamming embedding is obtained via minimizing the Bayes error and we encode face patches into binary hash codes, as discussed in Section 3.2. Second, given learned Bayesian hash codes of face images, we train boosted FERNs classifiers to performance recognition tasks. Details are reported in Section 3.3. Finally, to employ the relationships between different patches, a bit-stream permutation technique is adopted to learn multiple random FERNs models. Then the sequential forward floating search is applied to identify good permutation models to produce final results, as discussed in Section 3.4.

Below we first briefly introduce the notations, followed by the details of each key component.

3.1. Notations and settings

Note that we first transform the raw face images to Gradient location-orientation histogram (GLOH), as suggested in [8]. In particular, for each face image, n patches of different scales are extracted around landmarks. Furthermore, we make a mirror of each image, obtaining another n more patches. Therefore, we get a total of $K = 2n$ patches, each represented by 17 blocks, where every block is with a 8 dimensional histogram-style feature. Finally, each patch is represented by a $d_0 = 136$ dimensional feature vector.

In this paper, pair-wise (not limited to) representation for face recognition is used as suggested in [49]. Given a pair of face images, we obtain the corresponding feature vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j \in \mathbb{R}^d$ ($d = K^* d_0$). We define $\boldsymbol{x}_{ij}$ to be the pair representation which is element-wise absolute-difference $\boldsymbol{x}_{ij} = (\| \boldsymbol{v}_{i1} - \boldsymbol{v}_{j1} \|^p, \ldots, \| \boldsymbol{v}_{id} - \boldsymbol{v}_{jd} \|^p)$. Then, $\boldsymbol{x}_{ij}$ is assigned with a label $y = 1$ if $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are from the same subject, and $y = -1$ otherwise. The positive training set $\mathcal{P}$ consists of the same-subject pairs, and the negative training set $\mathcal{N}$ consists of the other pairs. Note that our method is not limited to univariate ($\boldsymbol{x}_{ij}$) form. We may easily extend our framework to bi-variables (pair inputs) form $\mathcal{G}(\boldsymbol{v}_i, \boldsymbol{v}_j)$, and obtain the discriminant function as in [16] or [50]. Nevertheless, we are not going to discuss the bi-variable case as it is beyond the scope of this work.

3.2. Bayesian hashing: formulation

We begin with the assumption that both $\mathcal{P}$ and $\mathcal{N}$ follow normal distribution, i.e., $P(\boldsymbol{x}|y = 1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_p, \boldsymbol{\Sigma}_p)$ and $P(\boldsymbol{x}|y = -1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$. Our goal is to predict the label y of the given pair representation $\boldsymbol{x}$. Thus the *log-ratio discriminant function* is:

$$\mathcal{G}(\boldsymbol{x}) = \ln \frac{P(\boldsymbol{x}, y = 1)}{P(\boldsymbol{x}, y = -1)} = -(\boldsymbol{x} - \boldsymbol{\mu}_p)^T \boldsymbol{\Sigma}_p^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_p) + (\boldsymbol{x} - \boldsymbol{\mu}_n)^T \boldsymbol{\Sigma}_n^{-1} (\boldsymbol{x} - \boldsymbol{\mu}_n) + \mathbf{1} \cdot \boldsymbol{b}, \quad (1)$$

where $\boldsymbol{b}$ is the bias parameter of the prior.

Similar to many hashing works, we need to seek a set of hash functions $\hat{y} = h(\boldsymbol{x}; \boldsymbol{b})$. While in Bayesian Hashing, we aim to find hash functions which *minimize the following Bayes error* on the training set,

$$\mathcal{L}(\boldsymbol{b}) = \int_{\boldsymbol{x} \in \mathcal{P}} P(\hat{y} \neq 1|\boldsymbol{x}) d\boldsymbol{x} + \int_{\boldsymbol{x} \in \mathcal{N}} P(\hat{y} \neq -1|\boldsymbol{x}) d\boldsymbol{x}. \quad (2)$$

Next we will first consider the simplest case with only 1-dimensional feature. An extension on multi-dimensional setting will then be discussed.

3.2.1. 1-dimensional case

The equations $P(x|y = 1) = \mathcal{N}(x|\mu_p, \sigma_p)$ and $P(x|y = -1) = \mathcal{N}(x|\mu_n, \sigma_n)$ hold in this case. Then we rewrite Eq. (1) to

$$\mathcal{G}(x) = -(x - \mu_p)^2/\sigma_p^2 + (x - \mu_n)^2/\sigma_n^2 + b. \quad (3)$$

The hash function in this case could be obtained by assuming $\sigma_p = \sigma_n = \sigma_a$:

$$h(x; b) = \text{sign}\{(x - \mu_p)^2 - (x - \mu_n)^2 + b\} \quad (4)$$

To solve b, a line search algorithm (such as golden section search) is applied, with the target of minimizing the cost in Eq. (2). Fig. 1 illustrates how the hash function $h(x; b)$ works in the 1-dimensional case.

3.2.2. d-dimensional case

Similar to many Hamming embedding algorithms [38], in d-dimensional case ($d > 1$), we adopt a standard two-stage procedure.

First, a subspace projection is employed to decorrelate features. To achieve this objective, many different well-known criteria could be used, such as Bhattacharyya criterion [51]. Here we adopt the *Fisher criterion*, which aims to *maximize the separation* between $\mathcal{P}$ and $\mathcal{N}$ using the ratio of variances on $\mathcal{P}$ and $\mathcal{N}$ as

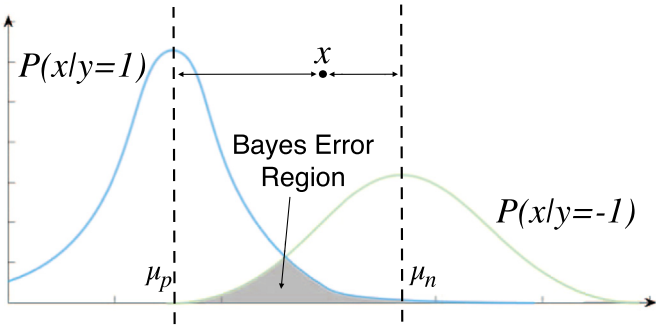


Fig. 1. An illustration of hash function $h(x)$ in the 1-dimensional case. We solve the problem by minimizing the Bayes error, where the sample x will be assigned to class $y = -1$ in this example.

$$S = \frac{(\boldsymbol{w}\cdot(\boldsymbol{\mu}_p - \boldsymbol{\mu}_n))^2}{\boldsymbol{w}^T(\boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_n)\boldsymbol{w}}. \tag{5}$$

We can obtain the projection $\boldsymbol{\Sigma}^{-1}$ with Lagrange multipliers, where $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_n$. It can be observed that $\boldsymbol{\Sigma}$ is a symmetric positive semi-definite matrix, thus the SVD decomposition $\boldsymbol{\Sigma}^{-1} = (U\,S\,U^T)^{-1} = (U^T S^{-\frac{1}{2}})(S^{-\frac{1}{2}}U) = A^T A$ exists, where $A = S^{-\frac{1}{2}}\,U$. By replacing $\boldsymbol{\Sigma}_p$ and $\boldsymbol{\Sigma}_n$ with A^{TA}, we simplify Eq. (1) to

$$\mathcal{G}(\boldsymbol{x}) = -(\boldsymbol{x}' - \boldsymbol{\mu}_p')^T(\boldsymbol{x}' - \boldsymbol{\mu}_p') + (\boldsymbol{x}' - \boldsymbol{\mu}_n')^T(\boldsymbol{x}' - \boldsymbol{\mu}_n') + \boldsymbol{1}\cdot\boldsymbol{b}. \tag{6}$$

Note that $\boldsymbol{x}' = A\boldsymbol{x}$ and $\boldsymbol{\mu}' = A\boldsymbol{\mu}$, indicating that $\boldsymbol{x}'$ and $\boldsymbol{\mu}'$ are both in the projection space. Second, a hash function $h_i(x_i'; b_i)$ similar to Eq. (4) is derived for each element of $\boldsymbol{x}'$.

The objective of our method is to minimize the Bayes error in Eq. (2), which is different from the two-stage procedures discussed in [38]. It is then used for training supervised hash functions in Eq. (4). To the best of our knowledge, we are the first to utilize Bayes error in the cost function of learning a Hamming embedding. Notice that the full feature dimension d of the images is very high. To reduce the cost, we perform subspace projection on each *face patch* with a relatively lower feature dimension, rather than on the whole d dimensional feature space.

3.3. Boosted FERNs

Now Bayesian hashing could encode the high-dimensional feature input $\boldsymbol{x}$ to a sequence of hash bits $\boldsymbol{f} = (f_1, \ldots, f_D)$, where $f_i \in \{0, 1\}$. To further utilize them for recognition, we propose to model the bit stream with a random boosted FERNs framework [52]. Fig. 2 shows our configuration of the boosted random FERNs.

In a general FERN framework, when given a bit-sequence $\boldsymbol{f}$, we partition it into M groups of size $S = \frac{D}{M}$, obtaining a set of feature vectors $\boldsymbol{F} = (F_1, \ldots, F_M)$. Each group F_i is called a FERN, and is modeled with a Naive Bayesian classifier $P(F_i|y)$. By assuming that all groups are independent, we could compute the joint probability for all FERNs $P(\boldsymbol{F}|y) = \prod_{i=1}^{M} P(F_i|y)$. In our implementation, we set S to 8 in most occasions and separately study the effect of S in the experiments. In addition, the Boosted framework is adopted to reduce possible redundancy among different groups. Each FERN is modeled as a weak classifier. Then a GentleBoost, a variant of AdaBoost classifiers, is trained to ensemble different FERNs. Table 1 summarizes the training algorithm of the boosted FERNs.

When performing boost training, we restrict that each FERN group can only be chosen once to reduce redundancy. Finally, a decision function can be obtained:

$$\mathcal{H}(\boldsymbol{F}) = \sum_{i=1}^{T} \{P(F_i, y = 1) - P(F_i, y = -1)\}, \tag{7}$$

where $T \le M$ is the number of selected groups. The calculation of the decision function is very fast as it is actually the accumulation of a set of look-up-tables (LUTs), which makes the prediction extremely efficient. In addition, we can further compress the model by quantizing the float-point LUTs into 8-bit chars.

3.4. Bit permutation and model selection

It has been discussed in Section 3.2.2 that if we perform Hamming embedding on the whole feature space with d to be very large, the computation and storage costs of the projection procedure will be huge. Thus we adopt patch-level Bayesian Hashing to overcome this problem. While such operation also brings another issue, i.e. how to exploit relationships among different patches. To this end, we employ a bit-stream permutation technique to explore possible correlations. Fig. 2 shows the overall framework. We produce a total of G permutations. With permutation, each group contains bits which come from different patches. We define the hash code of the g-th random permutation as $\boldsymbol{f}_g = \{f_{\delta(g,1)}, \ldots, f_{\delta(g,D)}\}$, where $\delta(,)$ denotes the indices after permutation.

We train a boosted FERNs model $\mathcal{H}(\boldsymbol{F}_{\delta g})$ for each permutation δ_g, forming a set of G models in total. Naturally, we could get better

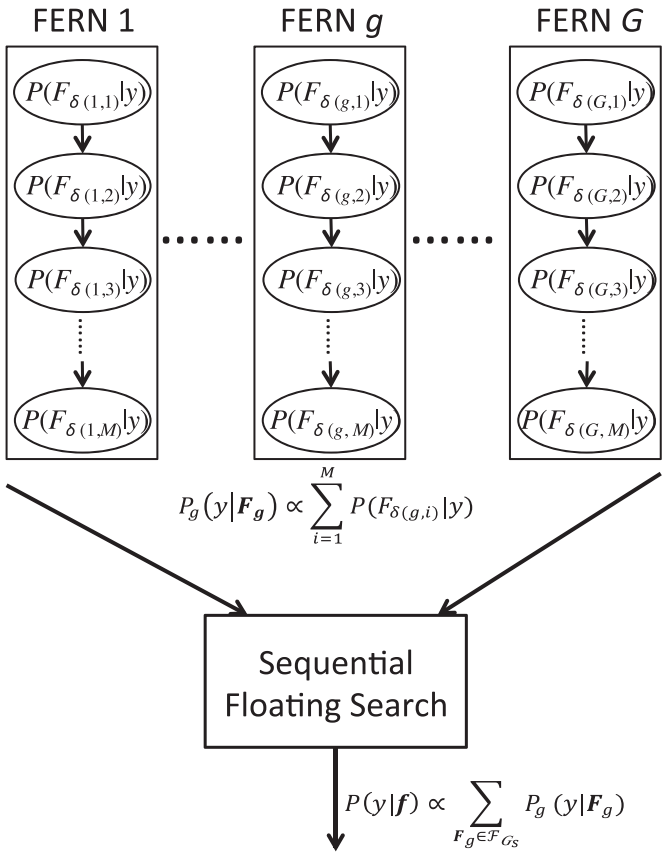


Fig. 2. Framework of the boosted FERNs classification model. Each column indicates one bit-stream permutation, which represents one boosted FERNs classifier. Each circle is one FERN group. In this example, each classifier has M groups of bits.

Table 1
The Boosted FERNs training algorithm using GentleBoost.

Input: Training set: $\{(F_i^k, y_i)\}_{i=1}^N$, k is from 1 to M, with N being the number of samples and M being the number of FERN bytes. F^k is the k-th FERN group.
Initialize: Maximum iteration number T and the current iteration $t=0$. Initialized weights for both positive and negative samples.
(1) when $y_i=1$, $w_{1,i}^{+} = \frac{1}{N_+}$;
(2) when $y_i=-1$, $w_{1,i}^{-} = \frac{1}{N_-}$.
Step 1: For each group F^k, build Naive Bayesian probability look-up table $P(F^k|y)$;
Step 2: Select the best group $F^{\sigma(t)}$ according to the Bayes error on the training set, and define the decision function as
$h_t(F^{\sigma(t)}) = P(F^{\sigma(t)}, y = 1) - P(F^{\sigma(t)}, y = -1)$.
Step 3: Weights update by $w_{t,i} = w_{t-1,i}\cdot\exp[-y_i h_t(F_i^{\sigma(t)})]$; then normalize the weights so as to keep $\sum_i w_{t,i}^{+} = 1$ and $\sum_i w_{t,i}^{-} = 1$;
Step 4: $t=t+1$. If $t=T$, break; otherwise go to Step 1.
Output: The final classifier $\mathcal{H}_T(\boldsymbol{F}) = \sum_{t=1}^{T} h_t(F^{\sigma(t)})$.

Table 2
SFFS algorithm used for permutation selection.

Input: Feature grouping set after random permutation $\mathcal{F} = \{\boldsymbol{F}_g, 1 \le g \le G\}$. $J(\mathcal{F}_k)$ represents the accuracy with the permutation set $\mathcal{F}_k$.
Initialize: Set $\mathcal{F}_0 = \varnothing$ and $k=0$. Pre-set the required permutation number G_S.
Step 1: Inclusion
- Select the best permutation $\boldsymbol{F}^+ = \arg\max_{\boldsymbol{F}\in\mathcal{F}\setminus\mathcal{F}_k} J(\mathcal{F}_k \cup \boldsymbol{F})$, where $\mathcal{F}\setminus\mathcal{F}_k$ is the relative complement of $\mathcal{F}_k$ in $\mathcal{F}$;
- $\mathcal{F}_{k+1} = \mathcal{F}_k \cup \boldsymbol{F}^+$; $k = k + 1$;

Step 2: Conditional exclusion
- Select the worst permutation $\boldsymbol{F}^- = \arg\max_{\boldsymbol{F}\in\mathcal{F}_k} J(\mathcal{F}_k - \boldsymbol{F})$;
- If $J(\mathcal{F}_k - \boldsymbol{F}^-) > J(\mathcal{F}_{k-1})$, set $\mathcal{F}_{k-1} = \mathcal{F}_k - \boldsymbol{F}^-$, $k = k - 1$, and go to Step 2; otherwise go to Step 3.

Step 3: If $k \ge G_S$, break; else go to Step 1.
Output: Selected subset of feature grouping set $\mathcal{F}_k$.

discrimination power with more permutations. However, it also leads to possible redundancy, and more computation and storage costs are required. To address this issue, we choose an optimal set of permutation models using Sequential Forward Floating Search (SFFS), achieving improved performance without any additional cost. Table 2 gives the details of the SFFS algorithm.

4. Experiments

4.1. Datasets and experimental settings

4.1.1. FRGC

The Face Recognition Grand Challenge (FRGC) version 2 [53] is designed to provide a comprehensive benchmark for evaluating new face recognition technologies. We focus on experiment-4 (in simple, FRGC-204), which has 12,776 training faces, 16,028 target face images and 8014 query faces. The target set and query set are taken under different environments, i.e. the former is under controlled environment and the latter is under uncontrolled environment. Such setting makes the FRGC-204 to be very challenging. Four landmarks (eye centers, nose tip and mouth center) are provided for each face. And face images are normalized to 128×128 according to these landmarks. We extract $n=240$ patches of different scales and ratios according to the landmark positions. And a mirror image is generated for each normalized face, leading to another $n=240$ patches. In the end, each face is represented using 480 GLOH patches, with each patch described by a 136-dimensional feature.

We report the true positive rate at a fixed false positive rate of 0.1% (TPR@FPR=0.1%), following the traditions on this dataset. In addition, Receiver Operating Characteristic (ROC) curves are adopted for some of the evaluated approaches.

4.1.2. LFW

The popular Labeled-Faces-in-the-Wild (LFW) dataset contains 13,233 images of 5749 individuals, collected from the Internet. We use LFW-a, which is the aligned version of LFW, and employ similar settings for data preprocessing following the previous works on this dataset.

There are several benchmark evaluation protocols for LFW. Here the *unrestricted with label-free outside data* protocol is adopted. A 10-fold cross-validation experiment is conducted. In each trial, we use nine subsets for training and one for testing. We report the mean Equal Error Rate (EER) and also plot the ROC curves.

4.1.3. YTF

The YouTube Faces (YTF) dataset [54] consists of 3425 YouTube videos of 1595 individuals. An average of 2.15 videos are available for each individual, and the average length of a video is 181.3 frames. 5000 video pairs are equally divided into 10 folds to form a cross-validation evaluation setting. We follow the standard *restricted* protocol defined in [54], i.e. one cannot access to the subject identity labels. As the frames have been aligned, we simply average the feature vectors of the frames within one video clip to form a video feature vector. Recognition accuracy is reported following prior works.

4.2. Results and discussions

In order to analyze the impacts of different components and parameters, we organize the experiments into several subsections.

4.2.1. Bayesian Hashing

In this section we compare our Bayesian Hashing with two popular alternative hashing methods, Product Quantization (PQ) [55] and Supervised Hashing with Kernels (KSH) [39]. For PQ, we use the same number of bit groups as Bayesian Hashing (8160). While for KSH, due to the expensive training cost, we only manage to train 4800 hash functions (600 bytes). Bit-permutation is not adopted for all the three methods, and the bit group size S is fixed to 8. Table 3 summarizes the results on FRGC and Fig. 3(j) shows the corresponding ROC curves.

As shown in the table, Bayesian Hashing shows superior performance on all the evaluation terms. In particular, our Bayesian Hashing outperforms KSH significantly. Even if under the same settings (only 500 bytes are used), our method still shows much better result (68.40% vs. 59.20%). Compared to PQ, the accuracy is also a little better. More significantly, it indicates excellent computational efficiency. The training/testing time of our method is over $5\times/10\times$ faster than PQ, and over $100\times/3\times$ faster than KSH. Finally, our memory cost is also the lowest one.

4.2.2. Boosted FERNs

The use of boosted FERNs classifier is one of the major components of our system. We conduct an experiment on FRGC to evaluate the capability of the boosted FERNs, by comparing its results with SVM. Here we also set $S=8$.

Table 4 shows the results (the bottom five rows). Obviously, it could be observed that the boosted FERNs classification plays an important role in our method. Replacing it with SVM, we obtain an approximately 14% lower result using the same set of binary codes, indicating the effectiveness of boosted FERNs. Besides, we also train an SVM classifier on the raw floating point GLOH features. It produces a similar accuracy, which is 0.5% higher than our method. However, the proposed method requires much lower computation and memory costs.

4.2.3. Impact of bit group size

The size S of the bit group (cf. Section 3.3) is an important parameter. Using either a too small or a too large S, the Naive Bayesian classifier may not be able to accurately model the probability distribution. In addition, we need to consider the time cost

Table 3
Accuracy and computation/storage comparison with Product Quantization and Supervised Hashing with Kernels on FRGC. All the 3 methods use Boosted FERNs as the classifier. Our method is consistently better.

Methods	TPR@ FPR=0.1 (%)	Training time (Seconds)	Testing time (Seconds)	Memory (MB)
PQ [55]	89.50	20,915.8	3624.2	57.4
KSH [39]	59.20	433,569.6	683.6	73.2
Bayesian Hashing	90.35	3873.3	213.9	34.2

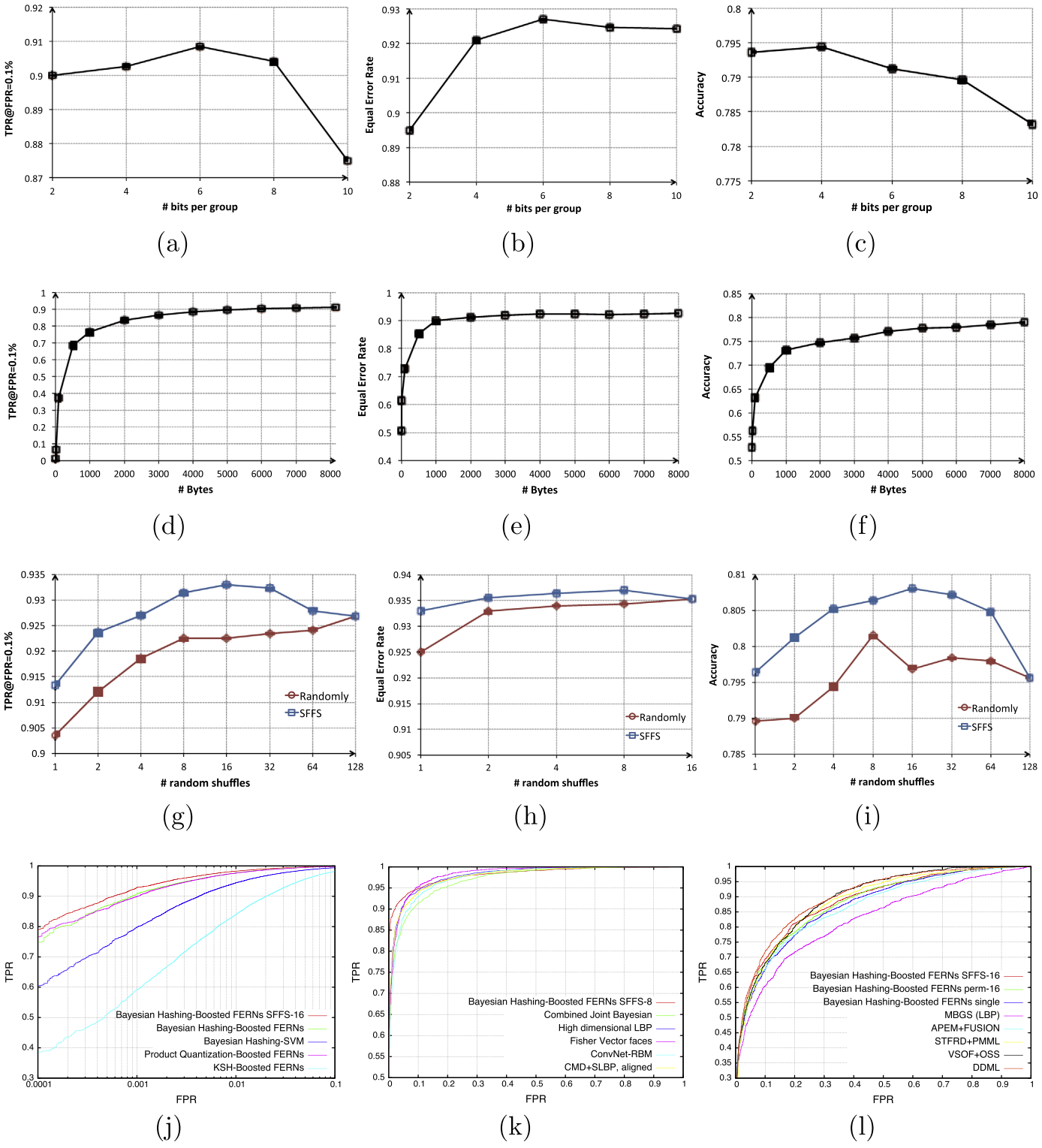


Fig. 3. *First column*: results on FRGC, *second column*: results on LFW, *third column*: results on YTF. *First row*: performance with different sizes of bit groups (w/o bit-permutation), *second row*: performance with different numbers of bytes (w/o bit-permutation), *third row*: performance with different numbers of bit permutations, *fourth row*: ROC curves. See texts for more discussions. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

—a small S will produce more groups, and thus more LUT query time will be needed.

Fig. 3 (a), (b) and (c) shows the performance vs. the number of bits per group on FRGC, LFW and YTF datasets, respectively. Bit-permutation is not adopted here. For FRGC and YTF, the performance drops significantly when using more than 8 bits (2–3%). Also, the performance is low when $S=2$ for LFW (3% drop). When $S=4$, 6 or 8, the performance is relatively stable and consistently good across all the datasets. In addition, it would cost double LUT query time when $S=4$, compared with $S=8$. Thus, $S=8$ is an appropriate choice. In all the other experiments, we fix $S=8$.

4.2.4. *Impact of the number of FERN bytes*

Our model assembles different bit groups using Boosted

Table 4
Results of our method and SVM on FRGC (the bottom five rows), in comparison with state-of-the-art results (the top five rows).

Methods	TPR@ FPR=0.1% (%)
Baseline, eigenface [53]	12
Gabor+Kernel [6]	76
LTP+Gabor+Kernel [31]	88.5
Gabor+Fourier [25]	89
LPQ-fusion [56]	91.59
GLOH floating+SVM	93.72
Bayesian Hashing+SVM	79.78
Bayesian Hashing+Boosted FERNs single	90.35
Bayesian Hashing+Boosted FERNs perm-128	92.68
Bayesian Hashing+Boosted FERNs SFFS-16	93.20

framework. As discussed above, we fix the size of bit group S to 8, indicating that each group could be represented by a byte, and modeled as a FERN. We then study how the number of the selected FERN bytes could affect the performance. In this experiment, with boosting training, the model selects top-N FERN bytes sequentially according to their contributions. Bit-permutation is also not adopted here and will be evaluated in the next experiment. Each GLOH block (8-dimensional; each patch has 17 blocks) is encoded into one byte. Fig. 3(d), (e) and (f) plots the performance vs. different numbers of bytes on FRGC, LFW and YTF, respectively.

Some interesting observations could be obtained from the figures. First, the performance rises rapidly when the number of bytes is very small, especially when $N < 1000$, on all the datasets. Steep curves could be observed in the figures. Second, on FRGC and YTF, a small portion of improvements are gained when $1000 < N < 4000$. While on LFW, very little improvement is gained. Finally, when $N > 4000$, the accuracies tend to be stable. Very little performance gains are obtained on FRGC and YTF. These observations indicate that we could achieve higher feature compression rate than the original Bayesian Hashing, which already yields $32\times$ compression rate of the high-dimensional floating point features. For example, if 4000 bytes are used, the feature compression ratio is more than $64\times$ with only 2% accuracy drop. And if 1000 bytes are used, we can also get a competitive result with over $256\times$ feature compression ratio.

This experiment also reveals a very promising property of our method. We can apply Hamming distance for fast coarse-level search (modern CPU supports hardware instructions for computing Hamming distance) when the database is very large. We could even make a progressively coarse-to-fine search strategy for large-scale applications. For instance, it is feasible to build a first level coarse search with just the top 1000 bytes, which can quickly narrow down the search space for fine-grained computations with more bytes.

4.2.5. Impact of bit permutation

This experiment studies the impact on bit-stream permutation. We train a set of boosted FERNs models based on different random permutations. Then different numbers of models are assembled to carry out the task. Fig. 3(g), (h) and (i) shows the results on FRGC, LFW and YTF, respectively (red curves). On FRGC and LFW, it is clear that the accuracy grows along with the increase of shuffles, especially at the beginning parts of the curves (# shuffles $\leq$8). While on YTF, the performance drops a bit after 8 shuffles. As a large number of shuffles may incur redundancy and also due to the randomization factor, the performance may not be always stable when using more shuffles.

Next we study the effectiveness of SFFS on all the datasets, which is expected to solve the redundancy problem of the random shuffles. As described above, a small subset of permutation models

Table 5
Performance (EER $\pm$ standard deviation) comparison with state-of-the-art approaches on LFW.

Methods	EER (%)
Combined Joint Bayes [50]	90.90 ± 1.48
CMD+SLBP [26]	92.58 ± 1.36
VMRS [1]	92.05 ± 0.45
ConvNet-RBM [22]	91.75 ± 0.48
Fisher Vector Faces [57]	93.03 ± 1.05
High-dim LBP [2]	93.18 ± 1.07
HPEN [58]	95.25 ± 0.36
Bayesian Hashing+Boosted FERNs single	92.50 ± 0.39
Bayesian Hashing+Boosted FERNs perm-16	93.53 ± 0.79
Bayesian Hashing+Boosted FERNs SFFS-8	93.70 ± 0.87

are selected from many models by evaluating on a separate validation set with SFFS. Good and complementary models are selected. For FRGC and YTF, we train a total of 128 models, and for LFW, we only train 16 models because the training time for LFW is much longer. As shown in Fig. 3(g), (h) and (i), SFFS is able to further improve the results with only a small number of selected permutations on all the datasets. Note that when large numbers of permutations are selected (>16 for FRGC and YTF, and >8 for LFW) by SFFS, the accuracies drop. It is not surprising because large number of permutations will lead to redundancy as discussed above, with the decrease on performance.

4.3. Comparison with state of the arts

Finally, we compare our method with several state-of-the-art works on all the datasets. Results on FRGC, LFW and YTF are summarized in Tables 4–6 respectively, where "perm-x" indicates x random permutations and "SFFS-x" indicates x permutations selected using SFFS. On all datasets, we achieve very competitive results. One surprising result is that our approach outperforms the deep learning based method on LFW [22]. It is quite appealing as only traditional hand-crafted GLOH features are adopted in our method. In addition, our method can be employed on any type of features, including the popular deep learning based ones. We expect that a further gain may be obtained by replacing GLOH with the recently developed deeply learning features. We further present the ROC curves of our method and other compared approaches on LFW and YTF in Fig. 3(k) and 3(l) respectively. While for FRGC, we only show the ROC curves of our method in Fig. 3(j), as we do not have the data needed for plotting curves of the compared works. Furthermore, Fig. 4 shows some successful or failed examples on YTF. Finally, we would like to stress again that our results are obtained with binary codes, which own the nice property of very lower computation and storage costs. In contrast, all the compared works rely on the expensive floating features.

Table 6
Performance (Accuracy $\pm$ standard deviation) comparison with state-of-the-art approaches on YTF.

Methods	Accuracy (%)
MBGS (LBP) [54]	76.40 ± 1.80
APEM+Fusion [59]	79.06 ± 1.51
STFRD+PMML [60]	79.48 ± 2.52
VSOF+OSS [61]	79.70 ± 1.80
DDML [62]	82.34 ± 1.47
Bayesian Hashing+Boosted FERNs single	78.96 ± 0.53
Bayesian Hashing+Boosted FERNs perm-128	79.56 ± 0.54
Bayesian Hashing+Boosted FERNs SFFS-16	80.80 ± 0.55

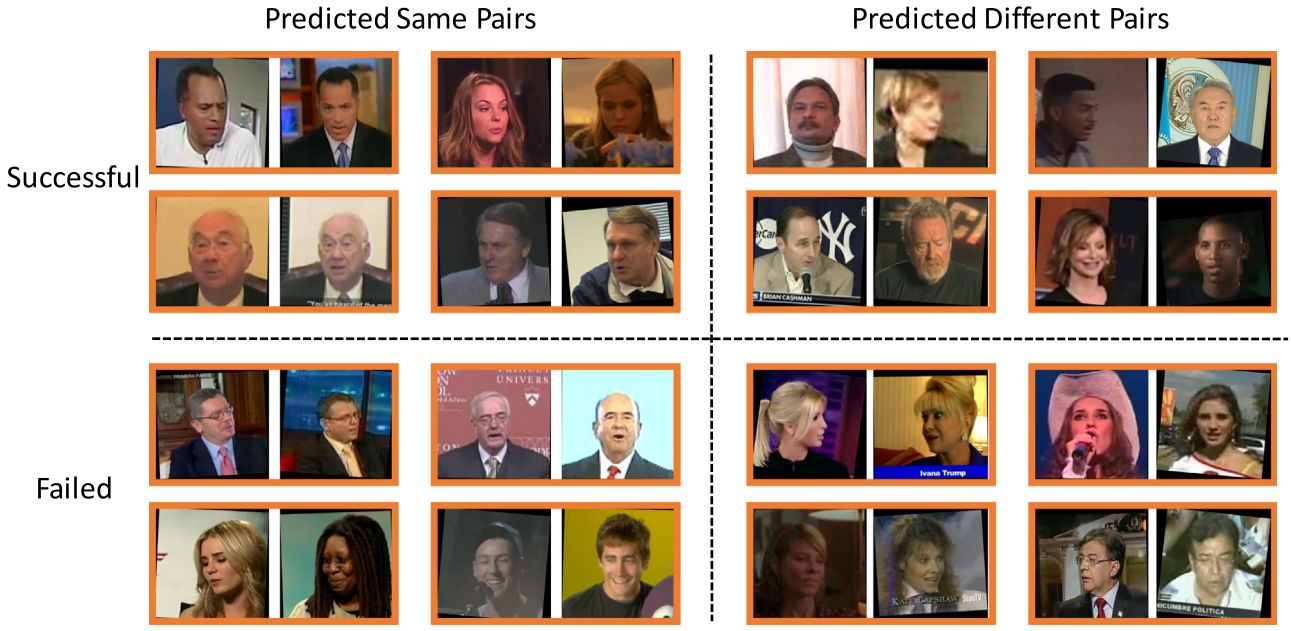


Fig. 4. Some predicted examples on YTF. The predicted same/different pair faces are on the left/right. And the top two rows are successful examples; the bottom two rows are failed examples.

5. Conclusions

In this paper, we have introduced a method to derive optimal Hamming embedding for high-dimensional features in visual recognition tasks. Our method, called Bayesian Hashing, uses a supervised Bayesian learning framework to generate binary codes. To achieve high recognition performance, we have designed a boosted FERNs classification framework to process the binary features. In addition, a random permutation technique was adopted to better exploit the bit correlations and train multiple classification models, where a SFFS algorithm was applied to perform model selection and fusion. Extensive experiments and comparative studies in the context of face recognition clearly demonstrated that the proposed method is able to achieve competitive performance with significantly reduced computation and memory costs.

We would like to emphasize that, although the proposed method was evaluated on the face recognition task, the Bayesian Hashing method is a fairly general supervised hashing technique and can be extended to various applications, such as large scale image search, object recognition, and other problems beyond computer vision.

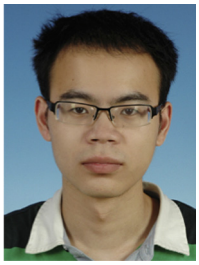
Acknowledgments

This work was supported in part by China's National 863 Program (#2014AA015101), a grant from NSF China (#61572134) and a grant from STCSM (#16QA1400500).

References

[1] O. Barkan, J. Weill, L. Wolf, H. Aronowitz, Fast high dimensional vector multiplication face recognition, in: ICCV, 2013.
[2] D. Chen, X. Cao, F. Wen, J. Sun, Blessing of dimensionality: high-dimensional feature and its efficient compression for face verification, in: CVPR, 2013.
[3] W. Zhao, R. Chellappa, P.J. Phillips, et al., Face recognition: a literature survey, ACM Comput. Surv. 35 (4) (2003) 399–458.
[4] S. Li, A.K. Jain, Handbook of Face Recognition, 2nd edition, vol. 1, New York: Springer, 2011.
[5] L. Wiskott, J.-M. Fellous, et al., Face recognition by elastic bunch graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 775–779.
[6] C. Liu, Capitalize on dimensionality increasing techniques for improving face recognition performance, IEEE Trans. Pattern Anal. Mach. Intell. 28 (5) (2006) 725–737.
[7] T. Ahonen, A. Hadid, M. Pietikainen, Face description with local binary patterns: application to face recognition, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 2037–2041.
[8] K. Mikolajczyk, C. Schmid, A performance evaluation of local descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 27 (10) (2005) 1615–1630.
[9] M. Turk, A. Pentland, Eigenfaces for recognition, J. Cogn. Neurosci. 3 (1) (1991) 71–86.
[10] P. Belhumeur, J. Hespanha, D. Kriegman, Eigenfaces vs.fisherfaces: recognition using class specific linear projection, IEEE Trans. Pattern Anal. Mach. Intell. 19 (7) (1997) 711–720.
[11] X. He, S. Yan, et al., Face recognition using Laplacianfaces, IEEE Trans. Pattern Anal. Mach. Intell. 27 (3) (2005) 328–340.
[12] J. Gui, W. Jia, L. Zhu, S.-L. Wang, D.-S. Huang, Locality preserving discriminant projections for face and palmprint recognition, Neurocomputing 73 (13) (2010) 2696–2707.
[13] N. Kumar, A. Berg, et al., Attribute and simile classifiers for face verification, in: ICCV, 2009.
[14] J. Wright, A.Y. Yang, A. Ganesh, S. Sastry, Y. Ma, Robust face recognition via sparse representation, IEEE Trans. Pattern Anal. Mach. Intell. 31 (2) (2009) 210–227.
[15] M. Guillaumin, J. Verbeek, C. Schmid, Is that you? Metric learning approaches for face identification, in: ICCV, 2009.
[16] M. Kostinger, M. Hirzer, P. Wohlhart, Large scale metric learning from equivalence constraints, in: CVPR, 2012.
[17] H.V. Nguyen, L. Bai, Cosine similarity metric learning for face verification, in: ACCV, 2010.
[18] W.-S. Chen, X. Dai, B. Pan, T. Huang, A novel discriminant criterion based on feature fusion strategy for face recognition, Neurocomputing 159 (2015) 67–77.
[19] B. Heisele, et al., Face recognition with support vector machines: Global versus component-based approach, in: ICCV, 2001.
[20] G.-D. Guo, H.-J. Zhang, Boosting for fast face recognition, in: ICCV Workshop, 2001.
[21] G.B. Huang, H. Lee, E. Learned-Miller, Learning hierarchical representations for face verification with convolutional deep belief networks, in: CVPR, 2012.
[22] Y. Sun, X. Wang, X. Tang, Hybrid deep learning for face verification, in: ICCV, 2013.
[23] Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: Closing the gap to human-level performance in face verification, in: CVPR, 2014.
[24] Y. Sun, X. Wang, X. Tang, Deep learning face representation by joint identification-verification, in: NIPS, 2014.
[25] Y. Su, S. Shan, X. Chen, W. Gao, Hierarchical ensemble of global and local classifiers for face recognition, IEEE Trans. Image Process. 18 (8) (2009) 1885–1896.
[26] C. Huang, S. Zhu, K. Yu, Large Scale Strongly Supervised Ensemble Metric Learning, With Applications to Face Verification and Retrieval, Tech. Rep., NEC Labs America, 2011.
[27] Y. Li, J. Liu, Z. Li, Y. Zhang, H. Lu, S. Ma, Learning Low-rank Representations With Classwise Block-diagonal Structure for Robust Face Recognition, in: AAAI, 2014.

[28] F. Pan, G. Song, X. Gan, Q. Gu, Consistent feature selection and its application to face recognition, J. Intell. Inf. Syst. 43 (2) (2014) 307–321.
[29] M. Yang, P. Zhu, F. Liu, L. Shen, Joint representation and pattern learning for robust face recognition, Neurocomputing 168 (2015) 70–80.
[30] Q. Dai, J. Li, J. Wang, Y. Chen, Y.-G. Jiang, Optimal bayesian hashing for efficient face recognition, in: Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), 2015.
[31] X. Tan, B. Triggs, Enhanced local texture feature sets for face recognition under difficult lighting conditions, IEEE Trans. Image Process. 19 (6) (2010) 1635–1650.
[32] W.-S. Chen, W. Wang, J.-W. Yang, Y.Y. Tang, Supervised regularization locality-preserving projection method for face recognition, Int. J. Wavel. Multiresol. Inf. Process. 10 (06) (2012) 1250053.
[33] O.M. Parkhi, K. Simonyan, A. Vedaldi, A. Zisserman, A compact and discriminative face track descriptor, in: CVPR, 2014.
[34] X. Ben, W. Meng, R. Yan, K. Wang, Kernel coupled distance metric learning for gait recognition and face recognition, Neurocomputing 120 (2013) 577–589.
[35] Y. Bi, B. Fan, F. Wu, Beyond mahalanobis metric: Cayley–Klein metric learning, in: CVPR, 2015.
[36] Z. Huang, R. Wang, S. Shan, X. Chen, Projection metric learning on grassmann manifold with application to video based face recognition, in: CVPR, 2015.
[37] Z. Huang, R. Wang, S. Shan, X. Li, X. Chen, Log-Euclidean metric learning on symmetric positive definite manifold with application to image set classification, in: ICML, 2015.
[38] C. Strecha, A. Bronstein, M. Bronstein, P. Fua, Ldahash: improved matching with smaller descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 34 (1) (2012) 66–78.
[39] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, S.-F. Chang, Supervised hashing with kernels, in: CVPR, 2012.
[40] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, Supervised hashing for image retrieval via image representation learning, in: AAAI, 2014.
[41] D. Zhang, W.-J. Li, Large-scale supervised multimodal hashing with semantic correlation maximization, in: AAAI, 2014.
[42] G. Lin, C. Shen, Q. Shi, A. Hengel, D. Suter, Fast supervised hashing with decision trees for high-dimensional data, in: CVPR, 2014.
[43] F. Shen, C. Shen, W. Liu, H. Tao Shen, Supervised discrete hashing, in: CVPR, 2015.
[44] F. Shen, W. Liu, S. Zhang, Y. Yang, H. Tao Shen, Learning binary codes for maximum inner product search, in: ICCV, 2015.
[45] P. Li, J. Cheng, H. Lu, Hashing with dual complementary projection learning for fast image retrieval, Neurocomputing 120 (2013) 83–89.
[46] M. Ou, P. Cui, F. Wang, J. Wang, W. Zhu, Non-transitive hashing with latent similarity components, in: ACM SIGKDD, 2015.
[47] M. Ou, P. Cui, J. Wang, F. Wang, W. Zhu, Probabilistic attributed hashing, in: AAAI, 2015.
[48] V. Satuluri, S. Parthasarathy, Bayesian locality sensitive hashing for fast similarity search, in: VLDB, 2012.
[49] N. Pinto, D. Cox, Beyond simple features: a large-scale feature search approach to unconstrained face recognition, in: IEEE Conference on FG, 2011.
[50] D. Chen, X. Cao, L. Wang, F. Wen, J. Sun, Bayesian face revisited: a joint formulation, in: ECCV, 2012.
[51] A. Bhattacharyya, On a measure of divergence between two multinomial populations, Sankhyā: Ind. J. Stat. (1946) 401–406.
[52] M. Ozuysal, M. Calonder, V. Lepetit, P. Fua, Fast keypoint recognition using random ferns, IEEE Trans. Pattern Anal. Mach. Intell. 32 (3) (2010) 448–461.
[53] P. Phillips, P. Flynn, T. Scruggs, et al., Overview of the face recognition grand challenge, in: CVPR, 2005.
[54] L. Wolf, T. Hassner, I. Maoz, Face recognition in unconstrained videos with matched background similarity, in: CVPR, 2011.
[55] H. Jégou, M. Douze, C. Schmid, Product quantization for nearest neighbor search, IEEE Trans. Pattern Anal. Mach. Intell. 33 (1) (2011) 117–128.
[56] C.H. Chan, M.A. Tahir, J. Kittler, M. Pietikainen, Multiscale local phase quantisation for robust component-based face recognition using kernel fusion of multiple descriptors, IEEE Trans. Pattern Anal. Mach. Intell. 35 (5) (2012) 1164–1177.
[57] K. Simonyan, O. M. Parkhi, A. Vedaldi, A. Zisserman, Fisher vector faces in the wild, in: BMVC, 2013.
[58] X. Zhu, Z. Lei, J. Yan, D. Yi, S.Z. Li, High-fidelity pose and expression normalization for face recognition in the wild, in: CVPR, 2015.
[59] H. Li, G. Hua, Z. Lin, J. Brandt, J. Yang, Probabilistic elastic matching for pose variant face verification, in: CVPR, 2013.
[60] Z. Cui, W. Li, D. Xu, S. Shan, X. Chen, Fusing robust face region descriptors via multiple metric learning for face recognition in the wild, in: CVPR, 2013.
[61] H. Mendez-Vazquez, Y. Martinez-Diaz, Z. Chai, Volume structured ordinal features with background similarity measure for video face recognition, in: ICB, 2013.
[62] J. Hu, J. Lu, Y.-P. Tan, Discriminative deep metric learning for face verification in the wild, in: CVPR, 2014.



Qi Dai received the B.Sc. degree in computer science from East China University of Science and Technology, Shanghai, China, in 2011. He is currently a Ph.D. student in the School of Computer Science, Fudan University, Shanghai, China. His research interests include multimedia retrieval and computer vision.

Jianguo Li is a Senior Researcher with Intel Labs, Intel Corporation. He got his B.S. degree from Huazhong University of Science and Technology in 2001 with Honor, and his Ph.D. degree from Tsinghua University in July 2006. His research interests include computer vision, machine learning, parallel computing and big data analysis.

Jun Wang received his Ph.D. degree from Columbia University, NY, in 2010. Currently, he is a Professor of School of Computer Science and Software Engineering, East China Normal University, Shanghai, and an adjunct faculty member of Columbia University, New York. He was a Research Staff Member in the Business Analytics and Mathematical Sciences Department at IBM T.J. Watson Research Center, Yorktown Heights, NY. He has been the recipient of several awards and scholarships, including the award of "Youth 1000 Talents Plan" program in 2014 and the Jury thesis award from Columbia University in 2011. His research interests include machine learning, computer vision, mobile intelligence, and hybrid neural-computer vision systems.

Yurong Chen is a Research Director with Intel Labs, Intel Corporation. He received his B.S. and Ph.D. degrees from Tsinghua University in 1998 and 2002, respectively. His research interests include cognitive computing, computer vision, parallel computing, workload analysis and benchmarking.

Yu-Gang Jiang received the Ph.D. degree in computer science from the City University of Hong Kong, Kowloon, Hong Kong, in 2009. During 2008–2011, he was with the Department of Electrical Engineering, Columbia University, New York, NY, first year as a Visiting Scholar and later as a Post-Doctoral Research Scientist. He is currently a Professor of computer science at Fudan University, Shanghai, China. His research interests include multimedia retrieval and computer vision.