Fried Binary Embedding for High-Dimensional Visual Features

Weixiang Hong Junsong Yuan Sreyasee Das Bhattacharjee
School of Electrical and Electronic Engineering,
Nanyang Technological University, Singapore
{wxhong,JSYUAN}@ntu.edu.sg, sreya.iitm@gmail.com

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 neighbour search. Generating long binary codes involves large projection matrix and high-dimensional matrix-vector multiplication, thus is memory and compute intensive. To tackle these problems, we propose Fried Binary Embedding (FBE) to decompose the projection matrix using adaptive Fastfood transform, which is the multiplication of several structured matrices. As a result, FBE 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 can regulate the projection matrix against overfitting and lead to even better accuracy than using unconstrained projection matrix (like ITQ [4]) 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 the FBE.

1. Introduction

Recently, the vision community has devoted a lot of attention to binary embedding [22, 4, 3, 25, 28, 21] due to the explosive growth in visual data, which makes efficient searching and storing to be urgent requirements. Binary embedding aims at encoding high-dimensional feature vectors to compact binary codes, while preserving the pair-wise similarities of the original high-dimensional features. By encoding high-dimensional features into binary codes, one can perform efficient searching with the Hamming distance computation; meanwhile, the storage cost is also significantly reduced. Since it is NP-hard to directly learn the optimal binary codes [22], 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.

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 [23], 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 $\mathcal{O}(d)$ [17, 3, 25]. 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 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. Representative works include Circulant Binary Embedding(CBE) [25], Bilinear Projection(BP) [3], Fast Orthogonal Projection(KBE) [28] and Sparse Projection(SP) [23], *etc*. To be more specific, CBE [25] utilizes circulant matrix to accelerate the matrix-vector multiplication, BP [3] constrains the projection matrix to be Kronecker product of two smaller matrices, while SP [23] seeks for a sparse projection matrix. Al-

though CBE [25] and BP [3] show promising encoding efficiency, they achieve inferior accuracy to dense projection method (like ITQ [4]) using the same code length. While SP [23] is competitive when compared with dense projection method in terms of performance. Unfortunately, it still suffers from the $\mathcal{O}(d^2)$ computational cost.

To address the two challenges above, we propose a novel approach, Fried Binary Embedding (FBE), for generating effective long binary codes efficiently. The idea is to decompose the projection matrix R using the adaptive Fastfood transform [12, 24], 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 [23] and 0.1% of ITQ [4]. 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.

The involvement of structured matrices makes the optimization problem difficult, thus we adopt the variable-splitting and penalty techniques [23, 2, 20] to develop an alternative optimization algorithm. We introduce an auxiliary variable to split the original optimization problem into several feasible sub-problems, then we iteratively solve these sub-problems till convergence. In Section 4.4, we further show that our algorithm provably converges to a local optimum. We call our approach as Fried Binary Embedding (FBE) following Deep Fried Convnets [24] and Circulant Binary Embedding [25]. Extensive experiments show that our approach not only achieves competitive performance in compact-bit case, but also outperforms state-of-the-art methods in long-bit scenario.

2. Related Work

A good review of binary embedding can be found in [19]. Here we focus on several closely related works, including Iterative Quantization (ITQ) [4] and its four variants. The Iterative Quantization is essential to the understanding of our work. The three variants of ITQ are Bilinear Projection (BP) [3], Circulant Binary Embedding (CBE) [25], Sparse Projection (SP) [23] and Fast Orthogonal Projection (KBE) [28], which are state-of-the-art works for high-dimensional binary embedding.

Iterative Quantization (ITQ) [4] aims to find the hash codes such that the difference between the hash codes and the data items, by viewing each bit as the quantization value along the corresponding dimension, is minimized. It consists of two steps: (1) dimension reduction via PCA; (2) find the hash codes as well as an optimal rotation.

Bilinear Projections (BP) [3] 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 [13], GIST [15], VLAD [8], and Fisher Vectors [16], but is not true for learned features such as those learned by the deep neural network(DNN) [11, 18].

Circulant Binary Embedding (CBE) [25] 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 [4]) using the same code length.

Fast Orthogonal Projection (KBE) [28] decomposes the projection matrix as a series of smaller orthogonal matrices. Similar to CBE [25], there exists a fast algorithm to compute Kronecker projection with a time complexity of $\mathcal{O}(d \log d)$, therefore, KBE has shown great advantages in terms of efficiency, compare with dense projection method (like ITQ [4]).

Sparse Projection (SP) [23] introduces a sparsity regularizer to achieve efficiency in encoding. They also show that there exist many redundant parameters in dense projection matrix. However, their method requires the percentage of non-zero elements to be around 10% for competitive performance, according to Figure 4 in [23], which can be still suffering in case that both *d* and *b* are very large.

3. Problem Formulation

Following [19, 23], we put dimension reduction and optimal rotation of ITQ [4] into one integrated objective:

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

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 [4] solves Equation 1 via alternative update. After finding R, ITQ can produce binary codes using the hash function below:

$$\mathbf{c} = \mathrm{sgn}(\mathrm{R}\mathbf{x}), \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^d$ denotes a data vector, and $\text{sgn}(\cdot)$ 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 [4] 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 [24], *i.e.*,

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

Consequently, our hash function turns to be

$$\mathbf{c} = \text{sgn}(\mathrm{SHG\Pi HB}\mathbf{x}). \tag{4}$$

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 module:

- S, G and B are diagonal matrices of tunable parameters. As a comparison, S, G and B in the original non-adaptive Fastfood formulation [12] 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.*, $\mathbf{s} = \text{diag}(\mathrm{S})$.

- $\Pi \in \{0,1\}^{d\times d}$ 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 our optimization objective by putting Equation 1 and 3 together:

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

4. 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 [23, 2, 20]. 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:

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

where β is a penalty weight. Such a relaxation is similar to Half-Quadratic Splitting [20]. 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 $\beta \to \infty$ [20]. We solve Equation 6 in an alternating manner: update one variable with others fixed.

4.1. Update 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 $\mathrm{C}_{ij} = \text{sgn}((\bar{\mathrm{R}}\mathrm{X})_{ij})$, or simply

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

4.2. Update $\bar{\mathrm{R}}$

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

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

where $\mathrm{Y} = (\mathrm{C} + \beta\mathrm{RX})/(1+\beta)$. This problem is known as the orthogonal procrustes problem [5] and is recently widely studied in binary embedding [4, 3, 23].

According to [5], 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 [23]. 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 4.4.

4.3. 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, and share the similar form of solution. 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:

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

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:

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

$\|\mathrm{SHG\Pi HBX}\|_{\mathrm{F}}^2$ 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:

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

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,

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

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 $\mathrm{S}, \mathrm{G}, \mathrm{B}$ respectively.

Update S

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

$$\begin{aligned} \mathrm{D}_\mathrm{S} &= \mathrm{I}, \\ \mathrm{E}_\mathrm{S} &= \mathrm{HG\Pi HBX}, \\ \mathrm{K}_\mathrm{S} &= \mathrm{HG\Pi HBXZ}^\mathrm{T}, \end{aligned} \tag{21}$$

so the update rule for S according to Equation 20 is

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

Update G

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

$$\begin{aligned} \mathrm{D}_\mathrm{G} &= \mathrm{SH}, \\ \mathrm{E}_\mathrm{G} &= \mathrm{\Pi HBX}, \\ \mathrm{K}_\mathrm{G} &= \mathrm{\Pi HBXZ}^\mathrm{T}\mathrm{SH}, \end{aligned} \tag{23}$$

so the update rule for G according to Equation 20 is

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

Update B

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

$$\begin{aligned} \mathrm{D}_\mathrm{B} &= \mathrm{SHG\Pi H}, \\ \mathrm{E}_\mathrm{B} &= \mathrm{X}, \\ \mathrm{K}_\mathrm{B} &= \mathrm{XZ}^\mathrm{T}\mathrm{SHG\Pi H}, \end{aligned} \tag{25}$$

so the update rule for B according to Equation 20 is

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

4.4. Discussions

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 $(\lceil b/d \rceil * d - b)$ 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 4.1–4.3. We initialize $\mathrm{S} = \frac{1}{\mathrm{d}^2}\mathrm{I}, \mathrm{G} = \mathrm{I}, \mathrm{B} = \mathrm{I}, \bar{\mathrm{R}} = \mathrm{R}$ to satisfy the orthogonal constraint. The training data is subtracted with its mean prior to learning. We summarize our proposed FBE in Algorithm 1.

[1]This strategy actually also works for Circulant Binary Embedding [25], which is previously considered unavailable to produce binary codes that are longer than original feature vector [23].

Algorithm 1 Fried Binary Embedding

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

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 YX^T such that $\mathrm{YX}^\mathrm{T} = \mathrm{U\Sigma V}^\mathrm{T}$
  $\bar{\mathrm{R}} \leftarrow \mathrm{UV}^\mathrm{T}$
  $\mathrm{D_S} \leftarrow \mathrm{I}, \mathrm{E_S} \leftarrow \mathrm{HG\Pi HBX}, \mathrm{K_S} \leftarrow \mathrm{HG\Pi HBXZ}^\mathrm{T}$
  $\mathrm{diag(S)} \leftarrow \left[\left(\mathrm{E_S E_S}^\mathrm{T}\right) \odot \left(\mathrm{D_S}^\mathrm{T} \mathrm{D_S}\right)\right]^{-1} \times \mathrm{diag(K_S)}$
  $\mathrm{D_G} \leftarrow \mathrm{SH}, \mathrm{E_G} \leftarrow \mathrm{\Pi HBX}, \mathrm{K_G} \leftarrow \mathrm{\Pi HBXZ}^\mathrm{T}\mathrm{SH}$
  $\mathrm{diag(G)} \leftarrow \left[\left(\mathrm{E_G E_G}^\mathrm{T}\right) \odot \left(\mathrm{D_G}^\mathrm{T} \mathrm{D_G}\right)\right]^{-1} \times \mathrm{diag(K_G)}$
  $\mathrm{D_B} \leftarrow \mathrm{SHG\Pi H}, \mathrm{E_B} \leftarrow \mathrm{X}, \mathrm{K_B} \leftarrow \mathrm{XZ}^\mathrm{T}\mathrm{SHG\Pi H}$
  $\mathrm{diag(B)} \leftarrow \left[\left(\mathrm{E_B E_B}^\mathrm{T}\right) \odot \left(\mathrm{D_B}^\mathrm{T} \mathrm{D_B}\right)\right]^{-1} \times \mathrm{diag(K_B)}$
until convergence
return $\mathrm{S}, \mathrm{G}, \mathrm{B}, \Pi$

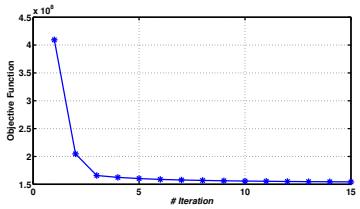


Figure 1: Convergence of our algorithm. The vertical axis represents the objective function value of Equation 6 and the horizontal axis corresponds to the number of iterations at Algorithm 1. The optimization of R is obtained on the training set of DNN-4096 [23].

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 [20]. 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.

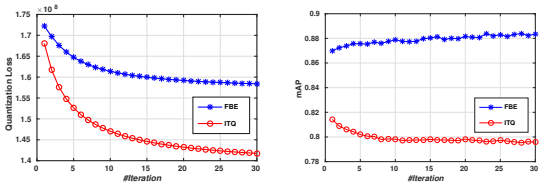
Since all 5 sub-problems (4 of them are convex) we tackle have optimal solutions individually, our Algorithm 1 should converge fast. As shown in Figure 1, the objective function value at each iteration in the Algorithm 1 always decreases. Considering that the objective function value is also lower-bounded (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.

The objective function of ITQ [4]

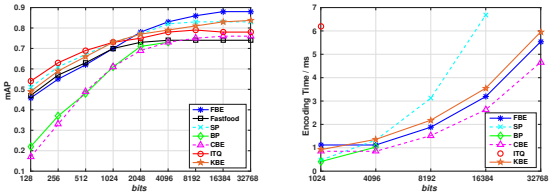
$$||\mathrm{RX} - \mathrm{C}||_{\mathrm{F}}^2 \tag{27}$$

is widely-used in hashing, because it has the physical meaning, *i.e.*, the smaller the quantization loss is, the more precisely the decimal space and binary space align, thus should lead to better performance. However, we found that this is not absolutely true. We monitor the quantization loss and mAP of ITQ and FBE when we train them on DNN-4096 dataset [23]. As shown in Figure 2, ITQ has the lower quantization loss as expected. However, the mAP of ITQ is going down while its quantization loss is decreasing, which validates the existence of overfitting in high-dimensional binary embedding. Our FBE can well regulate the projection loss, hence have higher quantization loss than ITQ, which perhaps leads to the superior mAP of our FBE.



(a) Quantization Loss (b) mAP

Figure 2: The quantization loss and mAP of ITQ and FBE when training on DNN-4096 dataset. While ITQ fits the data better and better, the mAP goes lower and lower.



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

Figure 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 $(b > d)$.

5. Experiments

To evaluate proposed Fried Binary Embedding (FBE), we conduct experiments on three tasks: approximate nearest neighbour (ANN) search, image retrieval, and image classification, following the experiments setting in [23]. For each task, we compare our method (FBE) with the original Fastfood transform [12], as well as the several state-of-the-art methods for high-dimensional visual feature embedding, including Iterative Quantization (ITQ) [4], Bilinear Projection (BP) [3], Circulant Binary Embedding (CBE) [25], Fast Orthogonal Projection (KBE) [28] and Sparse Projection (SP) [23]. For the original Fastfood transform [12] where S, G and B are randomly generated instead of being optimized, we report the best performance achieved by varying the 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 4. 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.

5.1. Approximate Nearest Neighbor Search

5.1.1 Experiments on DNN features

Recent research advances have demonstrated the advantage of deep learning features as image representations [11, 18]. We first conduct experiments on such features. We use the pre-trained AlexNet [11] provided by Caffe [9] to extract deep learning features for one million images in MIRFLICKR-1M dataset [6]. 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 [23], 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 neighbours 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.

bits	2048	4096	8192	16384	32768
ITQ [4]	8.3×10^6	1.6×10^7	3.2×10^7	6.4×10^7	1.2×10^8
SP [23]	8.3×10^5	1.6×10^6	3.2×10^6	6.4×10^6	1.2×10^7
BP [3]	6144	8192	-	-	-
CBE [25]	8192	8192	16384	32768	65536
KBE [28]	88	96	104	112	120
FBE(ours)	12288	12288	24576	49152	98304

Table 1: 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.

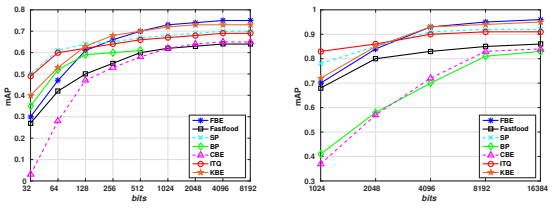
For example, with 2048 bits or more, FBE performs the best of embedding the 4096-dimensional CNN features, when compared with SP [23], KBE [28], BP [3], CBE [25] and ITQ [4]. Interestingly, although the performance of our proposed method is not better than that of Fastfood transform [12] below 1024 bits, it significantly outperforms Fastfood transform [12] 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 [12] does.

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 1. 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 being 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 [23] and 0.1% of ITQ [4]. Although BP [3], CBE [25] and KBE [28] 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 4.

5.1.2 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 [7], which contains one million 960-dimensional GIST features [15] and 10,000 queries. The second dataset is VLAD-25600 [23]. The VLAD features [8] are extracted from 100,000 images randomly sampled from the INRIA image set [7]. The 25600-dimensional VLAD features are generated by encoding 128-dimensional SIFT vectors [13] to a 200-center codebook. An extra random subset of 1000 samples are used as queries in this dataset.



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

Figure 4: Comparison on 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.

Figure 4 shows the approximate nearest neighbour 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.

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.

5.2. Image Retrieval

We evaluate the performance of binary embedding for the image retrieval task on the "Holidays + MIRFlickr-1M" dataset [7]. 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 neighbours under the same scene category. In light of the good performances reported in recent image retrieval and object search works that harness DNN features as image descriptors [1, 27, 26, 14], we represent each image by the 4096-dimensional response of the second fc layer of AlexNet [11].

Following previous practices [8, 3, 22, 23], 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 2 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 [3], CBE [25], SP [23], KBE [28] and original Fastfood trans-

		mAP	Encoding time (ms)
raw deep features (4096-d)		49.5%	-
1024 bits	BP [3]	44.5%	0.41
	CBE [25]	44.3%	0.85
	SP [23]	44.9%	0.47
	KBE [28]	45.0%	0.92
	Fastfood [12]	44.7%	1.12
	FBE(ours)	**45.6%**	
4096 bits	BP [3]	46.4%	1.03
	CBE [25]	46.6%	0.85
	SP [23]	47.1%	1.33
	KBE [28]	47.2%	1.36
	Fastfood [12]	46.2%	1.12
	FBE(ours)	**47.2%**	
8192 bits	CBE [25]	47.8%	1.52
	SP [23]	48.5%	3.12
	KBE [28]	48.8%	2.18
	Fastfood [12]	47.6%	1.88
	FBE(ours)	**49.3%**	

Table 2: Image retrieval performance on Holidays+1M.

form [12] 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.

5.3. Image Classification

We further evaluate the binary codes as compact features for image classification on CIFAR-10 dataset [10], using top-1 accuracy as the metric. As a baseline, we extract the 4096-dimensional responses of second fc layer in AlexNet [11] as image features. We first fine-tune the pre-trained model provided by Caffe [9] 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 [10] training set.

Following [23], 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 [3], CBE [25], SP [23], KBE [28] and original Fastfood transform [12] using 1024, 4096, 8192 and 16384 bits. We do not see significant improvement of the performance when further increasing the bit length.

Table 3 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 1/8 storage cost of raw 4096-dimensional feature of real numbers.

		Classification accuracy	Encoding time (ms)
raw deep feature (4096-d)		82.6%	-
1024 bits	BP [3]	76.6%	0.41
	CBE [25]	76.3%	0.85
	SP [23]	77.8%	0.47
	KBE [28]	78.3%	0.92
	Fastfood [12]	77.4%	1.12
	FBE(ours)	**79.7%**	
4096 bits	BP [3]	77.5%	1.03
	CBE [25]	77.4%	0.85
	SP [23]	78.6%	1.33
	KBE [28]	79.3%	1.36
	Fastfood [12]	78.3%	1.12
	FBE(ours)	**80.7%**	
8192 bits	CBE [25]	78.1%	1.52
	SP [23]	79.5%	3.12
	KBE [28]	80.5%	2.18
	Fastfood [12]	79.0%	1.88
	FBE(ours)	**81.6%**	
16384 bits	CBE [25]	78.6%	2.63
	SP [23]	80.2%	6.68
	KBE [28]	81.0%	3.55
	Fastfood [12]	79.3%	3.19
	FBE(ours)	**82.4%**	

Table 3: Classification accuracy on CIFAR-10 dataset.

5.4. Discussion

In the above experiments (Figure 3a,4 and Table 2,3), 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.

6. Conclusion

We propose a novel approach Fried Binary Embedding (FBE) 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 better generalization performance in our experiments. We introduce an auxiliary variable to 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 three tasks, including approximate nearest neighbour (ANN) search, image retrieval, and image classification. Experimental results validate the efficiency and accuracy advantages of our FBE.

Acknowledgements

This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114 and was carried out at the Rapid-Rich Object Search (ROSE) Lab in the Nanyang Technological University, Singapore. The ROSE Lab is supported by the National Research Foundation, Singapore, under its Interactive Digital Media (IDM) Strategic Research Programme. We gratefully acknowledge the support of NVAITC (NVIDIA AI Technology Centre) for their donation of a Tesla K80 and M60 GPU used for our research at the ROSE Lab.

The authors thank Tan Yu for valuable discussions.

References

[1] S. D. Bhattacharjee, J. Yuan, W. Hong, and X. Ruan. Query adaptive instance search using object sketches. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 1306–1315. ACM, 2016.

[2] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–23, 1943.

[3] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 484–491, 2013.

[4] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 817–824. IEEE, 2011.

[5] J. C. Gower and G. B. Dijksterhuis. *Procrustes problems*. Number 30. Oxford University Press on Demand, 2004.

[6] M. J. Huiskes and M. S. Lew. The mir flickr retrieval evaluation. In *Proceedings of the 1st ACM international conference on Multimedia information retrieval*, pages 39–43. ACM, 2008.

[7] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2011.

[8] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3304–3311. IEEE, 2010.

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.

[10] A. Krizhevsky. Learning multiple layers of features from tiny images. 2009.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[12] Q. Le, T. Sarlós, and A. Smola. Fastfood-approximating kernel expansions in loglinear time. In *Proceedings of the international conference on machine learning*, 2013.

[13] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[14] J. Meng, H. Wang, J. Yuan, and Y.-P. Tan. From keyframes to key objects: Video summarization by representative object proposal selection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1039–1048, 2016.

[15] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International journal of computer vision*, 42(3):145–175, 2001.

[16] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[17] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1665–1672. IEEE, 2011.

[18] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International conference on learning representations*, 2015.

[19] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[20] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 1(3):248–272, 2008.

[21] Z. Wang, L.-Y. Duan, J. Yuan, T. Huang, and W. Gao. To project more or to quantize more: Minimizing reconstruction bias for learning compact binary codes. *International Joint Conference on Artificial Intelligence*, 2016.

[22] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in neural information processing systems*, pages 1753–1760, 2009.

[23] Y. Xia, K. He, P. Kohli, and J. Sun. Sparse projections for high-dimensional binary codes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3332–3339, 2015.

[24] Z. Yang, M. Moczulski, M. Denil, N. de Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

[25] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *International conference on machine learning*, volume 6, page 7, 2014.

[26] T. Yu, Y. Wu, S. D. Bhattacharjee, and J. Yuan. Efficient object instance search using fuzzy object matching. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press, 2017.

[27] T. Yu, Y. Wu, and J. Yuan. Hope: Hierarchical object prototype encoding for efficient object instance search in videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[28] X. Zhang, F. X. Yu, R. Guo, S. Kumar, S. Wang, and S.-F. Chang. Fast orthogonal projection based on kronecker product. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2929–2937, 2015.