Fried Binary Embedding: From High-Dimensional Visual Features to High-Dimensional Binary Codes

Weixiang Hong and Junsong Yuan, *Senior Member, IEEE*

***Abstract*— Most existing binary embedding methods prefer compact binary codes (b-dimensional) to avoid high computational and memory cost of projecting high-dimensional visual features (d-dimensional, $b < d$). We argue that long binary codes ($b \sim \mathcal{O}(d)$) are critical to fully utilize the discriminative power of high-dimensional visual features, and can achieve better results in various tasks such as approximate nearest neighbor search. Generating long binary codes involves large projection matrix and high-dimensional matrix-vector multiplication, thus is memory and compute intensive. We propose Fried binary embedding (FBE) and Supervised Fried Binary Embedding (SuFBE), to tackle these problems. FBE is suitable for most of the practical applications in which the labels of training data are not given, while SuFBE can significantly boost the accuracy in the cases that the training labels are available. The core idea is to decompose the projection matrix using adaptive Fastfood transform, which is the multiplication of several structured matrices. As a result, FBE and SuFBE can reduce the computational complexity from $\mathcal{O}(d^2)$ to $\mathcal{O}(d \log d)$, and memory cost from $\mathcal{O}(d^2)$ to $\mathcal{O}(d)$, respectively. More importantly, by using the structured matrices, FBE and SuFBE can well regulate projection matrix by reducing its tunable parameters and lead to even better accuracy than using either unconstrained projection matrix (like ITQ) or sparse matrix such as SP and SSP with the same long code length. Experimental comparisons with state-of-the-art methods over various visual applications demonstrate both the efficiency and performance advantages of FBE and SuFBE.**

***Index Terms*— Binary embedding, image retrieval.**

I. Introduction

NEAREST neighbor (NN) search has been a fundamental research topic in machine learning, computer vision, and information retrieval [4]. The straightforward solution, linear scan, is memory intensive and computationally expensive in large-scale high-dimensional cases; hence approximate nearest neighbor (ANN) search is usually favored in practice.

Binary embedding [1], [5], which aims at encoding high-dimensional feature vectors to compact binary codes, has recently arisen as an effective and efficient way for ANN search. By encoding high- dimensional features into binary codes, one can perform rapid ANN search because (1) operations with binary vectors (such as computing Hamming distance) are very fast thanks to hardware support, and (2) the entire dataset can fit in (fast) memory rather than slow memory or disk. Since it is NP-hard to directly learn the optimal binary codes [5], most existing binary embedding methods work on a two-stage strategy: projection and quantization. Specifically, given a feature vector $\mathbf{x} \in \mathbb{R}^d$, these methods first multiply $\mathbf{x}$ with a projection matrix $\mathrm{R} \in \mathbb{R}^{b \times d}$ to produce a low-dimensional vector of b dimensions, then quantize this low-dimensional vector to b-dimensional binary codes by assigning it to its nearest vertex in Hamming space.

Representation learning using deep neural networks (DNN) [6], [7] has shown that DNN features are useful for various vision tasks such as object classification and image retrieval. Unlike traditional hand-crafted features like SIFT [8] and GIST [9], these DNN features are usually of thousands of dimensions or even more. Meanwhile, although compact binary codes are preferred to save the storage, recent works have demonstrated that long-bit codes can bring superior performance than compact ones, especially when the visual features are of thousands of dimensions. For example, the long binary codes of 4096 dimensions can achieve mAP at 82% on DNN-4096 dataset [2], while the mAP of 256-dimensional binary codes is only 51%.

However, generating long binary codes requires a large projection matrix, which leads to two challenges: (1) the high computational cost of high-dimensional matrix-vector multiplication, and (2) the risk of overfitting. For the first challenge, it has been noticed that for input feature vector of dimensionality d, the length b of binary codes required to achieve reasonable accuracy is usually $\mathcal{O}(d)$ [10]–[15]. When d is large and $b \sim \mathcal{O}(d)$, the projection matrix $\mathrm{R} \in \mathbb{R}^{b \times d}$ could involve millions or even billions of parameters. Such a high cost is not favored when we encode a big dataset of visual features, or when the computational resource is a concern, *e.g.*, at the mobile platform. For the second challenge, there have been efforts to address it by regulating the projection matrix and reducing the degree of free parameters. Interestingly, such regularizations may also bring fast matrix-vector multiplication, which will benefit the computation efficiency as well.

To address the two challenges above, we propose Fried Binary Embedding (FBE) and Supervised Fried Binary Embedding (SuFBE), for generating effective long binary

Manuscript received August 19, 2017; revised March 20, 2018; accepted May 26, 2018. Date of publication June 12, 2018; date of current version June 29, 2018. This work was supported in part by the Singapore Ministry of Education Academic Research Fund Tier-2 under Grant MOE2015-T2-2-114 and in part by startup funds from University at Buffalo. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Aydin Alatan. *(Corresponding author: Weixiang Hong.)*

W. Hong is with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: wxhong@ntu.edu.sg).

J. Yuan is with the Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY 14260-2500 USA (e-mail: jsyuan@buffalo.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2846670

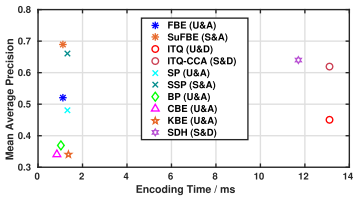


Fig. 1. Retrieval accuracy versus encoding time on the CIFAR-10 dataset [20] using 4096-dimensional AlexNet [6] features. We compare the proposed FBE and SuFBE against several state-of-the-art methods including ITQ [1], CCA-ITQ [1], SP [2], SSP [3], BP [11], CBE [12], KBE [24] and SDH [19]. We use tag "U" and "S" to be short for "Unsupervised" and "Supervised," "A" and "D" to be short for "Accelerated Matrix-Vector Multiplication" and "Dense Matrix-Vector Multiplication."

codes efficiently. The idea is to decompose the projection matrix R using the adaptive Fastfood transform [16], [17], which is the multiplication of several structured matrices. Structured matrix typically consists of dependent entries, which means that a fixed "budget of freedom" is distributed across the matrix. The involvement of structured matrices leads to fast matrix-vector multiplications by using Fast Fourier Transform or its variants. Moreover, the ultimate projection matrix R would have restricted freedom due to the inherent structure in each of its components. For example, when encoding a 4096-dimensional feature vector into 4096-bit binary codes, our FBE has only 12,288 tunable parameters, which are only 1% of Sparse Projection [2] and 0.1% of ITQ [1]. Restricted freedom is naturally against overfitting, thus can probably lead to good generalization performance. Another side benefit of FBE is that structured matrix can be efficiently stored with linear complexity $\mathcal{O}(d)$, or even do not need to be explicitly stored. As a result, the memory cost of storing R is also significantly reduced.

A preliminary version of FBE has been published in CVPR 2017 [18]. Based on the conference version, we propose Supervised Fried Binary Embedding (SuFBE) to extend FBE for the supervised binary embedding cases. The labels of training data are typically not provided in most of the practical applications, nevertheless, in case that the training labels are available, supervised hashing can marginally outperform their unsupervised counterparts by utilizing the training labels [1], [3], [19]. As shown in Figure 1, the mAP of FBE [18] with 4096-dimensional binary codes is only 52% on CIFAR-10 dataset [20], while its supervised version, SuFBE, can achieve mAP at 68% using the same code length, which justifies the rationale for developing SuFBE. Following [19], we generate FBE to SuFBE by fitting the binary embedding problem into the framework of multi-class classification and avoid ruining the merits introduced by the adaptive Fastfood transform [16], [17]. As a result, SuFBE inherits all advantages of FBE such as computational efficiency and the restricted freedom, meanwhile can harness labels of training data to achieve better accuracy than FBE.

The involvement of structured matrices makes the optimization problem difficult, thus we adopt the variable-splitting [2], [21] and penalty techniques [19], [22] to develop an alternative optimization algorithm. We split the original optimization problem into several feasible sub-problems, and iteratively solve these sub-problems till convergence. In Section IV-D and Section VII-D, we further show that both FBE and SuFBE provably converge to local optima, respectively. We call our approaches as Fried Binary Embedding following Deep Fried Convnets [17] and Circulant Binary Embedding [12]. Extensive experiments show that our approach not only achieves competitive performance in compact-bit case, but also outperforms state-of-the-art methods in the long-bit scenario.

II. Related Work

A good review of binary embedding can be found in [23]. Here we focus on several closely related works.

Iterative Quantization (ITQ) [1] aims to find the hash codes such that the difference between the hash codes and the data items is minimized, by viewing each bit as the quantization value along the corresponding dimension. It consists of two steps: (1) dimension reduction via PCA; (2) find the hash codes as well as an optimal rotation. When the labels of training data are available, ITQ could be incorporated with Canonical Correlation Analysis (CCA) [25] as CCA-ITQ, which has shown good performance in image retrieval application [1].

Bilinear Projections (BP) [11] projects a data vector by two smaller matrices rather than a single large matrix, based on the assumption that the data vectors are formulated by reshaping matrices. This assumption of BP is valid for many traditional hand-crafted features like SIFT [8], GIST [9], VLAD [26], and Fisher Vectors [27], but is not true for learned features such as those learned by the deep neural network(DNN) [6], [7].

Circulant Binary Embedding (CBE) [12] imposes a circulant structure on the projection matrix for efficient matrix-vector multiplication. In virtue of fast Fourier transform, the computational cost of CBE is only $\mathcal{O}(d \log d)$, much less than $\mathcal{O}(d^2)$ of the dense projection method. It is worth noting that CBE shares similar idea with our approach, however, both BP and CBE achieve inferior accuracy to dense projection method (like ITQ [1]) using the same code length. Another similar work Fast Orthogonal Projection (KBE) [24] also has the same computation complexity $\mathcal{O}(d \log d)$.

Sparse Projection (SP) [2] and **Supervised Sparse Projection (SSP)** [3] introduce the sparsity regularizer to achieve efficiency in encoding. They also show that there exist many redundant parameters in dense projection matrix. However, these methods require the percentage of non-zero elements to be around 10% for competitive performance, which can be still suffering if both d and b are very large.

Semi-Supervised Hashing (SSH) [28] is one of the seminar work for utilizing training labels for hashing. SSH extends spectral hashing [5] into the semi-supervised case, in which some pairs of data items are labeled as belonging to the same semantic concept, some pairs are labeled as belonging to different semantic concepts. However, due to the relaxation of binary constraint, the performance of SSH [28] is not promising.

Supervised Discrete Hashing (SDH) [19] casts the supervised binary embedding to a multi-class classification problem.

SDH keeps the binary variables in the optimization procedure and minimizes the quantization loss between binary space and decimal space, thus, achieves better performance than SSH [28]. However, SDH [19] use a dense projection matrix for binary embedding, which makes it not scalable for large-scale high-dimensional problems.

III. Fried Binary Embedding

Let us discuss the unsupervised binary embedding first. Following [2], [23], we put dimension reduction and optimal rotation of ITQ [1] into one integrated objective:

$$\begin{aligned} &\min_{\mathrm{R,C}} \|\mathrm{RX}-\mathrm{C}\|_{\mathrm{F}}^2 \\ &\text{s.t. } \mathrm{R}^{\mathrm{T}}\mathrm{R}=\mathrm{I}. \end{aligned} \tag{1}$$

where $\mathrm{X}\in\mathbb{R}^{d\times n}$ is the dataset, C is a b-by-n matrix containing only 1 and -1. The matrix $\mathrm{R}\in\mathbb{R}^{b\times d}$ serves for both dimension reduction and rotation. ITQ [1] solves Equation 1 via alternative update. After finding R, ITQ can produce binary codes using the hash function below:

$$\mathrm{c}=\mathrm{sgn}(\mathrm{Rx}), \tag{2}$$

where $\mathbf{x}\in\mathbb{R}^d$ denotes a data vector, and $\mathrm{sgn}(\cdot)$ is the sign function, which outputs 1 for positive numbers and -1 otherwise. For simplicity of presentation, we first make two assumptions: (1) $\mathrm{R}\in\mathbb{R}^{d\times d}$ and (2) there exists some integer l such that $d=2^l$. We will advance our discussion towards the more generalized cases later.

Although ITQ [1] has shown promising results of binary embedding, its computational cost of the matrix-vector multiplication in Equation 2 is $\mathcal{O}(d^2)$, which limits its application to high-dimensional binary embedding. To reduce the cost of calculating $\mathrm{R}\mathbf{x}$, we decompose the projection matrix R using the adaptive Fastfood transform [17], *i.e.*,

$$\mathrm{R}=\mathrm{SHG}\Pi\mathrm{HB}. \tag{3}$$

Consequently, our hash function turns to be

$$\mathrm{c}=\mathrm{sgn}(\mathrm{SHG}\Pi\mathrm{HBx}). \tag{4}$$

In order to explain the reason of such a decomposition, we need to describe the component modules of the adaptive Fastfood transform. The adaptive Fastfood transform has three types of modules:

- S, G and B are diagonal matrices of tunable parameters. As a comparison, S, G and B in the original non-adaptive Fastfood formulation [16] are random matrices whose entries are computed once and kept unchanged. Since they are diagonal matrices, the computational and storage costs are only $\mathcal{O}(d)$.
  In this work, we define $\mathcal{D}$ to be the set of all diagonal matrices. For any *square matrix* $\mathrm{S}\in\mathbb{R}^{d\times d}$, we use its lower case letter $\mathbf{s}\in\mathbb{R}^d$ to represent the vector that consists of the diagonal elements of S, *i.e.*, $\mathbf{s}=\mathrm{diag}(\mathrm{S})$.
- $\Pi\in\{0,1\}^{d\times d}$ is a random permutation matrix, generated by sorting random numbers. It can be implemented as a lookup table, so the storage and computational costs are also $\mathcal{O}(d)$.
- H denotes the Walsh-Hadamard matrix, which is defined recursively as

$$\mathrm{H}_2:=\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix} \text{ and } \mathrm{H}_{2\mathrm{d}}:=\begin{bmatrix}\mathrm{H}_\mathrm{d} & \mathrm{H}_\mathrm{d}\\ \mathrm{H}_\mathrm{d} & -\mathrm{H}_\mathrm{d}\end{bmatrix}$$

  The Fast Hadamard Transform, a variant of Fast Fourier Transform, enables us to compute $\mathrm{H}_\mathrm{d}\mathbf{x}$ in $\mathcal{O}(d\log d)$ time. Note that H does not need to be explicitly stored.

As a result, the computational cost of using adaptive Fastfood transform to compute $\mathrm{R}\mathbf{x}$ is $\mathcal{O}(d\log d)$, while the storage cost of storing R is only $\mathcal{O}(d)$. These are substantial theoretical improvements over the $\mathcal{O}(d^2)$ costs of ordinary dense projection matrix.

In summary, we can attain the optimization objective of our FBE by putting Equation 1 and 3 together:

$$\begin{aligned} &\min_{\mathrm{S,G,B,C}} \|\mathrm{RX}-\mathrm{C}\|_{\mathrm{F}}^2 \\ &\text{s.t. } \mathrm{R}^{\mathrm{T}}\mathrm{R}=\mathrm{I}, \\ &\quad \mathrm{R}=\mathrm{SHG}\Pi\mathrm{HB}, \\ &\quad \mathrm{S},\mathrm{G},\mathrm{B}\in\mathcal{D}. \end{aligned} \tag{5}$$

IV. Optimization

Due to the involvement of structured matrices, Equation 5 is a more challenging problem compared with Equation 1. Updating any entry of $\mathrm{S},\mathrm{G},\mathrm{B}$ could cause the violation of the orthonormal constraint on R. To find a feasible solution, we adopt the variable-splitting and penalty techniques in optimization [2], [21], [22]. Specifically, we move the orthonormal constraint onto an auxiliary variable $\bar{\mathrm{R}}$ and meanwhile penalize the difference between $\bar{\mathrm{R}}\mathrm{X}$ and RX. As a result, we relax the problem in Equation 5 to the following form:

$$\begin{aligned} &\min_{\mathrm{S,G,B,C},\bar{\mathrm{R}}} \|\bar{\mathrm{R}}\mathrm{X}-\mathrm{C}\|_{\mathrm{F}}^2+\beta\|\bar{\mathrm{R}}\mathrm{X}-\mathrm{RX}\|_{\mathrm{F}}^2 \\ &\text{s.t. } \bar{\mathrm{R}}^T\bar{\mathrm{R}}=\mathrm{I}, \\ &\quad \mathrm{R}=\mathrm{SHG}\Pi\mathrm{HB}, \\ &\quad \mathrm{S},\mathrm{G},\mathrm{B}\in\mathcal{D}, \end{aligned} \tag{6}$$

where β is a penalty weight. Such a relaxation is similar to Half-Quadratic Splitting [22]. By introducing an auxiliary variable, the original problem can be separated into feasible sub-problems, and the solution to Equation 6 will converge to that of Equation 5 when $\beta\to\infty$ [22]. We solve Equation 6 in an alternating manner: update one variable with others fixed.

A. Update $\mathbf{C}$

This sub-problem is equivalent to $\min_{\mathrm{C}}\|\bar{\mathrm{R}}\mathrm{X}-\mathrm{C}\|_{\mathrm{F}}^2=\max_{\mathrm{C}}\sum_{i,j}(\bar{\mathrm{R}}\mathrm{X})_{ij}\mathrm{C}_{ij}$, where i,j are the indexes of matrix elements. Because $\mathrm{C}_{ij}\in\{-1,1\}$, this problem can be easily solved by $\mathrm{C}_{ij}=\mathrm{sgn}((\bar{\mathrm{R}}\mathrm{X})_{ij})$, or simply

$$\mathrm{C}=\mathrm{sgn}(\bar{\mathrm{R}}\mathrm{X}). \tag{7}$$

B. Update $\bar{\mathbf{R}}$

With R fixed, the two terms in Equation 6 are both quadratic on $\bar{\mathrm{R}}$. By some derivations, the problem Equation 6 becomes:

$$\begin{aligned} &\min_{\bar{\mathrm{R}}} \|\bar{\mathrm{R}}\mathrm{X}-\mathrm{Y}\|_{\mathrm{F}}^2 \\ &\text{s.t. } \bar{\mathrm{R}}^T\bar{R}=\mathrm{I}, \end{aligned} \tag{8}$$

where $\mathrm{Y} = (\mathrm{C} + \beta\mathrm{RX})/(1+\beta)$. This problem is known as the orthogonal procrustes problem [29], [30] and is recently widely studied in binary embedding [1], [2], [11].

According to [30], the procrustes problem is solvable only if $b \geq d$. For a fixed Y, Equation 8 is minimized as following: first, find the SVD of the matrix YX^{T} as $\mathrm{YX}^{\mathrm{T}} = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}$, then let

$$\bar{\mathrm{R}} = \mathrm{UV}^{\mathrm{T}}. \tag{9}$$

In case that $b < d$, $\bar{\mathrm{R}}^{\mathrm{T}}\bar{\mathrm{R}} = \mathrm{I}$ is no longer a valid constraint, because $\mathrm{rank}(\bar{\mathrm{R}}^{\mathrm{T}}\bar{\mathrm{R}}) \leq \min(b, d)$ while $\mathrm{rank}(\mathrm{I})$ is d. Extra efforts are made to handle the case of $b < d$ in Sparse Projection [2]. However, we do not face such a problem because we always have $b = d$ in the adaptive Fastfood transform, we would simply drop the redundant bits after optimization, as mentioned in Section IV-D.

C. Update $\mathbf{S}, \mathbf{G}, \mathbf{B}$

For S, G and B, we update one of them each time, with other variables fixed. However, we observe that all these three sub-problems can be regarded as unconstrained quadratic programming problem [31], and share the similar form of solutions. Thus, we unify the optimization of them into one section.

In case that $\bar{\mathrm{R}}$ and C are fixed, we could reformulate Equation 6 as:

$$\begin{aligned} &\min_{\mathrm{S,G,B}} \|\mathrm{SHG\Pi HBX} - \mathrm{Z}\|_{\mathrm{F}}^2 \\ &\text{s.t. } \mathrm{S}, \mathrm{G}, \mathrm{B} \in \mathcal{D}, \end{aligned} \tag{10}$$

where $\mathrm{Z} = \bar{\mathrm{R}}\mathrm{X}$. To show that Equation 10 can be split into three quadratic programming sub-problems, we first expand the objective in Equation 10 as below:

$$\begin{aligned} &\|\mathrm{SHG\Pi HBX} - \mathrm{Z}\|_{\mathrm{F}}^2 \\ &\quad = \|\mathrm{SHG\Pi HBX}\|_{\mathrm{F}}^2 - 2\mathrm{trace}(\mathrm{Z}^{\mathrm{T}}\mathrm{SHG\Pi HBX}) + Constant. \end{aligned} \tag{11}$$

$\|\mathrm{SHG\Pi HBX}\|_{\mathrm{F}}^2$ is a Frobenius norm, so it is always non-negative and has a quadratic form. Therefore, for anyone of $\mathbf{s} = \mathrm{diag}(\mathrm{S})$, $\mathbf{g} = \mathrm{diag}(\mathrm{G})$ or $\mathbf{b} = \mathrm{diag}(\mathrm{G})$, there must exist one corresponding positive-semidefinite matrix $\mathrm{Q_s}, \mathrm{Q_g}$ or $\mathrm{Q_b} \in \mathbb{R}^{d\times d}$ that satisfies

$$\|\mathrm{SHG\Pi HBX}\|_{\mathrm{F}}^2 = \frac{1}{2}\mathbf{s}^{\mathrm{T}}\mathrm{Q_s}\mathbf{s} = \frac{1}{2}\mathbf{g}^{\mathrm{T}}\mathrm{Q_g}\mathbf{g} = \frac{1}{2}\mathbf{b}^{\mathrm{T}}\mathrm{Q_b}\mathbf{b}. \tag{12}$$

As a result, all the three sub-problems can be regarded as quadratic programming problems. According to Equation 12, the three quadratic programming problems of S, G and B share the same form, so we will derive the general solution for them first, then explain how to address each of them specifically. Let us use W to denote anyone of S, G or B, then all these three sub-problems can be unified as the following form based on Equation 10 and 11:

$$\begin{aligned} &\min_{\mathrm{W}} \mathrm{trace}(\mathrm{E}^{\mathrm{T}}\mathrm{WD}^{\mathrm{T}}\mathrm{DWE}) - 2\mathrm{trace}(\mathrm{KW}) \\ &\text{s.t. } \mathrm{W} \in \mathcal{D}, \end{aligned} \tag{13}$$

where D, E and K are constant matrices whose specific values depend on W is which one of S, G and B. To rewrite Equation 13 into a quadratic programming form like Equation 12, we need to find $\mathrm{Q} \in \mathbb{R}^{d\times d}$ and $\mathbf{k} \in \mathbb{R}^d$ such that

$$\mathrm{trace}(\mathrm{E}^{\mathrm{T}}\mathrm{WD}^{\mathrm{T}}\mathrm{DWE}) - 2\mathrm{trace}(\mathrm{KW}) = \mathbf{w}^{\mathrm{T}}\mathrm{Q}\mathbf{w} - 2\mathbf{k}^{\mathrm{T}}\mathbf{w}, \tag{14}$$

where $\mathbf{w} = \mathrm{diag}(\mathrm{W})$. The optimal solution for the right-hand problem of Equation 14 can be easily found as:

$$\mathbf{w} = \mathrm{Q}^{-1}\mathbf{k}. \tag{15}$$

For anyone of S, G or B, we need to find out its corresponding Q and $\mathbf{k}$, then put them into Equation 15 to obtain the solution. To derive the formula of Q, let us consider the first term in the left hand of Equation 14,

$$\begin{aligned} &\mathrm{trace}(\mathrm{E}^{\mathrm{T}}\mathrm{WD}^{\mathrm{T}}\mathrm{DWE}) \\ &\quad = \|\mathrm{DWE}\|_{\mathrm{F}}^2 \\ &\quad = \sum_{i=1}^{d}\sum_{j=1}^{n}(\mathrm{DWE})_{ij}^2 \\ &\quad = \sum_{i=1}^{d}\sum_{j=1}^{n}\left(\sum_{s=1}^{d}\sum_{t=1}^{d}\mathrm{D}_{it}\mathrm{W}_{tt}\mathrm{E}_{tj}\mathrm{D}_{is}\mathrm{W}_{ss}\mathrm{E}_{sj}\right) \\ &\quad = \sum_{s=1}^{d}\sum_{t=1}^{d}\mathrm{W}_{tt}\left[\sum_{i=1}^{d}\sum_{j=1}^{n}\mathrm{D}_{it}\mathrm{E}_{tj}\mathrm{D}_{is}\mathrm{E}_{sj}\right]\mathrm{W}_{ss}. \end{aligned} \tag{16}$$

Because $\mathrm{trace}(\mathrm{E}^{\mathrm{T}}\mathrm{WD}^{\mathrm{T}}\mathrm{DWE}) = \mathbf{w}^{\mathrm{T}}\mathrm{Q}\mathbf{w}$, we can get

$$\begin{aligned} \mathrm{Q}_{st} &= \sum_{i=1}^{d}\sum_{j=1}^{n}\mathrm{D}_{it}\mathrm{E}_{tj}\mathrm{D}_{is}\mathrm{E}_{sj} \\ &= \sum_{i=1}^{d}\mathrm{D}_{it}\mathrm{D}_{is}\sum_{j=1}^{n}\mathrm{E}_{tj}\mathrm{E}_{sj} \\ &= \left(\mathrm{EE}^{\mathrm{T}}\right)_{st} \times \left(\mathrm{D}^{\mathrm{T}}\mathrm{D}\right)_{st}, \end{aligned} \tag{17}$$

or simply

$$\mathrm{Q} = \left(\mathrm{EE}^{\mathrm{T}}\right) \odot \left(\mathrm{D}^{\mathrm{T}}\mathrm{D}\right), \tag{18}$$

where $\odot$ stands for Hadamard product, *i.e.*, $\mathrm{C} = \mathrm{A} \odot \mathrm{B} \Leftrightarrow \mathrm{C}_{ij} = \mathrm{A}_{ij}\mathrm{B}_{ij}$. Computing $\mathbf{k}$ is relatively easy. Since $\mathrm{trace}(\mathrm{KW}) = \sum_{i=1}^{d}\mathrm{K}_{ii}\mathrm{W}_{ii} = \mathbf{k}^{\mathrm{T}}\mathbf{w}$, we have

$$\mathbf{k} = \mathrm{diag}(\mathrm{K}). \tag{19}$$

Substituting Equation 18 and 19 into Equation 15, we can obtain the update rule for $\mathbf{w}$:

$$\mathbf{w} = \left[\left(\mathrm{EE}^{\mathrm{T}}\right) \odot \left(\mathrm{D}^{\mathrm{T}}\mathrm{D}\right)\right]^{-1} \times \mathrm{diag}(\mathrm{K}). \tag{20}$$

Now we turn to the specific cases of S, G, B respectively.

Update S: In this case, we have the following D, E and K in Equation 13

$$\begin{aligned} \mathrm{D_S} &= \mathrm{I}, \\ \mathrm{E_S} &= \mathrm{HG\Pi HBX}, \\ \mathrm{K_S} &= \mathrm{HG\Pi HBXZ^T}, \end{aligned} \tag{21}$$

so the update rule for S according to Equation 20 is

$$\mathrm{diag(S)} = \left[\left(\mathrm{E_S E_S^T}\right) \odot \left(\mathrm{D_S}^{\mathrm{T}} \mathrm{D_S}\right)\right]^{-1} \times \mathrm{diag(K_S)}. \tag{22}$$

Update G: In this case, we have the following D, E and K in Equation 13

$$\begin{aligned} \mathrm{D_G} &= \mathrm{SH}, \\ \mathrm{E_G} &= \mathrm{\Pi HBX}, \\ \mathrm{K_G} &= \mathrm{\Pi HBXZ^T SH}, \end{aligned} \tag{23}$$

so the update rule for G according to Equation 20 is

$$\mathrm{diag(G)} = \left[\left(\mathrm{E_G E_G^T}\right) \odot \left(\mathrm{D_G}^{\mathrm{T}} \mathrm{D_G}\right)\right]^{-1} \times \mathrm{diag(K_G)}. \tag{24}$$

Update B: In this case, we have the following D, E and K in Equation 13

$$\begin{aligned} \mathrm{D_B} &= \mathrm{SHG\Pi H}, \\ \mathrm{E_B} &= \mathrm{X}, \\ \mathrm{K_B} &= \mathrm{XZ^T SHG\Pi H}, \end{aligned} \tag{25}$$

so the update rule for B according to Equation 20 is

$$\mathrm{diag(B)} = \left[\left(\mathrm{E_B E_B^T}\right) \odot \left(\mathrm{D_B}^{\mathrm{T}} \mathrm{D_B}\right)\right]^{-1} \times \mathrm{diag(K_B)}. \tag{26}$$

D. Implementation Details

In many practical applications, the input dimension d and code length b are usually power of 2. In case that $d = 2^l$ does not hold for any $l \in \mathbb{N}$, we can trivially pad the vectors with zeros until $d = 2^l$ holds. When b is not equal to d after zero-padding, we stack $\lceil b/d \rceil$ adaptive Fastfood transforms and attain the desired code length by simply dropping the extra $(\lceil b/d \rceil \times d - b)$ bits, where $\lceil d \rceil$ denotes the smallest integer greater than or equal to d. In doing so, the computational and storage costs of our FBE become $\mathcal{O}(b \log d)$ and $\mathcal{O}(b)$, respectively.[1]

To optimize our objective function in Equation 6, we iteratively solve the 5 sub-problems as described in Section IV-A–IV-C. We initialize $\mathrm{S} = \frac{1}{d^2}\mathrm{I}, \mathrm{G} = \mathrm{I}, \mathrm{B} = \mathrm{I}, \bar{\mathrm{R}} = \mathrm{R}$ to satisfy the orthogonal constraint. The training data is subtracted with its mean prior to learning. We summarize our proposed FBE in Algorithm 1.

Our problem formulation has only one hyperparameter β as shown in Equation 6. To tune this hyperparameter, we should in principle start from a small β and gradually increase it to infinity [22]. But in our experiments, we find that simply using a fixed β leads to comparable accuracy, and the accuracy is very insensitive to the choice of fixed β (we tried from 0.1 to 100). So we simply fix $\beta = 1$ for all experiments in this paper. The experiments show such a setting of β works well for features of various dimensions on all datasets.

[1]This strategy actually also works for Circulant Binary Embedding [12], which is previously considered unable to produce binary codes that are longer than original feature vector [2].

Algorithm 1 Fried Binary Embedding

Input: X
Output: S, G, B, Π

Subtract X by its mean
Initialize Π as random permutation matrix
$\mathrm{S} \leftarrow \frac{1}{d^2}\mathrm{I}, \mathrm{B} \leftarrow \mathrm{I}, \mathrm{G} \leftarrow \mathrm{I}, \mathrm{R} \leftarrow \mathrm{SHG\Pi HB}, \bar{\mathrm{R}} \leftarrow \mathrm{R}$
repeat
  $\mathrm{C} \leftarrow \mathrm{sgn}(\bar{\mathrm{R}}\mathrm{X})$
  $\mathrm{Y} \leftarrow (\mathrm{C} + \beta \mathrm{RX})/(1+\beta)$
  Perform SVD on $\mathrm{YX^T}$ such that $\mathrm{YX^T} = \mathrm{U\Sigma V^T}$
  $\bar{\mathrm{R}} \leftarrow \mathrm{UV^T}$
  $\mathrm{D_S} \leftarrow \mathrm{I}$, $\mathrm{E_S} \leftarrow \mathrm{HG\Pi HBX}$, $\mathrm{K_S} \leftarrow \mathrm{HG\Pi HBXZ^T}$
  $\mathrm{diag(S)} \leftarrow \left[\left(\mathrm{E_S E_S^T}\right) \odot \left(\mathrm{D_S}^{\mathrm{T}} \mathrm{D_S}\right)\right]^{-1} \times \mathrm{diag(K_S)}$
  $\mathrm{D_G} \leftarrow \mathrm{SH}$, $\mathrm{E_G} \leftarrow \mathrm{\Pi HBX}$, $\mathrm{K_G} \leftarrow \mathrm{\Pi HBXZ^T SH}$
  $\mathrm{diag(G)} \leftarrow \left[\left(\mathrm{E_G E_G^T}\right) \odot \left(\mathrm{D_G}^{\mathrm{T}} \mathrm{D_G}\right)\right]^{-1} \times \mathrm{diag(K_G)}$
  $\mathrm{D_B} \leftarrow \mathrm{SHG\Pi H}$, $\mathrm{E_B} \leftarrow \mathrm{X}$, $\mathrm{K_B} \leftarrow \mathrm{XZ^T SHG\Pi H}$
  $\mathrm{diag(B)} \leftarrow \left[\left(\mathrm{E_B E_B^T}\right) \odot \left(\mathrm{D_B}^{\mathrm{T}} \mathrm{D_B}\right)\right]^{-1} \times \mathrm{diag(K_B)}$
until convergence
return S, G, B, Π

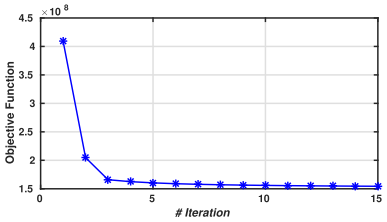
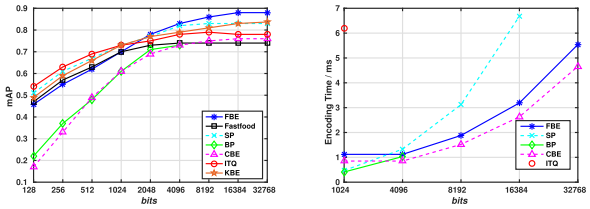


Fig. 2. Convergence of Algorithm 1. The vertical axis represents the objective function value of Equation 6 and the horizontal axis corresponds to the number of iterations at Algorithm 1. The optimization of R is obtained on the training set of DNN-4096 [2].

Since all 5 sub-problems (4 of them are convex) we tackle have optimal solutions individually, our Algorithm 1 should converge fast. As shown in Figure 2, the objective function value at each iteration in the Algorithm 1 always decreases. Considering that the objective function value is also lower-bounded (not smaller than 0), it validates the convergence of our algorithm and demonstrates that it only takes a few iterations to converge. Such a fast learning procedure will benefit learning R on large datasets.

V. Evaluate Fried Binary Embedding

To evaluate proposed Fried Binary Embedding (FBE), we conduct experiments on three tasks: approximate nearest neighbor (ANN) search, image retrieval, and image classification, following the experiments setting in [2]. For each task, we compare our method FBE with the original Fastfood transform [16], as well as the several state-of-the-art methods for unsupervised high-dimensional visual feature embedding, including Iterative Quantization (ITQ) [1], Bilinear Projection (BP) [11], Circulant Binary Embedding (CBE) [12], Fast Orthogonal Projection (KBE) [24] and Sparse Projection (SP) [2]. For the original Fastfood transform [16] where S, G and B are randomly generated instead of being optimized, we report the best performance achieved by varying the



(a) mAP *vs.* bits (b) encoding time *vs.* bits

Fig. 3. Comparison results on DNN-4096 dataset. (a) The change of mAP with respect to bits. (b) The change of encoding time with respect to bits. Here Fastfood transform is omitted because it takes the same encoding time as FBE. For clarify, we only show the encoding time of ITQ at 1024 bits, and the encoding time of SP up to 16384 bits. Note that BP is unavailable for the longer codes ($b > d$).

standard deviation of the random Gaussian matrix over the set $\{0.001, 0.005, 0.01, 0.05\}$. For the optimized variant FBE, we learn these matrices by iterative optimization as described in Section IV. We use the implementations of ITQ, BP, CBE and SP that are released by their authors.

All experiments are conducted using Matlab, while the evaluation of encoding time is implemented in C++ with a single thread. The server we use is equipped with Intel Xeon CPUs E5-2630 (2.30GHz) and 96 GB memory.

A. Approximate Nearest Neighbor Search

1) Experiments on DNN features: Recent research advances have demonstrated the advantage of deep learning features as image representations [32], [33]. We first conduct experiments on such features. We use the pre-trained AlexNet [6] provided by Caffe [34] to extract deep learning features for one million images in MIRFLICKR-1M dataset [35], [36]. AlexNet contains five convolutional layers and two fully-connected (fc) layers, followed by a softmax classifier. Using this network, we extract 4096-dimensional outputs of the second fc layer as image features. Each image is resized to keep the same aspect ratio but smaller side to be 256, and the center 224×224 region is used to compute features. We refer to this dataset as DNN-4096. Extra 1,000 random samples are used as queries. Note that each 4096-dimensional raw feature (real number) requires a storage of 16,384 bytes (131,072 bits).

Following the protocol in [37] and [38], we measure the search quality using mean Average Precision (mAP), *i.e.*, the mean area under the precision-recall curve. Given a query, we perform Hamming ranking, *i.e.*, samples in the dataset are ranked according to their Hamming distances to the query, based on their binary codes. The 50 nearest neighbors of each query in the dataset using original features are defined as the true positive samples, which are the ground truths for us to evaluate the mAP.

In Figure 3a, we show how mAP changes with various code length b. The proposed FBE achieves competitive mAP at the short-bit scenario, and significantly outperforms other state-of-the-art methods at the long-bit scenario, *i.e.*, bit lengths comparable to or longer than feature dimension. For example, with 2048 bits or more, our proposed method performs the best of embedding the 4096-dimensional CNN features, when compared with SP [2], KBE [24], BP [11], CBE [12] and ITQ [1]. Interestingly, although the performance of our proposed method is not better than that of Fastfood transform [16] below 1024 bits, it significantly outperforms Fastfood transform [16] when above 1024 bits. This verifies that our optimization of S, G and B to Equation 5 does improve the performance compared with using random matrices for S, G and B as Fastfood transform [16] does.

TABLE I

THE NUMBER OF TUNABLE PARAMETERS WHEN ENCODING DNN-4096 FEATURE TO BINARY CODES OF VARYING LENGTHS. NOTE THAT BP IS NOT APPLICABLE TO GENERATE BINARY CODES THAT ARE LONGER THAN THE ORIGINAL FEATURE VECTOR

bits	2048	4096	8192	16384	32768
ITQ [1]	8.3×10^6	1.6×10^7	3.2×10^7	6.4×10^7	1.2×10^8
SP [2]	8.3×10^5	1.6×10^6	3.2×10^6	6.4×10^6	1.2×10^7
BP [11]	6144	8192	-	-	-
CBE [12]	8192	8192	16384	32768	65536
KBE [24]	88	96	104	112	120
FBE(ours)	12288	12288	24576	49152	98304

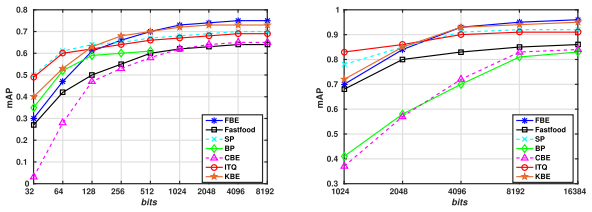
As shown in Figure 3b, the encoding time for computing the binary codes does not linearly increase to b. The proposed FBE takes less encoding time compared with SP and KBE, but not as fast as BP and CBE. Although CBE and BP can achieve superior speedup ratios to FBE, they have relatively low performance, and BP is unavailable for producing the longer codes ($b > d$).

We compare the number of parameters of our proposed algorithm and the baselines in Table I. Besides the advantage of less encoding time, the proposed FBE requires much fewer parameters to build the projection matrix R, which reduces not only the cost of memory but also the risk of overfitting. For example, when encoding a 4096-dimensional feature vector into 4096-bit binary codes, our FBE has only 12,288 tunable parameters, which are only 1% of Sparse Projection [2] and 0.1% of ITQ [1]. Although BP [11], CBE [12] and KBE [24] require even fewer parameters than ours when producing binary codes of the same length, these methods are inferior to the proposed FBE in terms of performance, as shown in Figure 3a and Figure 5.

2) Experiments on traditional features: To validate the generality of our proposed FBE, besides using deep learning features, we also evaluate our method on two datasets of traditional features. The first dataset is GIST-960 [39], which contains one million 960-dimensional GIST features [9] and 10,000 queries. The second dataset is VLAD-25600 [2]. The VLAD features [26] are extracted from 100,000 images randomly sampled from the INRIA image set [39]. The 25600-dimensional VLAD features are generated by encoding 128-dimensional SIFT vectors [8] to a 200-center codebook. An extra random subset of 1000 samples are used as queries in this dataset.

Figure 5 shows the approximate nearest neighbor search results on these datasets, using the same protocol as in Figure 3a. For each query, we retrieve the top-50 nearest neighbors based on the hamming distance of the binary codes, and compare it with the ground truths in the original feature space. The proposed FBE still outperforms state-of-the-art methods in long-bit case, meanwhile encodes high-dimensional visual features faster than ITQ, SP and KBE.

Fig. 4. Visualization of Holidays+1M dataset retrieval results. Red border means false positive. (a) FBE with different bits. (b) Different hashing methods with 8,192 bits.



(a) GIST-960 (b) VLAD-25600

Fig. 5. Comparison of traditional features. (a) The change of mAP with respect to bits on GIST-960 dataset. (b) The change of mAP with respect to bits on VLAD-25600 dataset.

The experiments on GIST and VLAD features show that our method is also applicable to traditional features, demonstrating the potential of the adaptive Fastfood transform to high-dimensional visual features again.

B. Image Retrieval

In the work of Krizhevsky *et al.* [6], the responses of the second fc layer of CNN are used as image features for image retrieval. We evaluate the performance of binary embedding for this task, on the "Holidays + MIRFlickr-1M" dataset [39]. This dataset contains 1,419 images in 500 different scenes, with extra one million MIRFlickr-1M images as distracters. Another 500 query images are provided along with their ground truth neighbors under the same scene category. We represent each image by a 4096-dimensional deep feature as introduced in the above experiments.

Following previous practices [2], [5], [11], [26], we treat image retrieval as an ANN search problem of the encoded features, while the ground truth neighbors are defined by scene labels. Given a query image, we perform Hamming ranking and evaluate mAP using the semantic ground truth.

TABLE II
IMAGE RETRIEVAL PERFORMANCE ON HOLIDAYS+1M.

		mAP	Encoding time (ms)
raw deep features (4096-d)		49.5%	-
1024 bits	BP [11]	44.5%	0.41
	CBE [12]	44.3%	0.85
	SP [2]	44.9%	0.47
	KBE [24]	45.0%	0.92
	Fastfood [16]	44.7%	1.12
	FBE(ours)	**45.6%**	
4096 bits	BP [11]	46.4%	1.03
	CBE [12]	46.6%	0.85
	SP [2]	47.1%	1.33
	KBE [24]	47.2%	1.36
	Fastfood [16]	46.2%	1.12
	FBE(ours)	**47.2%**	
8192 bits	CBE [12]	47.8%	1.52
	SP [2]	48.5%	3.12
	KBE [24]	48.8%	2.18
	Fastfood [16]	47.6%	1.88
	FBE(ours)	**49.3%**	

Table II shows the results on the Holidays+1M dataset. As a baseline of using the raw features, the mAP of 4096-dimensional deep learning features is 49.5%. To maintain such a performance, we compare our method with BP [11], CBE [12], SP [2], KBE [24] and original Fastfood transform [16] using 1024, 4096 and 8192 bits. Our method can lead to the best mAP in all bit lengths. In case of 8192 bits, our method has almost no degradation (49.3% mAP) compared with the use of the raw deep learning features. However, we do not observe better performance when using 16384 bits.

Figure 4 illustrates the top 10 retrieval results of a selected query. From Figure 4a we could observe that the retrieval quality is tending to be better with more binary bits used, which validate the rationale between our high-dimensional binary embedding. As shown in Figure 4b, our FBE achieves the best retrieval performance among all compared methods.

C. Image Classification

We further evaluate the binary codes as compact features for image classification on CIFAR-10 dataset [20], using top-1 accuracy as the metric. As a baseline, we extract the 4096-dimensional responses of second fc layer in AlexNet [6] as image features. We first fine-tune the pretrained model provided by Caffe [34] on the training set of CIFAR-10, then we use the fine-tuned model to generate features for both training images and testing images. We then learn the hashing parameters on the features of CIFAR-10 [20] training set.

Following [2], we use one-vs-rest linear SVM as the classifier. We observe that one-vs-rest linear SVM achieves 82.6% classification accuracy, which is higher than that from the softmax layer (78.9%). We compare our method with BP [11], CBE [12], SP [2], KBE [24] and original Fastfood transform [16] using 1024, 4096, 8192 and 16384 bits. We do not see significant improvement in the performance when further increasing the bit length.

Table III lists the comparison results. The proposed FBE performs better than BP, CBE, SP, and KBE with the same number of bits. It is worth noting that even the number of bits is more than the input dimension 4096, these representations are still more compact than the original features. For example, 16,384 bits require only the 1/8 storage cost of the raw 4096-dimensional feature of real numbers.

TABLE III
CLASSIFICATION ACCURACY ON CIFAR-10 DATASET

		Classification accuracy	Encoding time (ms)
raw deep feature (4096-d)		82.6%	-
1024 bits	BP [11]	76.6%	0.41
	CBE [12]	76.3%	0.85
	SP [2]	77.8%	0.47
	KBE [24]	78.3%	0.92
	Fastfood [16]	77.4%	1.12
	FBE(ours)	**79.7%**	
4096 bits	BP [11]	77.5%	1.03
	CBE [12]	77.4%	0.85
	SP [2]	78.6%	1.33
	KBE [24]	79.3%	1.36
	Fastfood [16]	78.3%	1.12
	FBE(ours)	**80.7%**	
8192 bits	CBE [12]	78.1%	1.52
	SP [2]	79.5%	3.12
	KBE [24]	80.5%	2.18
	Fastfood [16]	79.0%	1.88
	FBE(ours)	**81.6%**	
16384 bits	CBE [12]	78.6%	2.63
	SP [2]	80.2%	6.68
	KBE [24]	81.0%	3.55
	Fastfood [16]	79.3%	3.19
	FBE(ours)	**82.4%**	

D. Discussion

In the above experiments (Figure 3a,5 and Table II,III), we observe that the binary code length b required to achieve graceful degradation (compared with no encoding) is usually around $b \sim \mathcal{O}(d)$, which justifies the rationality of using long binary codes for high-dimensional data. Short binary codes have considerable degradation of accuracy, and may impact the quality of real-world usage, thus in practice, it is desired to have a feasible and accurate solution to high-dimensional binary embedding.

VI. SUPERVISED FRIED BINARY EMBEDDING

As shown in [1], [19], and [3], supervised binary embedding can significantly outperform their unsupervised counterparts if the labels of training data are utilized. In this section, we propose Supervised Fried Binary Embedding (SuFBE) for supervised hashing problem. We assume we have a label matrix $\mathrm{Y} = \{\mathrm{Y}_i\}_{i=1}^n \in \{0,1\}^{t\times n}$ available, where $\mathrm{Y}_{ki} = 1$ if the i-th training sample X_i belongs to the k-th class and 0 otherwise. To take advantage of such label information, we fit the supervised hashing problem into the framework of linear classification. As has been shown in [19] and [40], good binary codes are suitable for classification too.

Following [3], [19], we adopt the following multi-class classification formulation

$$\begin{aligned} \min_{\mathrm{P,S,G,B,C}} \ & \|\mathrm{Y} - \mathrm{P}^{\mathrm{T}}\mathrm{C}\|_{\mathrm{F}}^2 + \lambda\|\mathrm{P}\|_{\mathrm{F}}^2 \\ \text{s.t. } & \mathrm{C} = \mathrm{sgn}(\mathrm{RX}), \\ & \mathrm{R} = \mathrm{SHG}\Pi\mathrm{HB}, \\ & \mathrm{S}, \mathrm{G}, \mathrm{B} \in \mathcal{D}. \end{aligned} \tag{27}$$

where the matrix $\mathrm{P} \in \mathbb{R}^{b\times t}$ serves as the classifier. The first term in 27 measures the classification accuracy by computing the difference between the labels and predictions, while the second term is a regulation term weighted by λ.

The problem 27 is in general NP hard to solve due to the discrete variable C. One can always obtain an approximate solution by simply relaxing the binary constraint to be continuous $\mathrm{C} = \mathrm{RX}$. With this relaxation, the continuous embeddings C are first learned, which are then thresholded to be binary codes. This relaxation approach has been adopted in many existing hashing algorithms, such as Spectral Hashing [5], SSH [28], AGH [41], *etc*. Although relaxation to continuous embeddings usually makes the original problem much easier to solve, clearly, it is only sub-optimal.

In order to attain better binary codes, here we solve it with the binary constraint on C kept in the optimization procedure. We rewrite problem 27 as

$$\begin{aligned}\min_{\mathrm{P,S,G,B,C}}\ & \|\mathrm{Y}-\mathrm{P}^{\mathrm{T}}\mathrm{C}\|_{\mathrm{F}}^2+\lambda\|\mathrm{P}\|_{\mathrm{F}}^2+\nu\|\mathrm{RX}-\mathrm{C}\|_{\mathrm{F}}^2\\ \text{s.t. }& \mathrm{R}=\mathrm{SHG}\Pi\mathrm{HB},\\ & \mathrm{S},\mathrm{G},\mathrm{B}\in\mathcal{D}.\end{aligned}\tag{28}$$

where the last term measures the quantization loss of the binary embedding. In theory, problem 28 becomes arbitrarily close to 27 with ν large enough. In practice, small differences between C and RX are acceptable as shown in Section IV-D. Although the above joint optimization problem is still highly non-convex and difficult to solve, however, it is tractable to solve the problem with respect to one variable while keeping other two variables fixed. Therefore, we again adopt the iterative optimization methods to efficiently find a local optimum of problem 28.

VII. Optimization

In this section, we describe our iterative solution to the problem 28. We update each variable with others fixed.

A. Update P

With other variables fixed, solving P can be considered as a regularized least squares problem, which has a closed-form solution:

$$\mathrm{P}=(\mathrm{CC}^{\mathrm{T}}+\lambda\mathrm{I})^{-1}\mathrm{CY}^{\mathrm{T}}\tag{29}$$

B. Update C

It is challenging to solve C due to the binary constraints. With all variables except C fixed, we write problem 28 as:

$$\begin{aligned}\min_{\mathrm{C}}\ & \|\mathrm{Y}-\mathrm{P}^{\mathrm{T}}\mathrm{C}\|_{\mathrm{F}}^2+\nu\|\mathrm{C}-\mathrm{RX}\|_{\mathrm{F}}^2\\ \text{s.t. }& \mathrm{C}\in\{-1,1\}^{b\times n}.\end{aligned}\tag{30}$$

Although the above problem is NP hard, a closed-form solution for one row of C can be achieved by fixing all the other rows. In other words, we can iteratively learn one bit at a time. To see this, let us expand 30 as:

$$\begin{aligned}&\|\mathrm{Y}-\mathrm{P}^{\mathrm{T}}\mathrm{C}\|_{\mathrm{F}}^2+\nu\|\mathrm{RX}-\mathrm{C}\|_{\mathrm{F}}^2\\ &\quad=\|\mathrm{Y}\|_{\mathrm{F}}^2-2\mathrm{trace}(\mathrm{Y}^{\mathrm{T}}\mathrm{P}^{\mathrm{T}}\mathrm{C})+\|\mathrm{P}^{\mathrm{T}}\mathrm{C}\|_{\mathrm{F}}^2\\ &\qquad+\nu(\|\mathrm{C}\|_{\mathrm{F}}^2-2\mathrm{trace}(\mathrm{C}^{\mathrm{T}}\mathrm{RX})+\|\mathrm{RX}\|_{\mathrm{F}}^2)\end{aligned}\tag{31}$$

By omitting the irrelevant terms to C, we could equivalently write problem 28 as:

$$\begin{aligned}\min_{\mathrm{C}}\ & \|\mathrm{P}^{\mathrm{T}}\mathrm{C}\|_{\mathrm{F}}^2-2\mathrm{trace}(\mathrm{C}^{\mathrm{T}}\mathrm{Q})\\ \text{s.t. }& \mathrm{C}\in\{-1,1\}^{b\times n}.\end{aligned}\tag{32}$$

where $\mathrm{Q}=\mathrm{PY}+\nu\mathrm{RX}$. One bit of binary codes corresponds to one row in the matrix $\mathrm{C}\in\{-1,1\}^{b\times n}$, and we learn C bit by bit. Let c^{T} represent for the k-th row of C, and C' for the matrix of C excluding c, then $c\in\{-1,1\}^n$ is one bit for all n training samples. Similarly, let q^{T} represent for the k-th row of Q, p^{T} for the k-th row of P, and P' for the matrix of P excluding p. Then we have

$$\begin{aligned}\|\mathrm{P}^{\mathrm{T}}\mathrm{C}\|_{\mathrm{F}}^2&=\mathrm{trace}(\mathrm{C}^{\mathrm{T}}\mathrm{PP}^{\mathrm{T}}\mathrm{C})\\ &=Constant+\|\mathrm{cp}^{\mathrm{T}}\|_2^2+2\mathrm{p}^{\mathrm{T}}\mathrm{P}'^{\mathrm{T}}\mathrm{C}'\mathrm{c}\\ &=Constant+2\mathrm{p}^{\mathrm{T}}\mathrm{P}'^{\mathrm{T}}\mathrm{C}'\mathrm{c}\end{aligned}\tag{33}$$

where $\|\mathrm{cp}^{\mathrm{T}}\|_2^2=\mathrm{trace}(\mathrm{pc}^{\mathrm{T}}\mathrm{cp}^{\mathrm{T}})=n\mathrm{p}^{\mathrm{T}}\mathrm{p}=Constant$. Similarly, we have

$$\|\mathrm{C}^{\mathrm{T}}\mathrm{Q}\|_{\mathrm{F}}^2=Constant+\mathrm{q}^{\mathrm{T}}\mathrm{c}\tag{34}$$

Putting Equation 33, 34 and 32 together, we have the following problem with respect to c:

$$\begin{aligned}\min_{\mathrm{c}}\ & (\mathrm{p}^{\mathrm{T}}\mathrm{P}'^{\mathrm{T}}\mathrm{C}'-\mathrm{q}^{\mathrm{T}})\mathrm{c}\\ \text{s.t. }& \mathrm{c}\in\{-1,1\}^n.\end{aligned}\tag{35}$$

whose optimal solution could be obtained by:

$$\mathrm{c}=\mathrm{sgn}(\mathrm{q}-\mathrm{C}'^{\mathrm{T}}\mathrm{P}'\mathrm{p})\tag{36}$$

Clearly, each bit c is computed based on the pre-learned $b-1$ bits C'. In our experiments, the whole b bits can be iteratively learned in rb times by using Equation 36, where usually $r=2\sim5$.

C. Update S,G,B

With P and C fixed, the problem 28 could be rewrite as:

$$\begin{aligned}\min_{\mathrm{S,G,B}}\ & \|\mathrm{SHG}\Pi\mathrm{HBX}-\mathrm{C}\|_{\mathrm{F}}^2\\ \text{s.t. }& \mathrm{S},\mathrm{G},\mathrm{B}\in\mathcal{D}.\end{aligned}\tag{37}$$

which has the same form as problem 10, hence we adopt the optimization method in Section IV-C to solve the above problem 37.

Update S: In this case, we have the following D, E and K in Equation 13

$$\begin{aligned}\mathrm{D_S}&=\mathrm{I},\\ \mathrm{E_S}&=\mathrm{HG}\Pi\mathrm{HBX},\\ \mathrm{K_S}&=\mathrm{HG}\Pi\mathrm{HBXC}^{\mathrm{T}},\end{aligned}\tag{38}$$

so the update rule for S according to Equation 20 is

$$\mathrm{diag(S)}=\left[\left(\mathrm{E_S}\mathrm{E_S}^{\mathrm{T}}\right)\odot\left(\mathrm{D_S}^{\mathrm{T}}\mathrm{D_S}\right)\right]^{-1}\times\mathrm{diag}(\mathrm{K_S}).\tag{39}$$

Algorithm 2 Supervised Fried Binary Embedding

Input: X, X
Output: $\mathrm{S}, \mathrm{G}, \mathrm{B}, \Pi$

Initialize Π as random permutation matrix
$\mathrm{S} \leftarrow \frac{1}{d^2}\mathrm{I}, \mathrm{B} \leftarrow \mathrm{I}, \mathrm{G} \leftarrow \mathrm{I}, \mathrm{R} \leftarrow \mathrm{SHG}\Pi\mathrm{HB}, \mathrm{C} \leftarrow \mathrm{sgn}(\bar{\mathrm{R}}\mathrm{X})$
repeat
  $\mathrm{P} = (\mathrm{CC}^{\mathrm{T}} + \lambda\mathrm{I})^{-1}\mathrm{CY}^{\mathrm{T}}$
  $\mathrm{Q} = \mathrm{PY} + \nu\mathrm{X}$
  repeat
    for $k \in \{1, \ldots, b\}$ **do**
      $\mathrm{c}^{\mathrm{T}} \leftarrow$ the k-th row of C
      $\mathrm{C}' \leftarrow$ the matrix C excluding c
      $\mathrm{q}^{\mathrm{T}} \leftarrow$ the k-th row of Q
      $\mathrm{p}^{\mathrm{T}} \leftarrow$ the k-th row of P
      $\mathrm{P}' \leftarrow$ the matrix P excluding p
      $\mathrm{c} = \mathrm{sgn}(\mathrm{q} - \mathrm{C}'^{\mathrm{T}}\mathrm{P}'\mathrm{p})$
    end for
  until reaching maximum iterations r
  $\mathrm{D_S} \leftarrow \mathrm{I}, \mathrm{E_S} \leftarrow \mathrm{HG}\Pi\mathrm{HBX}, \mathrm{K_S} \leftarrow \mathrm{HG}\Pi\mathrm{HBXC}^{\mathrm{T}}$
  $\mathrm{diag(S)} \leftarrow \left[\left(\mathrm{E_S}\mathrm{E_S}^{\mathrm{T}}\right) \odot \left(\mathrm{D_S}^{\mathrm{T}}\mathrm{D_S}\right)\right]^{-1} \times \mathrm{diag}(\mathrm{K_S})$
  $\mathrm{D_G} \leftarrow \mathrm{SH}, \mathrm{E_G} \leftarrow \Pi\mathrm{HBX}, \mathrm{K_G} \leftarrow \Pi\mathrm{HBXC}^{\mathrm{T}}\mathrm{SH}$
  $\mathrm{diag(G)} \leftarrow \left[\left(\mathrm{E_G}\mathrm{E_G}^{\mathrm{T}}\right) \odot \left(\mathrm{D_G}^{\mathrm{T}}\mathrm{D_G}\right)\right]^{-1} \times \mathrm{diag}(\mathrm{K_G})$
  $\mathrm{D_B} \leftarrow \mathrm{SHG}\Pi\mathrm{H}, \mathrm{E_B} \leftarrow \mathrm{X}, \mathrm{K_B} \leftarrow \mathrm{XC}^{\mathrm{T}}\mathrm{SHG}\Pi\mathrm{H}$
  $\mathrm{diag(B)} \leftarrow \left[\left(\mathrm{E_B}\mathrm{E_B}^{\mathrm{T}}\right) \odot \left(\mathrm{D_B}^{\mathrm{T}}\mathrm{D_B}\right)\right]^{-1} \times \mathrm{diag}(\mathrm{K_B})$
until convergence
return $\mathrm{S}, \mathrm{G}, \mathrm{B}, \Pi$

Update G: In this case, we have the following D, E and K in Equation 13

$$\begin{aligned} \mathrm{D_G} &= \mathrm{SH}, \\ \mathrm{E_G} &= \Pi\mathrm{HBX}, \\ \mathrm{K_G} &= \Pi\mathrm{HBXC}^{\mathrm{T}}\mathrm{SH}, \end{aligned} \tag{40}$$

so the update rule for G according to Equation 20 is

$$\mathrm{diag(G)} = \left[\left(\mathrm{E_G}\mathrm{E_G}^{\mathrm{T}}\right) \odot \left(\mathrm{D_G}^{\mathrm{T}}\mathrm{D_G}\right)\right]^{-1} \times \mathrm{diag}(\mathrm{K_G}). \tag{41}$$

Update B: In this case, we have the following D, E and K in Equation 13

$$\begin{aligned} \mathrm{D_B} &= \mathrm{SHG}\Pi\mathrm{H}, \\ \mathrm{E_B} &= \mathrm{X}, \\ \mathrm{K_B} &= \mathrm{XC}^{\mathrm{T}}\mathrm{SHG}\Pi\mathrm{H}, \end{aligned} \tag{42}$$

so the update rule for B according to Equation 20 is

$$\mathrm{diag(B)} = \left[\left(\mathrm{E_B}\mathrm{E_B}^{\mathrm{T}}\right) \odot \left(\mathrm{D_B}^{\mathrm{T}}\mathrm{D_B}\right)\right]^{-1} \times \mathrm{diag}(\mathrm{K_B}). \tag{43}$$

D. Implementation Details

We use the strategy in Section IV-D to deal with the case that: (1) the input dimension $d = 2^l$ does not hold for any $l \in \mathbb{N}$; (2) the code length b is not equal to d after zero-padding. Therefore, the computational and storage costs of our SuFBE are $\mathcal{O}(b \log d)$ and $\mathcal{O}(b)$, respectively.

To optimize our objective function in Equation 28, we iteratively solve the 5 sub-problems as described in Section VII-A–VII-C. We initialize $\mathrm{S} = \frac{1}{\mathrm{d}^2}\mathrm{I}, \mathrm{G} = \mathrm{I}, \mathrm{B} = \mathrm{I}$. We empirically set $\lambda = 1$ and $\nu = 10^{-5}$; the maximum iteration number r is 5. The entire flow of the proposed SuFBE is summarized in Algorithm 2. Figure 6 shows that the objective function value at each iteration in the Algorithm 2 always decreases. Considering that the objective function value is also lower-bounded (not smaller than 0), it validates the convergence of our algorithm and demonstrates that it only takes around 15 iterations to converge.

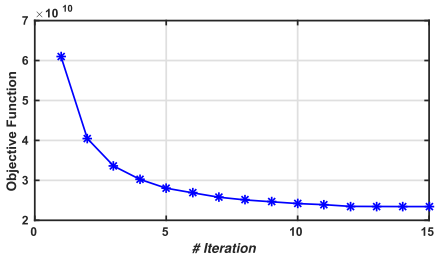


Fig. 6. Convergence of our algorithm. The vertical axis represents the objective function value of Equation 28 and the horizontal axis corresponds to the number of iterations at Algorithm 2. The optimization of R is obtained on the training set of CIFAR-10 [20].

VIII. Evaluate Supervised Fried Binary Embedding

In this section, we evaluate the proposed Supervised Fried Binary Embedding (SuFBE) for the image retrieval task on two benchmark datasets. For each dataset, we compare the SuFBE with the unsupervised counterpart FBE, as well as 4 state-of-the-art supervised hashing approaches. The comparisons with FBE are mainly conducted to show the advantages of utilizing labels of training data, while the comparisons with state-of-the-art hashing methods demonstrate both accuracy and efficiency advantage of the proposed SuFBE. The compared supervised hashing methods are CCA-ITQ [1], SSH [28], SDH [19] and SSP [3]. Our method can be extended to a non-linear embedding by RBF kernel in the same way as SDH [19]. For the fair comparisons with other non-kernel methods like SSH [28] and SSP [3], we evaluate all methods without kernel embedding. However, our approach should show more advantages in case of kernelized hashing, because the feature vector after kernel mapping is usually of higher dimensionality than the original feature.

We perform image retrieval experiments on two datasets: CIFAR-10 dataset [20] and NUS-WIDE dataset [42]. We evaluate the retrieval quality and encoding speed for all methods. Retrieval accuracy is measured by the mean average precision (mAP). With labeled data, we are interested in preserving semantic similarity. The retrieval ground truth for computing the mean average precision consists of database examples sharing the same semantic category label as the query. We compute the encoding time as the processor time required for the matrix-vector multiplication $\mathrm{c} = \mathrm{sgn}(\mathrm{Rx})$ (Equation 2). All experiments are conducted using Matlab, while the evaluation of encoding time is implemented in C++ with a single thread. The server we use is equipped with Intel Xeon CPUs E5-2630 (2.30GHz) and 96 GB memory.

A. Evaluation on CIFAR-10 dataset

We first evaluate on CIFAR-10 dataset with the deep learning features that we have extracted in Section V-C,

Fig. 7. Visualization of CIFAR-10 dataset retrieval results with 4096-dim binary codes. Red border means false positive.

TABLE IV
IMAGE RETRIEVAL PERFORMANCE ON CIFAR-10 DATASET [20]. NOTE THAT CCQ-ITQ CANNOT PRODUCE BINARY CODES THAT ARE LONGER THAN THE ORIGINAL FEATURE VECTOR

		mAP	Encoding time (ms)
raw deep features (4096-d)		72.1%	-
1024 bits	CCA-ITQ [1]	59.8%	4.34
	SSH [28]	58.2%	4.34
	SDH [19]	**62.7%**	4.34
	SSP [3]	60.1%	0.47
	FBE (ours)	45.4%	1.12
	SuFBE (ours)	61.2%	
4096 bits	CCA-ITQ [1]	61.6%	12.8
	SSH [28]	60.8%	12.8
	SDH [19]	64.4%	12.8
	SSP [3]	65.9%	1.33
	FBE (ours)	52.4%	1.12
	SuFBE (ours)	**69.1%**	
8192 bits	CCA-ITQ [1]	-	-
	SSH [28]	62.7%	21.7
	SDH [19]	66.8%	21.7
	SSP [3]	68.3%	3.12
	FBE (ours)	60.2%	1.88
	SuFBE (ours)	**71.8%**	

i.e., the 4096-dimensional responses of second fc layer of a fine-tuned AlexNet [6]. We follow the settings in the previous works [43], [44] for the CIFAR experiments, *i.e.*, we randomly select 100 images per class (1,000 images in total) as the test query set, 500 images per class (5,000 images in total) as the training set. We then learn the hashing functions of each method on the feature vectors of training set of CIFAR-10, and use the learned hashing functions to project feature vectors of both training set and test set into binary codes.

The quantitative results are reported in Table IV, from which we have the following three observations:

- First, we could see that the mAP of all methods in Table IV increase with the code length, which suggests that long-bit binary codes can better preserve the discriminative power of the high-dimensional deep feature than compact ones. As shown in the last row of Table IV, the 8, 192-dimensional binary codes of our SuFBE achieve the mAP at 71.8%, which has almost no degradation compared with 72.1% of the raw deep learning feature. However, we didn't observe better mAP using more bits.
- Second, the proposed SuFBE can outperform other state-of-the-art methods in case of long-bit case such as 4, 096 and 8, 192 bits. In terms of the encoding efficiency, the proposed SuFBE has the least encoding time using 4, 096 and 9, 192 bits. In case of 8, 192 bits, SuFBE can be 10+ times faster than CCA-ITQ [1], SSH [28] and SDH [19]. The retrieval results of a selected query is visualized in Figure 7.
- Third, by taking the labels of training data into consideration, the SuFBE can significantly outperform its unsupervised counterpart FBE. The mAP gap can be up to 17%, which validates the rationale of building SuFBE based on FBE.

B. Evaluation on NUS-WIDE dataset

Besides the CIFAR-10 dataset of tiny natural image with single label, we also evaluate our method on NUS-WIDE dataset, which contains about 270, 000 images from Flickr. NUS-WIDE is associated with 81 ground truth concept labels, with each image containing multiple semantic labels. We define the true neighbours of a query as the images sharing at least one labels with the query image. Due to the promising performance of DNNs [6], [7], we use the pre-trained AlexNet [6] provided by Caffe [34] to extract deep learning features for images in NUS-WIDE datasets. Each image is represented by the 4096-dimensional responses of the second fc layer. As in [19], we collect the 21 most frequent label for test. For each label, 100 images are uniformly sampled for the query set and the remaining images are for the training set.

The mAP obtained of all methods with varying code lengths are shown in Table V. In case of long code lengths such as 4096 and 8192 bits, SuFBE performs the best with the shortest encoding time. The mAP does not increase if we set the code length as 16, 394 bits. The superior results of SuFBE

TABLE V
IMAGE RETRIEVAL PERFORMANCE ON NUS-WIDE DATASET [42]

		mAP	Encoding time (ms)
raw deep features (4096-d)		57.4%	-
1024 bits	CCA-ITQ [1]	42.7%	4.34
	SSH [28]	40.1%	4.34
	SDH [19]	44.8%	4.34
	SSP [3]	**46.2%**	0.47
	FBE (ours)	28.6%	1.12
	SuFBE (ours)	45.6%	
4096 bits	CCA-ITQ [1]	48.7%	12.8
	SSH [28]	46.2%	12.8
	SDH [19]	49.6%	12.8
	SSP [3]	50.5%	1.33
	FBE (ours)	37.8%	1.12
	SuFBE (ours)	**52.4%**	
8192 bits	CCA-ITQ [1]	-	-
	SSH [28]	47.8%	21.7
	SDH [19]	51.9%	21.7
	SSP [3]	53.2%	3.12
	FBE (ours)	41.0%	1.88
	SuFBE (ours)	**56.9%**	

validate the observations in Section VIII-A and demonstrate the effectiveness and efficiency of our method on the retrieval task of data with multiple semantic labels.

C. Discussions

In the above experiments (Table IV,V), we observe that our SuFBE not only inherits the efficiency advantages of FBE, but also takes the training labels into considerations and leads to good performance in supervised hashing tasks. Specifically, our SuFBE can outperform other state-of-the-art supervised methods in terms of both accuracy and efficiency in long-bit cases. Meanwhile, SuFBE achieves far better results than FBE [18] as shown in above experiments, which validate the rationale of developing SuFBE.

IX. CONCLUSION

We have proposed Fried Binary Embedding (FBE) and Supervised Fried Binary Embedding (SuFBE), for high-dimensional binary embedding. By decomposing the dense projection matrix using the adaptive Fastfood transform, our proposed FBE reduces the original computational and memory cost of $\mathcal{O}(d^2)$ to $\mathcal{O}(d \log d)$ and $\mathcal{O}(d)$, respectively. Moreover, due to the inherent structure in each of its components, the ultimate projection matrix would have restricted freedom, which is naturally against overfitting and shows promising accuracy in our experiments. We split the optimization problem with structured matrices involved into several feasible sub-problems, then we iteratively solve these sub-problems till convergence. We compare FBE with several state-of-the-art methods on various tasks, including approximate nearest neighbor (ANN) search, image retrieval, and image classification. Experimental results validate the efficiency and accuracy advantages of our FBE.

REFERENCES

[1] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[2] Y. Xia, K. He, P. Kohli, and J. Sun, "Sparse projections for high-dimensional binary codes," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 3332–3339.

[3] F. Tung and J. J. Little, "SSP: Supervised sparse projections for large-scale retrieval in high dimensions," in *Proc. Asian Conf. Comput. Vis.* New York, NY, USA: Springer, 2016, pp. 338–352.

[4] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice* (Neural Information Processing series). Cambridge, MA, USA: MIT Press, 2006.

[5] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1753–1760.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[7] K. Simonyan and A. Zisserman. (2014). "Very deep convolutional networks for large-scale image recognition." [Online]. Available: https://arxiv.org/abs/1409.1556

[8] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[9] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[10] J. Sánchez and F. Perronnin, "High-dimensional signature compression for large-scale image classification," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2011, pp. 1665–1672.

[11] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2013, pp. 484–491.

[12] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang, "Circulant binary embedding," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1–9.

[13] W. Hong, J. Meng, and J. Yuan, "Tensorized projection for high-dimensional binary embedding," in *Proc. 13th AAAI Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, 2018, pp. 69–76.

[14] J. Song, H. Zhang, X. Li, L. Gao, M. Wang, and R. Hong, "Self-supervised video hashing with hierarchical binary auto-encoder," *IEEE Trans. Image Process.*, vol. 27, no. 7, pp. 3210–3221, Jul. 2018.

[15] J. Song, L. Gao, L. Liu, X. Zhu, and N. Sebe, "Quantization-based hashing: A general framework for scalable image and video retrieval," *Pattern Recognit.*, vol. 75, pp. 175–187, Mar. 2018.

[16] Q. Le, T. Sarlós, and A. Smola, "Fastfood—Approximating kernel expansions in loglinear time," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1–11.

[17] Z. Yang *et al.*, "Deep fried convnets," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 1476–1483.

[18] W. Hong, J. Yuan, and S. D. Bhattacharjee, "Fried binary embedding for high-dimensional visual features," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 6221–6229.

[19] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 37–45.

[20] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," Tech. Rep., 2009.

[21] R. Courant, "Variational methods for the solution of problems of equilibrium and vibrations," *Bull. Amer. Math. Soc.*, vol. 49, no. 1, pp. 1–23, 1943.

[22] Y. Wang, J. Yang, W. Yin, and Y. Zhang, "A new alternating minimization algorithm for total variation image reconstruction," *SIAM J. Imag. Sci.*, vol. 1, no. 3, pp. 248–272, Aug. 2008.

[23] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.

[24] X. Zhang, F. X. Yu, R. Guo, S. Kumar, S. Wang, and S.-F. Chang, "Fast orthogonal projection based on kronecker product," in *Proc. IEEE Int. Conf. Comput. Vis.*, Dec. 2015, pp. 2929–2937.

[25] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, nos. 3–4, pp. 321–377, 1936.

[26] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 3304–3311.

[27] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2007, pp. 1–8.

[28] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 12, pp. 2393–2406, Dec. 2012.

[29] P. H. Schönemann, "A generalized solution of the orthogonal procrustes problem," *Psychometrika*, vol. 31, no. 1, pp. 1–10, 1966.
[30] J. C. Gower and G. B. Dijksterhuis, *Procrustes Problems*. London, U.K.: Oxford Univ. Press, 2004, no. 30.
[31] J. Nocedal and S. J. Wright, *Sequential Quadratic Programming*. New York, NY, USA: Springer, 2006.
[32] J. Meng, H. Wang, J. Yuan, and Y.-P. Tan, "From keyframes to key objects: Video summarization by representative object proposal selection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1039–1048.
[33] S. D. Bhattacharjee, J. Yuan, W. Hong, and X. Ruan, "Query adaptive instance search using object sketches," in *Proc. ACM Multimedia Conf.*, 2016, pp. 1306–1315.
[34] Y. Jia *et al.*, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. 22nd ACM Int. Conf. Multimedia*, 2014, pp. 675–678.
[35] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *Proc. 1st ACM Int. Conf. Multimedia Inf. Retr.*, 2008, pp. 39–43.
[36] Z. Wang, J. Meng, T. Yu, and J. Yuan, "Common visual pattern discovery and search," in *Proc. Asia–Pacific Signal Inf. Process. Assoc. Annu. Summit Conf.*, Dec. 2017, pp. 1011–1018.
[37] T. Yu, Z. Wang, and J. Yuan, "Compressive quantization for fast object instance search in videos," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Oct. 2017, pp. 726–735.
[38] W. Hong, J. Meng, and J. Yuan, "Distributed composite quantization," in *Proc. 30th AAAI Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, 2018, pp. 61–68.
[39] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.
[40] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2018.
[41] W. Liu, J. Wang, S. Kumar, and S.-F. Chang, "Hashing with graphs," in *Proc. 28th Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 1–8.
[42] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "Nus-wide: A real-world Web image database from national university of Singapore," in *Proc. ACM Conf. Image Video Retr.*, Santorini, Greece, 2009, Art. no. 48.
[43] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proc. 25th Int. Joint Conf. Artif. Intell.* Palo Alto, CA, USA: AAAI Press, 2016, pp. 1711–1717.
[44] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 3270–3278.

Weixiang Hong received the bachelor's degree in software engineering from Fudan University, China, in 2015. He is currently pursuing the master's degree in software engineering with National University of Singapore. Since 2015, he has been with Nanyang Technological University under the supervision of Prof. J. Yuan.

His current research interests include computer vision, and machine learning and optimization. He has served as a reviewer for the IEEE TRANSACTIONS ON IMAGE PROCESSING, *The Visual Computer*, the IEEE Conference on Computer Vision and Pattern Recognition, and the IEEE Conference on Computer Vision and Pattern Recognition.

Junsong Yuan (M'08–SM'14) received the B.E. degree in engineering from the Special Class for the Gifted Young of the Huazhong University of Science and Technology, China, in 2002, and received the M.Eng. degree from the National University of Singapore in 2005 and the Ph.D. degree from Northwestern University in 2009.

He was an Associate Professor with the School of Electrical and Electronics Engineering, Nanyang Technological University (NTU), Singapore. He is currently an Associate Professor with the Computer Science and Engineering Department, University at Buffalo, The State University of New York, USA. He received the Best Paper Award from the IEEE TRANSACTIONS ON MULTIMEDIA, the Doctoral Spotlight Award from the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'09), the Nanyang Assistant Professorship from NTU, and the Outstanding EECS Ph.D. Thesis Award from Northwestern University.

Dr Yuan is currently the Senior Area Editor of the *Journal of Visual Communication and Image Representation*, an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, and served as the Guest Editor of *International Journal of Computer Vision*. He is the Program Co-Chair of ICME'18 and VCIP'15, and the Area Chair of ACM MM, ACCV, ICPR, CVPR, and ICIP. He is a Fellow of the International Association of Pattern Recognition.