Optimal Bayesian Hashing for Efficient Face Recognition

Qi Dai[1], **Jianguo Li**[2], **Jun Wang**[3], **Yurong Chen**[2], **Yu-Gang Jiang**[1]
[1]Shanghai Key Lab of Intelligent Information Processing,
School of Computer Science, Fudan University, Shanghai, China
[2]Intel Labs China, Beijing, China
[3]Institute of Data Science and Technology, Alibaba Group, Seattle, WA, USA
{daiqi, ygj}@fudan.edu.cn, {jianguo.li, yurong.chen}@intel.com, wongjun@gmail.com

Abstract

In practical applications, it is often observed that high-dimensional features can yield good performance, while being more costly in both computation and storage. In this paper, we propose a novel method called *Bayesian Hashing* to learn an optimal Hamming embedding of high-dimensional features, with a focus on the challenging application of face recognition. In particular, a boosted random FERNs classification model is designed to perform efficient face recognition, in which bit correlations are elaborately approximated with a random permutation technique. Without incurring additional storage cost, multiple random permutations are then employed to train a series of classifiers for achieving better discrimination power. In addition, we introduce a sequential forward floating search (SFFS) algorithm to perform model selection, resulting in further performance improvement. Extensive experimental evaluations and comparative studies clearly demonstrate that the proposed Bayesian Hashing approach outperforms other peer methods in both accuracy and speed. We achieve state-of-the-art results on well-known face recognition benchmarks using compact binary codes with significantly reduced computational overload and storage cost.

1 Introduction

For image-based visual recognition tasks, feature representations are often in very high dimensions. Many existing works have demonstrated that the high-dimensional features can yield better accuracies [Jia *et al.*, 2012] than the lower dimensional ones. Face recognition is one of the typical tasks with this phenomenon [Barkan *et al.*, 2013; Chen *et al.*, 2013]. To reduce the cost of both storage and computation, different feature compression (selection, filtering, projection, etc.) techniques have been proposed. This paper takes face recognition as the use-case to study the feature compression problem.

Face recognition has received significant attentions in the past several decades, and it has been applied in a wide range of applications [Zhao *et al.*, 2003; Li and Jain, 2011]. To stimulate research in this area, several real-world benchmarks have recently been created with challenging uncontrolled face images. For instance, Labeled Face in the Wild (LFW) and Face Recognition Grand Challenges (FRGC) have been the popular testbeds for face recognition.

Among all the face recognition techniques developed in recent years, there are roughly two major categories of approaches, i.e., low-level representation design and learning-based methods. The former aims at designing robust handcrafted feature representations, such as Gabor [Wiskott *et al.*, 1997; Liu, 2006], LBP [Ahonen *et al.*, 2006], HoG [Mikolajczyk and Schmid, 2005], etc. The latter leverages modern machine learning techniques in the recognition process. For instance, some methods attempt to learn more discriminative features from the low-level representations or raw images. Representative techniques include subspace based methods [Turk and Pentland, 1991; Belhumeur *et al.*, 1997; He *et al.*, 2005] and mid-level representation (a.k.a. attributes) learning [Kumar *et al.*, 2009; Wright *et al.*, 2009]. Other popular methods build advanced learning models with improved discriminative power. The learning models can be distance metrics [Guillaumin *et al.*, 2009; Kostinger *et al.*, 2012; Nguyen and Bai, 2010], classifiers [Heisele and et al, 2001] and Boosting [Guo and Zhang, 2001]. In addition, some recent works rely on deep learning to simultaneously perform feature learning and model training in a unified framework [Huang *et al.*, 2012; Sun *et al.*, 2013; Taigman *et al.*, 2014; Sun *et al.*, 2014].

In most existing approaches, people tend to employ high-dimensional feature representations or over-complete feature sets since they often have better discriminative ability and can yield higher recognition accuracies [Su *et al.*, 2009; Huang *et al.*, 2011; Chen *et al.*, 2013; Barkan *et al.*, 2013]. However, such high-dimensional face features demand additional storage cost and computational time for training and prediction. To alleviate this issue, several works suggest to learn low-rank representations from the high-dimensional features [Li *et al.*, 2014].

Rather than performing the low-rank matrix recovery, in this paper, we present an efficient way to embed high-dimensional features into a more compact Hamming space for face recognition. In particular, we propose a novel Bayesian framework to encode face features into compact binary codes that require significantly reduced computation and

storage cost. After that, we develop a boosted FERNs based model with a random permutation technique to further exploit bit-level relationships among different binary codes. Finally, we perform sequential forward floating search (SFFS) to select models that lead to high accuracy for face recognition. Different from most existing face recognition techniques that use continuous feature representation, we utilize the proposed Bayesian Hashing framework to extract compact binary codes that can still maintain state-of-the-art recognition performance. We summarize the main contributions below:

(1) We propose a novel Bayesian optimal Hamming embedding method, namely Bayesian Hashing, which can efficiently encode floating point features into compact binary codes.

(2) We build boosted FERNs classification models on bitstreams and exploit bit-level relationships with random permutation technique. Further performance improvement could be obtained by sequential model selection. We show that such a framework performs much better than other classifiers like the SVM for binary codes.

(3) We conduct extensive experiments on two popular face benchmark datasets, i.e., the FRGC and the LFW. The results show that the proposed method outperforms other competing methods in both accuracy and speed.

This paper is organized as follows. 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. Conclusions are drawn in Section 5.

2 Related Works

We divide related works into two categories, namely *feature projection and metric learning* and *supervised hashing*.

2.1 Feature Projection and Metric Learning

High-dimensional feature learning has been extensively studied. For example, subspace methods are considered as one of the first choices to reduce redundancy of high-dimensional features. If we project a d-dimensional raw feature into p-dimensional discriminant subspace, the projection matrix will be of the size $d \times p$. Usually, the original dimension d of the face features is very high, e.g., the popular Gabor features being with tens of thousands of dimensions [Liu, 2006; Tan and Triggs, 2010]. In some over-complete feature learning algorithms like [Huang *et al.*, 2011; Barkan *et al.*, 2013; Chen *et al.*, 2013], the value of d could be more than several hundreds of thousands. Note that learning such a projection matrix could be very costly for high-dimensional cases. To address the scalability issue, researchers employed various techniques, including the divide-and-conquer strategy [Su *et al.*, 2009], sparse projection matrix learning with ℓ_1 regularization [2013], and low-rank representations learning [Li *et al.*, 2014], where promising performance has been achieved with reduced cost in either computation or storage.

Another closely related topic is metric learning, which aims to learn a metric that maximally separates different subject classes. Given features of two face images $\mathbf{v}_i$ and $\mathbf{v}_j$, the distance metric is often defined in a quadric form $d(\mathbf{v}_i, \mathbf{v}_j) = (\mathbf{v}_i - \mathbf{v}_j)^T \mathbf{M} (\mathbf{v}_i - \mathbf{v}_j)$. The metric $\mathbf{M} \in R^{d \times d}$ is a symmetric positive definite matrix that can be decomposed as $\mathbf{M} = \mathbf{A}^T \mathbf{A}$ with $\mathbf{A}$ being a linear transformation matrix. The learned distance function $d(\mathbf{v}_i, \mathbf{v}_j)$ can be incorporated into objectives like the logistic discriminant function [Guillaumin *et al.*, 2009], the Cosine function [Nguyen and Bai, 2010], etc. In [Huang *et al.*, 2011], sparse block diagonal constraints are imposed on the metric matrix $\mathbf{M}$ for fast training. In [Parkhi *et al.*, 2014], the authors proposed a binary representation with metric learning to reduce the dimensionality of high-dimensional Fisher vector (FV) features. In these approaches, however, the metric learning process is often very time consuming for high-dimensional data. In this paper, we propose to use a fast Bayesian Hashing method to encode the original high-dimensional face features to significantly reduce the computation and storage costs.

2.2 Supervised Hashing

The objective of hashing techniques is to map raw features to compact binary codes, where the similarity in the original feature space can be preserved in the binary code space, namely Hamming space. Supervised hashing techniques have attracted increasing attentions in recent years since the goal is to design hash functions that can preserve semantic similarity in the Hamming space. For example, LDA-Hash [Strecha *et al.*, 2012] maximizes the difference between label-same/label-different pairs in the linear discriminative projection space. Supervised Hash with Kernels (KSH) [Liu *et al.*, 2012] 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 [Xia *et al.*, 2014] simultaneously learns the hash functions as well as feature representations. Semantic correlation maximization Hash (SCM) [Zhang and Li, 2014] uses supervised multimodal hashing for large scale data modeling. Note that most of the existing supervised hashing techniques attempt to preserve semantic similarity in the final Hamming space. The proposed Bayesian Hashing method directly aims at minimizing the Bayes error of the classification problem to generate compact hash codes.

Additionally, a method called BayesLSH was proposed in [Satuluri and Parthasarathy, 2012]. Although the name is similar, this work is fundamentally different from ours as it only utilizes the Bayes theorem to estimate the similarity between two existing hash codes.

3 Bayesian Hashing

In order to speed up face recognition using the high-dimensional feature representations, we aim at learning a compact representation from the original features without significant loss of recognition performance. Briefly, we consider the Bayes error as an objective function to find the optimal Hamming embedding. Given the learned hash codes, we further employ the boosted FERNs to perform the classification process. The proposed approach consists of the following three key components:

(1) Bayesian Hashing: We obtain an optimal Hamming embedding by minimizing the Bayes error. To reduce the

computational cost, we use face patches to generate hash codes, as discussed in Sec. 3.1.

(2) Boosted FERNs based classifiers: We model hash bit-stream with boosted FERNs classifiers. Details are discussed in Sec. 3.2.

(3) Bit permutation: To exploit the relationships among different patches, we introduce a bit-stream permutation technique, which leads to multiple random FERNs models. Then the sequential forward floating search is applied to select a few good permutation models, as discussed in Sec. 3.3.

For the raw face features, We use Gradient location-orientation histogram (GLOH) in our implementation [Mikolajczyk and Schmid, 2005]. In particular, given the face images with some landmark points, we extract n patches around landmarks with different scales. In addition, we further make a mirror of each face image and extract another n more patches. Thus, we obtain a total of $K = 2n$ patches, each having 17 block segments with a 8 dimensional histogram-style feature. Finally, we represent each patch with a d_0=136 dimensional feature vector.

We use a pair-wise (not limited to) representation for face recognition as suggested in [Pinto and Cox, 2011]. Given a pair of face images $\boldsymbol{v}_i$ and $\boldsymbol{v}_j \in \mathbb{R}^d$ (d=K*d_0), let $\boldsymbol{x}_{ij}$ be the pair representation which is element-wise absolute-difference $\boldsymbol{x}_{ij} = (\|\boldsymbol{v}_{i1} - \boldsymbol{v}_{j1}\|^p, ..., \|\boldsymbol{v}_{id} - \boldsymbol{v}_{jd}\|^p)$. We then define y = 1 if $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ are from the same subject, and $y = -1$ otherwise. Thus, we obtain a positive sample set $\mathcal{P}$ from the same-subject pairs, and a negative set $\mathcal{N}$ from the other pairs.

3.1 Bayesian Hashing: Formulation

Assuming 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 $P(\boldsymbol{x}|y = -1) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_n, \boldsymbol{\Sigma}_n)$, the log-ratio discriminant function is

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

where $\boldsymbol{b}$ is the bias parameter for the prior [1].

The Bayesian Hashing seeks a set of hash functions $\hat{y} = h(\boldsymbol{x}; \boldsymbol{b})$ by minimizing Bayes error on training set,

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

We will start from the simplest case with only 1-dimensional feature and then extend to multi-dimensional settings.

1-dimensional case

In this case, we have $P(x|y = 1) = \mathcal{N}(x|\mu_p, \sigma_p)$ and $P(x|y = -1) = \mathcal{N}(x|\mu_n, \sigma_n)$. Then Eq (1) can be rewritten as

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

[1] The Bayesian Hashing is not limited to univariate discriminant function. We may easily extend our framework to bi-variables (pair inputs) form $\mathcal{G}(\boldsymbol{x}_1, \boldsymbol{x}_2)$, and obtain the discriminant function as in [Kostinger *et al.*, 2012] or [Chen *et al.*, 2012]. Due to space limitation, we will not describe the deduction of hash function for bi-variable discriminant function in this paper.

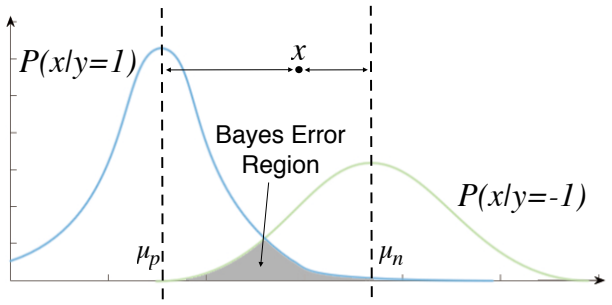

Figure 1: Illustration of hash function $h(x)$ in the 1-dimensional case. The objective is to minimize the Bayes error, where the sample x will be assigned to class y=-1 in this example.

Assuming $\sigma_p = \sigma_n = \sigma_a$, we have the hash function for 1-dimensional case as

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

In practice, we obtain b by applying a line search algorithm (like golden section search) to minimize the cost in Eq (2). Figure 1 illustrates how the hash function $h(x; b)$ works in the 1-dimensional case.

d-dimensional case

For the d-dimensional case with d>1, we follow the two-stage procedure like many Hamming embedding algorithms [Strecha *et al.*, 2012]. At the first stage, we perform a de-correlation subspace projection. There are many different criteria which could be used for this objective. This paper adopts the Fisher criterion to maximize the separation between $\mathcal{P}$ and $\mathcal{N}$ by the ratio of variances on $\mathcal{P}$ and $\mathcal{N}$,

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

With Lagrange multipliers, this gives the projection $\boldsymbol{\Sigma}^{-1}$, where $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}_p + \boldsymbol{\Sigma}_n$. Observing that $\boldsymbol{\Sigma}$ is a symmetric positive semi-definite matrix, it has an SVD decomposition $\boldsymbol{\Sigma}^{-1} = (USU^T)^{-1} = (U^T S^{-\frac{1}{2}})(S^{-\frac{1}{2}} U) = A^T A$, where $A = S^{-\frac{1}{2}} U$. Replacing $\boldsymbol{\Sigma}_p$ and $\boldsymbol{\Sigma}_n$ with $A^T A$, Eq (1) can be simplified to

$$\begin{aligned}\mathcal{G}(\boldsymbol{x}) = &- (\boldsymbol{x}' - \boldsymbol{\mu}'_p)^T (\boldsymbol{x}' - \boldsymbol{\mu}'_p) \\ &+ (\boldsymbol{x}' - \boldsymbol{\mu}'_n)^T (\boldsymbol{x}' - \boldsymbol{\mu}'_n) + \mathbf{1} \cdot \boldsymbol{b},\end{aligned} \tag{5}$$

where $\boldsymbol{x}'$ and $\boldsymbol{\mu}'$ are all in the projection space (i.e., $\boldsymbol{x}' = A\boldsymbol{x}, \boldsymbol{\mu}' = A\boldsymbol{\mu}$). At the second stage, we derive a set of hash functions $h_i(x'_i; b_i)$ for each element of $\boldsymbol{x}'$, similar to Eq (4).

Different from the two-stage procedures in [Strecha *et al.*, 2012], our objective is to minimize the Bayes error in Eq (2) to train supervised hash functions in Eq (4). To the best of our knowledge, we are the first to employ Bayes error in learning a Hamming embedding. Notice that the full feature dimension d of the images is very high. Instead of applying subspace projection on the whole d dimensional feature space, we perform subspace projection for each *face patch* with a relatively lower feature dimension.

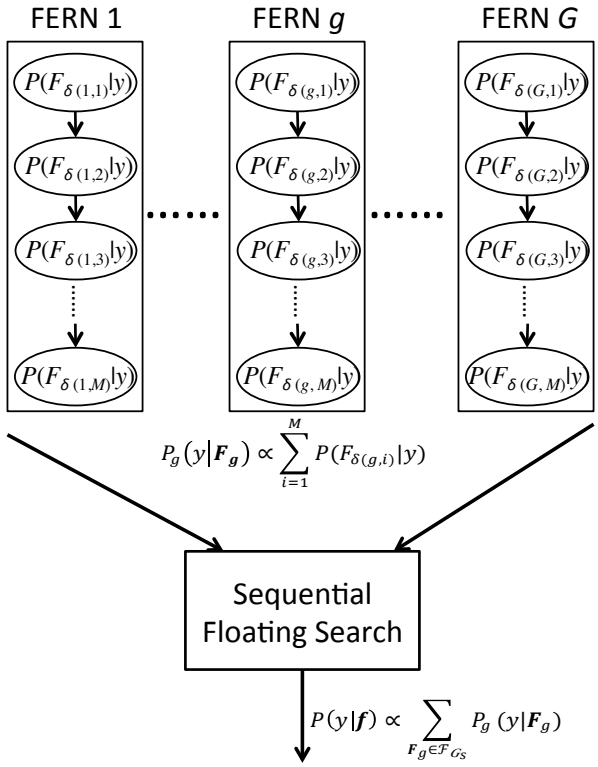

Figure 2: The framework of the boosted FERNs classification model. Each column indicates one bit-stream permutation, which corresponds to one boosted FERNs classifier. Each circle is one FERN byte with the number of bytes $T = M$ in this example.

3.2 Boosted FERNs

Given a high-dimensional feature input $\boldsymbol{x}$, Bayesian Hashing will generate a bit sequence $\boldsymbol{f} = (f_1, \cdots, f_D)$, where $f_i \in \{0, 1\}$. We then model the bit stream with a random boosted FERNs framework [Ozuysal *et al.*, 2010]. Figure 2 shows our configuration of the boosted random FERNs.

Generally, given bit-sequence $\boldsymbol{f}$, we partition them into M groups of size $S = \frac{D}{M}$, obtaining a set of feature groups $\boldsymbol{F} = (F_1, \cdots, F_M)$. Each group F_i is called a FERN, and is modeled with a Naive Bayesian classifier $P(F_i|y)$. Then the joint probability for all FERNs is computed by $P(\boldsymbol{F}|y) = \prod_{i=1}^{M} P(F_i|y)$, assuming that all groups are independent. In our implementation, we group every 8-bit ($S = 8$) together to one byte F. In addition, to reduce possible redundancy among different bytes, we adopt the Boosted framework. We take each FERN as a weak classifier, and train a GentleBoost, a variant of AdaBoost classifiers, to ensemble different FERN bytes. Table 1 summarizes the training algorithm of the boosted FERNs.

During the training procedure, we restrict that each FERN byte can be chosen only once to avoid redundancy. Finally, we can obtain a decision function for recognition as

$$\mathcal{H}(\boldsymbol{F}) = \sum_{i=1}^{T} \{P(F_i, y = 1) - P(F_i, y = -1)\}, \quad (6)$$

where $T \leq M$ is the number of bytes used/picked. This decision function is in fact a summation of a set of look-up-tables (LUTs), which makes the prediction very efficient. In addition, these LUTs can be further quantized into 8-bit char type so that the model storage can be further reduced.

Table 1: Boosted FERNs training algorithm using GentleBoost.

Input: Training set: $\{(F_i^k, y_i)\}_{i=1}^{N}$, $k = 1 : M$, where N is the number of samples and M is the number of FERN bytes. F^k is k-th FERN byte.
Initialize: Iteration number T. $t = 0$. Initialize weight for positive samples and negative samples.

(1) $w_{1,i}^{+} = \frac{1}{N_+}$ for those y_i=1;

(2) $w_{1,i}^{-} = \frac{1}{N_-}$ for those y_i= -1.

Step 1: For each byte F^k, build Naive Bayesian probability look-up table $P(F^k|y)$;
Step 2: Pick the best byte $F^{\sigma(t)}$ according to the Bayes error on the training set, and define decision function as

$$h_t(F^{\sigma(t)}) = P(F^{\sigma(t)}, y = 1) - P(F^{\sigma(t)}, y = -1).$$

Note, here we restrict that each byte can be taken only once.
Step 3: Update weight $w_{t,i} = w_{t-1,i} \cdot \exp[-y_i h_t(F_i^{\sigma(t)})]$; and normalize the weight so that $\sum_i w_{t,i}^{+}$ =1 and $\sum_i w_{t,i}^{-}$ =1;
Step 4: $t = t$+1. If $t = T$, break; else go to Step 1.
Output: Strong classifier $\mathcal{H}_T(\boldsymbol{F}) = \sum_{t=1}^{T} h_t(F^{\sigma(t)})$.

3.3 Bit Permutation and Model Selection

As discussed in Sec. 3.1, with high dimensional raw features, it is inefficient to perform Hamming embedding on the whole feature space since the projection procedure will be fairly costly. To address this issue, we conduct patch-level Bayesian Hashing. In order to exploit possible relationships between different patches, we introduce a bit-stream permutation technique. As illustrated in Figure 2, a total of G permutations are generated. After performing permutation, bits in the same byte could come from different patches. Given the original hash code $\boldsymbol{f}$, the g-th random permutation is defined as $\boldsymbol{f}_g = \{f_{\delta_{(g,1)}}, \cdots, f_{\delta_{(g,D)}}\}$, where $\delta(,)$ denotes the indices after permutation.

For a set of bit-stream permutations, each permutation δ_g will produce a boosted FERNs model $\mathcal{H}(\boldsymbol{F}_{\delta_g})$. Intuitively, more permutations will produce better discrimination power to gain more performance improvement. However, this also incurs possible redundancy and additional cost. By introducing the Sequential Forward Floating Search (SFFS), we are able to select an optimal set of permutation models to achieve improved performance without additional overhead. Note that the random permutation will not increase the storage significantly since the Hamming embedding is already fixed. Table 2 shows the details of the SFFS algorithm.

4 Experiments

4.1 Datasets and Experimental Settings

FRGC: The FRGC version-2 [Phillips *et al.*, 2005] was designed to be a comprehensive benchmark for face recognition. We focus on experiment-4 (known as FRGC-204), which

Table 3: Comparison with Product Quantization and Supervised Hashing with Kernels on FRGC. Our method is consistently better.

Methods	TPR@FPR=0.1%	Training Time (Seconds)	Testing Time (Seconds)	Memory (MB)
PQ [Jégou *et al.*, 2011] + Boosted FERNs	89.50%	20915.8	3624.2	57.4
KSH [Liu *et al.*, 2012] + Boosted FERNs	59.20%	433569.6	683.6	73.2
Bayesian Hashing + Boosted FERNs	90.35%	3873.3	213.9	34.2

Table 2: SFFS algorithm for permutation selection.

Input: Random permutation feature grouping set $\mathcal{F} = \{\boldsymbol{F}_g, 1 \leq g \leq G\}$. $J(\mathcal{F}_k)$ measures classification accuracy based on permutation set $\mathcal{F}_k$.
Initialize: $\mathcal{F}_0 = \varnothing, k = 0$, preset permutation number G_S.
Step 1: Inclusion

- Find the best permutation $\boldsymbol{F}^+ = \arg\max_{\boldsymbol{F} \in \mathcal{F} \backslash \mathcal{F}_k} J(\mathcal{F}_k \cup \boldsymbol{F})$, where $\mathcal{F} \backslash \mathcal{F}_k$ is the set $\mathcal{F}$ excludes subset $\mathcal{F}_k$;
- $\mathcal{F}_{k+1} = \mathcal{F}_k \cup \boldsymbol{F}^+; k = k + 1$;

Step 2: Conditional exclusion

- Find the worst permutation $\boldsymbol{F}^- = \arg\max_{\boldsymbol{F} \in \mathcal{F}_k} J(\mathcal{F}_k - \boldsymbol{F})$;
- If $J(\mathcal{F}_k - \boldsymbol{F}^-) > J(\mathcal{F}_{k-1})$, then $\mathcal{F}_{k-1} = \mathcal{F}_k - \boldsymbol{F}^-, k = k - 1$, go to Step 2; else go to Step 3.

Step 3: If $k \geq G_S$, break; else go to Step 1.
Output: Good permutation grouping subset $\mathcal{F}_k$.

contains 12,776 training faces, 16,028 target face images and 8,014 query faces. The target face images are taken under controlled environment, while query faces are taken under uncontrolled environment. This makes the FRGC-204 the most challenging task in the FRGC benchmark. The dataset provides manual annotations for 4 landmarks (eye centers, nose tip and mouth center). Each face is normalized to 128×128 according to these landmarks. We extract n=240 patches from the normalized faces according to the landmark positions with different patch sizes. A mirror image is generated for each normalized face and another n=240 patches can be extracted. Therefore, there are totally 480 GLOH patches, and each patch is described by a 136-dimensional feature.

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

LFW: The popular LFW dataset consists of 13,233 images of 5,749 individuals, and all the images are collected from the Internet. We adopt the LFW-a (the aligned version of LFW), and similar settings are adopted to for data preprocessing following the previous works on this dataset.

There are several evaluation protocols for LFW. We conduct experiments strictly following the *unrestricted with label-free outside data* protocol. The evaluation dataset is divided into 10 subsets to form a 10-fold cross-validation. In each trail, we use nine subsets for training and one for testing. We report the mean Equal Error Rate (EER). ROC curves are also plotted.

4.2 Results and Discussions

We conduct several experiments to analyze the impacts from different components of the proposed approach.

Bayesian Hashing: We first evaluate our Bayesian Hashing and compare with two alternative hashing methods: Product Quantization (PQ) [Jégou *et al.*, 2011] and Supervised Hashing with Kernels (KSH) [Liu *et al.*, 2012]. The number of bytes for PQ is the same as that of Bayesian Hashing (8,160). For KSH, because the training process is too expensive, we only manage to train 4,800 hash functions (600 bytes). Table 3 summarizes the results on FRGC and Figure 3(c) shows the corresponding ROC curves. Bit-permutation is not adopted for all the three methods. As shown in the table, Bayesian Hashing has higher accuracy. More importantly, it significantly reduces the computation and memory cost. Our Bayesian Hashing still outperforms KSH with a significant margin when only 500 bytes are used (68.40% vs. 59.20%). For PQ, the accuracy is just slightly lower, but is over 10x slower than our Bayesian Hashing in testing time.

Boosted FERNs classification: Next, we examine the effectiveness of the boosted FERNs and compare with SVM. Table 4 shows the results on FRGC (the bottom five rows). The boosted FERNs classification plays an important role in our method. Replacing it with SVM, we only obtain 79.78% on FRGC using the same set of binary codes. Besides, we also train an SVM classifier on the raw floating point GLOH features, which produces an accuracy of 93.72%. In comparison, the proposed method can achieve 93.20%, while requiring lower computation and memory costs.

Impact of the number of FERN bytes: We now study the impact of the number of the selected FERN bytes. Bit-permutation is 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. The boosting training considers the FERN bytes sequentially according to their contributions. Figure 3(a) and 3(d) plot the performances vs. different numbers of bytes on FRGC and LFW, respectively.

For FRGC, the accuracy tends to be stable after 4,000 bytes, and the trend on LFW is more or less the same. It is worth noting that the original Bayesian Hashing with 8,160 bytes can already yield a 32x compression rate of the original high-dimensional floating point features. If only using 4,000 bytes, the feature compression ratio is more than 64x with only 2% accuracy drop.

This experiment also indicates a very promising property of our method that we can possibly adopt a 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 1,000 bytes, which can quickly narrow down the search space for fine-grained computations with more bytes.

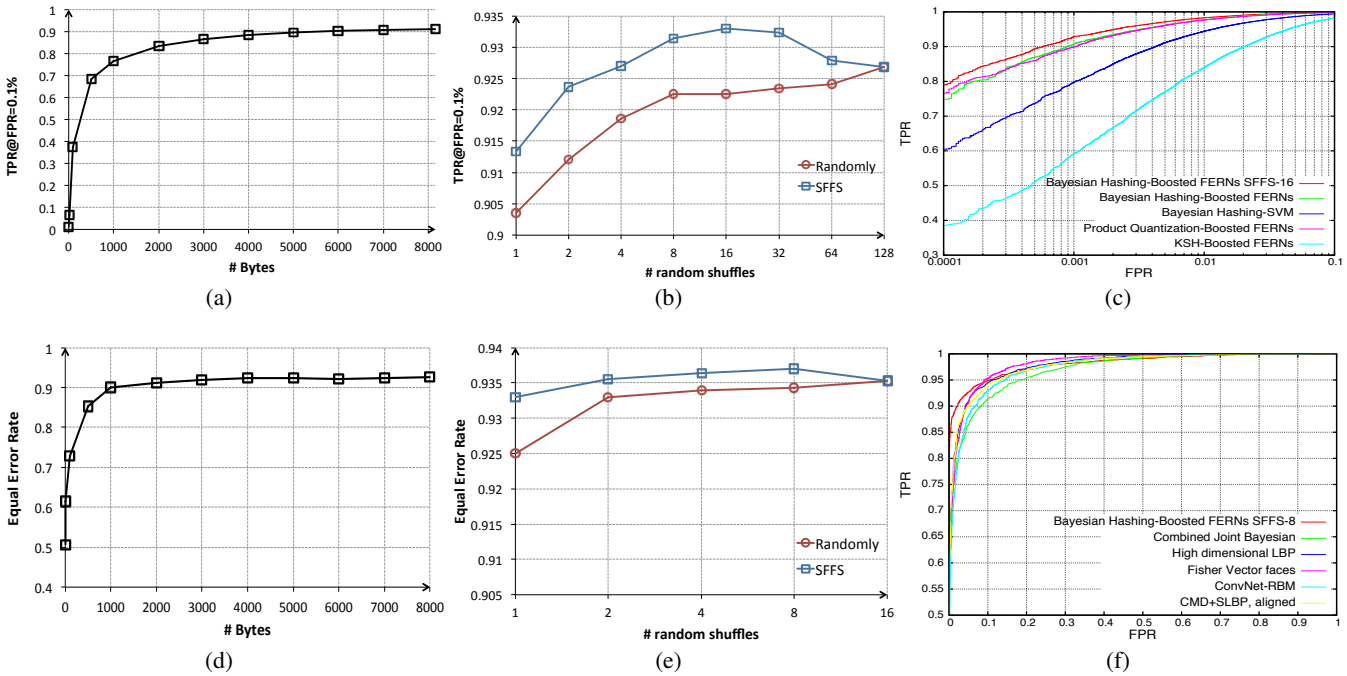

Figure 3: First row: results on FRGC. Second row: results on LFW. (a) and (d) are the accuracies with different numbers of bytes (w/o bit-permutation). (b) and (e) are results with different numbers of bit permutations. (c) and (f) are ROC curves. See texts for more explanations.

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

Methods	TPR@ FPR=0.1%
Baseline, eigenface [Phillips *et al.*, 2005]	12%
Gabor + Kernel [Liu, 2006]	76%
LTP + Gabor + Kernel [Tan and Triggs, 2010]	88.5%
Gabor + Fourier [Su *et al.*, 2009]	89%
LPQ-fusion [Chan *et al.*, 2012]	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%

Impact of bit permutation: We train multiple boosted FERNs models using different random permutations. Figures 3(b) and 3(e) shows the accuracy vs. different numbers of random permutations on FRGC and LFW, respectively. It is clear that the accuracy grows with an increasing number of shuffles, especially at the beginning parts of the curves (# shuffles $\leq$8).

We also study the effectiveness of SFFS on both FRGC and LFW. Given a large set of different permutation models, we adopt SFFS to select good and complementary ones by evaluating on a separate validation set. As shown in the Figures 3(b) and 3(e), SFFS is able to further improve the results with only a small number of selected permutations on both datasets.

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

Methods	EER (%)
Combined Joint Bayes [Chen *et al.*, 2012]	90.90±1.48
CMD+SLBP [Huang *et al.*, 2011]	92.58±1.36
VMRS [Barkan *et al.*, 2013]	92.05±0.45
ConvNet-RBM [Sun *et al.*, 2013]	91.75±0.48
Fisher Vector Faces [Simonyan *et al.*, 2013]	93.03±1.05
high-dim LBP [Chen *et al.*, 2013]	93.18±1.07
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

4.3 Comparison with State of the Arts

In this subsection, we compare our results with several state-of-the-art works. Table 4 and Table 5 summarize the results on FRGC and LFW respectively, where "perm-x" indicates x random permutations. On both datasets, we achieve very competitive results. It even outperforms the deep learning based approach [Sun *et al.*, 2013], which is very appealing as we only adopt the traditional hand-crafted GLOH features. Notice that our method can be deployed on any type of features, including the powerful deep learning based ones. We expect a significant gain may be achieved by replacing GLOH with the recently developed deeply learning features. Figure 3(f) further shows the ROC curves of our method and the compared approaches on LFW. For FRGC, we do not have the data needed for plotting ROC curves of the compared works. Finally, we would like to emphasize again that our results are

obtained using binary codes that possess the nice property of significantly lower computation and memory costs, while all the compared works reply on the expensive floating features.

5 Conclusions

Recent advances in visual recognition tasks have shown that high-dimensional feature representations often yield high accuracies, while suffering from heavy computational overload and expensive memory cost. In this paper, we have presented a novel method to derive optimal Hamming embedding for high-dimensional features, namely Bayesian Hashing, with a focus on the challenging application of face recognition. To achieve high recognition performance, we also designed a boosted FERNs classification framework to handle the binary features. In addition, a random permutation technique was used to better exploit bit correlations and train multiple classification models, where a SFFS algorithm can be applied to perform model selection and fusion. Extensive experiments and comparison studies using two popular face recognition benchmarks clearly demonstrated that the proposed method achieved competitive performance with significantly reduced computation and memory costs.

Although the proposed method was evaluated in the face recognition task, the Bayesian Hashing technique is a fairly general supervised hashing method and can be extended to various computer vision applications, such as large scale image search and object recognition. One of our future directions is to design a unified framework to learn the Hamming embedding and train classification models simultaneously for general computer vision tasks.

Acknowledgments

This work was supported in part by the National 863 Program of China (#2014AA015101), a grant from NSF China (#61201387) and two grants from the STCSM (#13PJ1400400, #13511504503).

References

[Ahonen *et al.*, 2006] T. Ahonen, A. Hadid, and M. Pietikainen. Face description with local binary patterns: Application to face recognition. *IEEE TPAMI*, 28, 2006.

[Barkan *et al.*, 2013] Oren Barkan, Jonathan Weill, Lior Wolf, and Hagai Aronowitz. Fast high dimensional vector multiplication face recognition. In *ICCV*, 2013.

[Belhumeur *et al.*, 1997] P. Belhumeur, J. Hespanha, and D. Kriegman. Eigenfaces vs.fisherfaces: recognition using class specific linear projection. *IEEE TPAMI*, 19, 1997.

[Chan *et al.*, 2012] C. H. Chan, M. A. Tahir, J. Kittler, and M. Pietikainen. Multiscale local phase quantisation for robust component-based face recognition using kernel fusion of multiple descriptors. *IEEE TPAMI*, 2012.

[Chen *et al.*, 2012] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun. Bayesian face revisited: A joint formulation. In *ECCV*, 2012.

[Chen *et al.*, 2013] D. Chen, X. Cao, F. Wen, and J. Sun. Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification. In *CVPR*, 2013.

[Guillaumin *et al.*, 2009] M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *ICCV*, 2009.

[Guo and Zhang, 2001] G.-D. Guo and H.-J. Zhang. Boosting for fast face recognition. In *ICCV workshop*, 2001.

[He *et al.*, 2005] X. He, S. Yan, and et al. Face recognition using laplacianfaces. *IEEE TPAMI*, 2005.

[Heisele and et al, 2001] B. Heisele and et al. Face recognition with support vector machines: Global versus component-based approach. In *ICCV*, 2001.

[Huang *et al.*, 2011] C. Huang, S. Zhu, and K. Yu. Large scale strongly supervised ensemble metric learning, with applications to face verification and retrieval. Technical report, NEC Labs America, 2011.

[Huang *et al.*, 2012] G. B. Huang, H. Lee, and E. Learned-Miller. Learning hierarchical representations for face verification with convolutional deep belief networks. In *CVPR*, 2012.

[Jégou *et al.*, 2011] H. Jégou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE TPAMI*, 2011.

[Jia *et al.*, 2012] Y. Jia, C. Huang, and T. Darrell. Beyond spatial pyramids: Receptive field learning for pooled image features. In *CVPR*, 2012.

[Kostinger *et al.*, 2012] M. Kostinger, M. Hirzer, and P. Wohlhart. Large scale metric learning from equivalence constraints. In *CVPR*, 2012.

[Kumar *et al.*, 2009] N. Kumar, A. Berg, and et al. Attribute and simile classifiers for face verification. In *ICCV*, 2009.

[Li and Jain, 2011] S. Li and A. K. Jain. *Handbook of Face recognition*. Springer, 2nd edition, 2011.

[Li *et al.*, 2014] Y. Li, J. Liu, Z. Li, Y. Zhang, H. Lu, and S. Ma. Learning low-rank representations with classwise block-diagonal structure for robust face recognition. In *AAAI*, 2014.

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

[Liu, 2006] C. Liu. Capitalize on dimensionality increasing techniques for improving face recognition performance. *IEEE TPAMI*, 2006.

[Mikolajczyk and Schmid, 2005] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE TPAMI*, 2005.

[Nguyen and Bai, 2010] H. V. Nguyen and L. Bai. Cosine similarity metric learning for face verification. In *ACCV*, 2010.

[Ozuysal *et al.*, 2010] M. Ozuysal, M. Calonder, V. Lepetit, and P. Fua. Fast keypoint recognition using random ferns. *IEEE TPAMI*, 2010.

[Parkhi *et al.*, 2014] O. M. Parkhi, K. Simonyan, A. Vedaldi, and A. Zisserman. A compact and discriminative face track descriptor. In *CVPR*, 2014.

[Phillips *et al.*, 2005] P. Phillips, Patrick Flynn, Todd Scruggs, and et al. Overview of the face recognition grand challenge. In *CVPR*, 2005.

[Pinto and Cox, 2011] N. Pinto and David Cox. Beyond simple features: A large-scale feature search approach to unconstrained face recognition. In *IEEE Conf. FG*, 2011.

[Satuluri and Parthasarathy, 2012] Venu Satuluri and Srinivasan Parthasarathy. Bayesian locality sensitive hashing for fast similarity search. In *VLDB*, 2012.

[Simonyan *et al.*, 2013] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman. Fisher Vector Faces in the Wild. In *BMVC*, 2013.

[Strecha *et al.*, 2012] C. Strecha, A.M. Bronstein, M.M. Bronstein, and P. Fua. Ldahash: Improvedmatching with smaller descriptors. *IEEE TPAMI*, 2012.

[Su *et al.*, 2009] Y. Su, S. Shan, X. Chen, and W. Gao. Hierarchical ensemble of global and local classifiers for face recognition. *IEEE TIP*, 2009.

[Sun *et al.*, 2013] Y. Sun, X. Wang, and X. Tang. Hybrid deep learning for face verification. In *ICCV*, 2013.

[Sun *et al.*, 2014] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, 2014.

[Taigman *et al.*, 2014] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, 2014.

[Tan and Triggs, 2010] X. Tan and B. Triggs. Enhanced local texture feature sets for face recognition under difficult lighting conditions. *IEEE TIP*, 2010.

[Turk and Pentland, 1991] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neurosicence*, 1991.

[Wiskott *et al.*, 1997] L. Wiskott, J.-M. Fellous, and et al. Face recognition by elastic bunch graph matching,. *IEEE TPAMI*, 1997.

[Wright *et al.*, 2009] J. Wright, A. Y. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE TPAMI*, 31, 2009.

[Xia *et al.*, 2014] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *AAAI*, 2014.

[Zhang and Li, 2014] D. Zhang and W.-J. Li. Large-scale supervised multimodal hashing with semantic correlation maximization. In *AAAI*, 2014.

[Zhao *et al.*, 2003] W. Zhao, R. Chellappa, P. J. Phillips, and et al. Face recognition: A literature survey. *ACM Computing Surveys*, 22, 2003.