# Computationally Bounded Retrieval

Mohammad Rastegari[1], Cem Keskin[2], Pushmeet Kohli[2], Shahram Izadi[2]
[1]University of Maryland, [2]Microsoft Research
mrastega@cs.umd.edu, cemke,pkohli,shahrami@microsoft.com

## Abstract

*The increase in size of large image databases makes the problem of efficient retrieval extremely challenging. This is especially true in the case of high dimensional data where even operations like hashing become expensive because of costly projection operators. Unlike most hashing methods that sacrifice accuracy for speed, we propose a novel method that improves the speed of high dimensional image retrieval by several orders of magnitude without any significant drop in performance. To do this, we propose to learn computationally bounded sparse projections for the encoding step. To further increase the accuracy of the method, we add an orthogonality constraint on projections to reduce bit correlation. We then introduce an iterative scheme that jointly optimizes this objective, which helps us obtain fast and efficient projections. We demonstrate this technique on large retrieval databases, specifically ImageNET, GIST1M and SUN-attribute for the task of nearest neighbor retrieval, and show that our method achieves a speed-up of up to a factor of 100 over state-of-the-art methods, while having on-par and in some cases even better accuracy.*

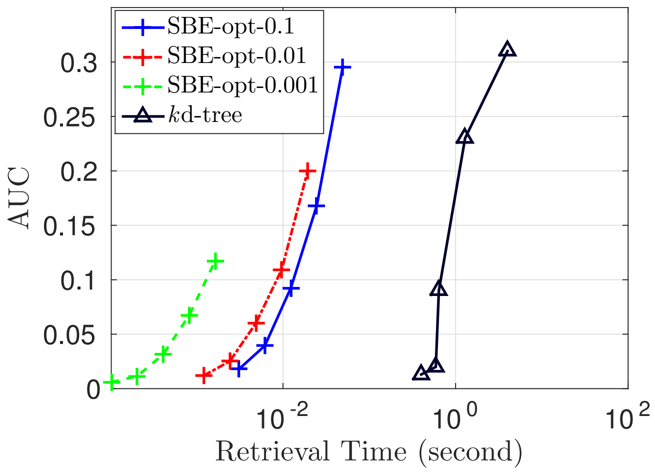

Figure 1: Comparison of the accuracy (area-under-the-curve) and computational cost of the proposed algorithm (SBE) with the $kd$-tree method for different sparsity values. The curves correspond to increasing code lengths for SBE and decreasing bucket size for $kd$-tree. Our method achieves the same accuracy as a $kd$-tree while being 100 times faster.

## 1. Introduction

Many computer vision problems can be formulated as a nearest neighbor search in some large dataset. When the data involved is high dimensional, doing this search efficiently becomes crucial but also extremely challenging. This is especially the case for the task of content based image retrieval, where efficiency is a requirement due to the high dimensionality of the data and the increase in size and availability of these databases.

The most common retrieval approaches are *tree based* and *hashing based* methods. In tree based approaches, the search space is embedded into a tree structure. At query time, the tree is traversed until a leaf node is reached, which points to a set of images that are similar to the query image. Then a final search is conducted over this set of images to find the nearest neighbor. Decision trees [17] and $kd$-trees[13, 12] are two such methods. When the number of dimensions grows, these conventional approaches often become less efficient, since the time or space requirements of these approaches often grows exponentially with the dimensionality.

Hashing based methods, on the other hand, reduce the number of dimensions involved with projections. This in turn removes the exponential dependence on dimensionality that trees suffer from. For instance, Locality Sensitive Hashing (LSH) uses random projections for the *encoding* step [2]. The images are indexed by hash tables, and at test time the query is encoded into an index that points to these hash tables. Typically the encoding step is the bottleneck in these image retrieval methods.

In this paper, we focus primarily on image retrieval using binary codes [26]. In this approach, each image is encoded into a binary code such that the nearest neighbors in the corresponding Hamming space remain the same as the actual nearest neighbors in the original feature space. These binary codes can be used as hash keys to construct a hash table for real-time nearest neighbor search [15]. Calculating the binary code, *i.e.* encoding, is in this case evaluating a mapping function $f$ from some $d$-dimensional inputs to $k$-dimensional outputs. However if $d$ and $k$ are large, computational efficiency of $f$ becomes a bottleneck at test time. Since the output is binary, this can be treated as a linear classification problem per bit. Even though there have been several efforts on making non-linear classification more efficient [21, 11], none of these methods are faster than linear classifiers, whereas making $f$ more efficient would require sub-linear time complexity classifiers.

Typically, a single bit in a binary code is obtained by a dot product between the normal vector of a hyperplane and the feature vector, which implies $\mathcal{O}(dk)$ time complexity. In [5] a bilinear projection is employed to reduce the time complexity to $\mathcal{O}(d\sqrt{k})$. Recently, [25] proposed to use the columns of a circulant matrix as linear projections. This enables faster projection to generate $k$-bits ($k > 1$) by Fast Fourier Transform (FFT) which is $\mathcal{O}(d\log(d))$. In this paper, we propose an optimization that learns a sparse projection to binary codes with a constraint on the computational budget.

We propose to find a mapping $f$ that quantizes the data into binary values while: 1-*minimizing the quantization error*, and 2-*minimizing the computational cost of* $f$. We introduce two optimizations that employ $\ell_1$-norm to constrain the computational operations in $f$ based on the given sparsity rate $\alpha$. In the first case we assume the binary codes are given and we only optimize for the mapping function $f$. In the second case we jointly optimize for binary codes and $f$. The joint optimization also introduces a new framework for learning binary codes, where the linear mapping function can be replaced by any arbitrary function (*e.g.* Neural Networks).

We demonstrate our method in several image retrieval datasets (ImageNet, GIST, SUN) for nearest neighbor search. Our method achieves a high accuracy while having orders of magnitude less operations in comparison with the state-of-the-art methods on fast binary embedding. Surprisingly, our joint optimization even outperforms the baseline iterative quantization (ITQ) method [6] in terms of accuracy for some values of $\alpha$. $\alpha$ provides a way of controlling the trade-off between speed and accuracy. Figure 1 shows that we achieve comparable accuracies to the $kd$-tree method using our algorithm, which requires only a fraction of the time a $kd$-tree requires.

## 2. Background and Related Work

Most binary coding methods generate binary codes by projecting the data on linear hyperplanes that is followed by a binarization step (*e.g.* sign function). Given a data point $\mathbf{x} \in \mathbb{R}^d$, a $k$-bit binary code is generated by a hash function $h(\mathbf{x}) \in \{+1, -1\}^k$

$$h(\mathbf{x}) = \text{sign}(\mathbf{W}^{\mathsf{T}}\mathbf{x}); \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d\times k}$ .There are two common approaches for generating the projection matrix $\mathbf{W}$; 1- **Random projection**: Elements of $\mathbf{W}$ are randomly chosen from a Gaussian distribution. This approach has been referred to as LSH in literature[2, 1, 10]. 2-**Data dependent**: The matrix $\mathbf{W}$ is learned from a set of training data to be optimized for a particular task (e.g. nearest neighbour retrieval or semantic search). There are many works on data dependent hashing [23, 6, 14, 22, 20, 24]. ITQ [6] is one of the popular methods that shows high accuracy in retrieval. It minimizes the quantization error over the training data after the dimensionality is reduced by PCA. In order to have uncorrelated bits in ITQ, the projection matrix is constrained to be a rotation matrix.

$$Q(\mathbf{R}, \mathbf{B}) = \min\|\mathbf{RPX} - \mathbf{B}\| \quad s.t. \quad \mathbf{RR}^T = \mathbf{I} \tag{2}$$

where $\mathbf{P} \in \mathbb{R}^{k\times d}$ is the projection matrix from PCA and $\mathbf{X} \in \mathbb{R}^{d\times n}$ is the data matrix, such that each column in $\mathbf{X}$ is a $d$ dimensional feature vector. $\mathbf{B} \in \{+1, -1\}^{k\times n}$ is the data in quantized form, and $\mathbf{R} \in \mathbb{R}^{k\times k}$ is the rotation matrix. The optimization is an iterative process over two steps; 1- fix $\mathbf{R}$ and solve for $\mathbf{B}$, and 2- fix $\mathbf{B}$ and solve for $\mathbf{R}$.

When $\mathbf{x}$ is a high dimensional vector, the computation time for this projection is prohibitive, since the computational complexity is $\mathcal{O}(dk)$. As discussed earlier, the computation cost for the binary projections can be very crucial in many applications. To overcome this cost [5] presents bilinear projection (BITQ) as an efficient way of generating binary codes.

$$h(\mathbf{x}) = \text{sign}(\mathbf{R}_1^T\mathbf{Z}\mathbf{R}_2) \tag{3}$$

where $\mathbf{Z} \in \mathbb{R}^{\sqrt{d}\times\sqrt{d}}$ is the reshaped version of data vector $\mathbf{x}$ ($\mathbf{vec}(\mathbf{Z}) = \mathbf{x}$) and $\mathbf{R}_1, \mathbf{R}_2 \in \mathbb{R}^{\sqrt{d}\times\sqrt{k}}$ are forced to be rotation matrices. This can be reformulated as follow:

$$h(\mathbf{x}) = \text{sign}((\mathbf{R}_1^T \otimes \mathbf{R}_2)\mathbf{x}) \tag{4}$$

where $\otimes$ is the tensor product. The computational complexity of bilinear projection in Equation 3 is $\mathcal{O}(d\sqrt{k} + \sqrt{d}k)$. Since $d >> k$, the dominant part of complexity would be $\mathcal{O}(d\sqrt{k})$. Similar to ITQ, there is an iterative process which can find the optimal $\mathbf{R}_1, \mathbf{R}_2$ to minimize quantization error.

In [25] Circulant Binary Embedding (CBE) is proposed for fast projection. The hashing function in CBE is in this form:

$$h(\mathbf{x}) = \text{sign}(\mathbf{CDx}) \tag{5}$$

where $\mathbf{C} \in \mathbb{R}^{d\times d}$ is a circulant matrix where the elements of the $i^{th}$ column of $\mathbf{C}$ can be generated by one circular shift of $(i-1)^{th}$ column. and $\mathbf{D}$ is a diagonal matrix. The operation in Equation 5 can be implemented efficiently via Fast Fourier Transform (FFT) and the computational complexity of Equation 5 will be $\mathcal{O}(d\log(d))$.

In order to achieve faster projection in both BITQ and CBE, the projection matrix is forced to have a certain structure. In BITQ the projection has to come from a tensor product of two smaller rotation matrices and in CBE the projection matrix has to be a circulant matrix. These limitations on the structure of projection matrix can be seen as a type of regularization. In this work, we present a method that learns a sparse projection matrix $\mathbf{W}$ by directly regularizing the projection matrix using $\ell_1$-norm. We can obtain very sparse projection matrices with this method, with sparsity rates of $0.1$, $0.01$ or even $0.001$, which enables fast and accurate binary projection. Our results shows that we can reach comparable accuracies in retrieval with a sparsity rate of $0.01$.

Figure 2 illustrates the computational cost of complexities $dk$, $d\log(d)$, $d\sqrt{k}$ and $\alpha dk$ with respect to the number of bits ($k$), where $\alpha$ is the sparsity rate of our proposed method. Evidently, $\alpha dk$ is more efficient with growing $d$.

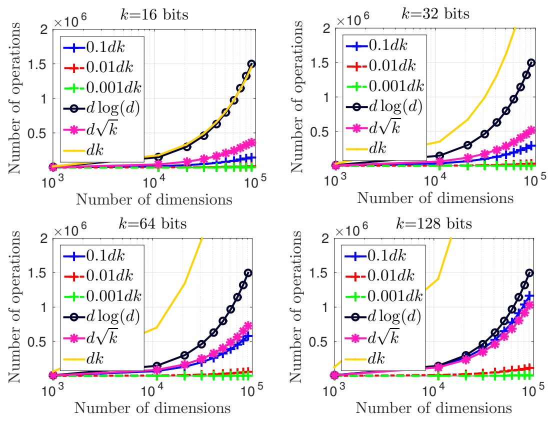

Figure 2: Complexity growth as a function of dimensions

## 3. Learning Computationally Constrained Binary Codes

Most machine learning problems can be formulated as learning of a prediction function $f_w : \mathcal{X} \rightarrow \mathcal{Y}$ which is a many-one mapping between some input space $\mathcal{X}$ and an output space $\mathcal{Y}$ that is parameterized by a parameter vector $\mathbf{w}$. Given a set of $n$ training input-output pairs $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, the conventional empirical risk minimization approach for learning the optimal parameter vector $\mathbf{w}^*$ involves solving the following optimization problem:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_{i=1}^{n} \ell(\mathbf{y}_i, f_w(\mathbf{x}_i)). \tag{6}$$

where $\ell(\mathbf{y}, \hat{\mathbf{y}})$ is a loss function that measures the discrepancy between the prediction $\hat{\mathbf{y}}$ and the ground truth output $\mathbf{y}$.

The choice of the representation of the prediction function $f_w$ and the loss function $\ell$ results in different learning algorithms. For instance, a simple but popular representation for the prediction function for binary classification problems is:

$$f_w(\mathbf{x}) = \operatorname{sign}(\mathbf{w}^\mathsf{T}\mathbf{x}) \ , \tag{7}$$

where sign is element wise sign function $\{\text{sign} : \mathbb{R} \mapsto \{+1, -1\}\}$. In many large-scale computer vision and machine learning tasks, such as image labeling or classification, the inputs are high-dimensional, *i.e.* $d$ is large. Many tasks also require multiple computations of the prediction function for a single input. For instance, this is the case for the problem of foreground-background segmentation where the task is to assign each pixel the foreground or background label by computing the predictor in a sliding window fashion. Furthermore, one may need this computation to work in real time: more than 20 frame per second. In these case, even computing the simple linear mapping defined in Equation 7 may become computationally expensive because of the large number of multiplication required to compute the dot-product between $\mathbf{w}$ and $\mathbf{x}$.

In the case of binary codes learning the parameter vector $\mathbf{w}$ is replaced by a parameter matrix $\mathbf{W}$. In this paper, we focus on binary code predictors that tie themselves to a fixed computational budget. Specifically, we consider the case where there is a limit on the number of arithmetic operations that can be performed at test time. More formally, we want to solve the computation-bounded risk minimization problem that is defined as:

$$\underset{\mathbf{W},\mathbf{b}}{\operatorname{argmin}} \quad \textstyle\sum_{i=1}^{n} \ell(\mathbf{b}_i, f_w(\mathbf{x}_i)) \tag{8}$$

$$st. \qquad \tau(f_w) \leq l \tag{9}$$

where $\mathbf{b}_i \in \{-1, 1\}^k$ is the desired binary code for $\mathbf{x}_i$ and $\tau$ measures the computation complexity of evaluating the prediction and $l$ is the required computational bound. For the case of predictor defined in Equation 7, the above problem translates to

$$\underset{\mathbf{W},\mathbf{b}}{\operatorname{argmin}} \quad \textstyle\sum_{i=1}^{n} \ell(\mathbf{b}_i, \operatorname{sign}(\mathbf{W}^\mathsf{T}\mathbf{x}_i)) \tag{10}$$

$$st. \qquad \|\mathbf{w}\|_0 \leq l \tag{11}$$

where $\|\mathbf{w}\|_0$ denotes the $\ell_0$ norm that counts the number of non-zero components of $\mathbf{W}$ since only these many multiplications are needed to evaluate the function.

In contrast to the classification problem, there are no ground truth labels for our task of binary code learning. In fact, the classifier and the labels should both be estimated, which is a complicated task. In the next subsection we explain how to learn an optimal computationally bounded function when the desired binary codes are given. In subsection 3.2, we propose a joint optimization that solves for both binary codes and the mapping function jointly.

### 3.1. Sparse Projection When Binary Codes Are Given

To improve the computation cost of $f$, a relatively straightforward idea is to make the matrix $\mathbf{W}$ sparse. When $\|\mathbf{W}\|_0 \leq l$, *i.e.* when number of non-zero entries of $\mathbf{W}$ is at most $l$, then clearly $f$ can be computed in $\mathcal{O}(l)$. However, directly solving for $\ell_0$-norm is intractable. Therefore, sparsity is often incorporated by introducing an $\ell_1$ penalty on the parameter matrix $\mathbf{W}$ followed by thresholding.

In our approach we minimize the quantization error in the original feature space, which is in essence similar to ITQ which minimizes the quantization error in the PCA space. However, we force the $\mathbf{W}$ to be a sparse matrix.

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\{\|\mathbf{W}^\mathsf{T}\mathbf{X} - \mathbf{B}\|_F + \lambda|\mathbf{W}|_{\ell_1}\} \tag{12}$$

Here $|.|_{\ell_1}$ operator on a matrix is equivalent to sum of the absolute values of the elements in that matrix. We assume that the binary codes $\mathbf{B}$ are computed as a part of training via one of the best binary coding methods (*e.g.* ITQ). It can be shown that the optimal solution for $\mathbf{W}$ can be computed independently for each column of $\mathbf{W}$. Given the matrix $\mathbf{B}$ from the output of ITQ (Equation 2), we can rewrite Equation 12 as follows:

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\{\sum_{i=1}^{k} \|\mathbf{w}^{i\mathsf{T}}\mathbf{X} - \mathbf{b}^i\|_{\ell_2} + \lambda|\mathbf{w}^i|_{\ell_1}\} \tag{13}$$

where $\mathbf{w}^i$ and $\mathbf{b}^i$ are $i^{th}$ row of $\mathbf{W}^\mathsf{T}$ and $\mathbf{B}$ respectively, and $\lambda$ is a trade-off parameter. Equation 13 is a minimization over sum of $k$ positive elements such that the parameters in each element is independent. Therefore, the global minimum is equivalent to the sum of the minimum values in each element:

$$\mathbf{W}^* = \underset{\mathbf{W}}{\operatorname{argmin}}\{\sum_{i=1}^{k} \underset{\mathbf{w}^i}{\min}\{\|\mathbf{w}^{i\mathsf{T}}\mathbf{X} - \mathbf{b}^i\|_{\ell_2} + \lambda|\mathbf{w}^i|_{\ell_1}\}\} \tag{14}$$

$$\mathbf{w}^{*i} = \underset{\mathbf{w}^i}{\operatorname{argmin}}\{\|\mathbf{w}^{i\mathsf{T}}\mathbf{X} - \mathbf{b}^i\|_{\ell_2} + \lambda|\mathbf{w}^i|_{\ell_1}\} \tag{15}$$

$$\mathbf{W}^* = \begin{bmatrix} \mathbf{w}^{*1} & \mathbf{w}^{*2} & \dots & \mathbf{w}^{*k} \end{bmatrix} \tag{16}$$

Intuitively, elements in each row of $\mathbf{B}$ can be considered a binary class label for the columns of $\mathbf{X}$. Similarly, each column of $\mathbf{W}$ can be considered a classifier that predicts the binary labels. Therefore, we can reformulate the optimization for $i^{th}$ column of $\mathbf{W}$ as a max-margin $\ell_1$ regularized linear classifier:

$$\begin{aligned} &\underset{\mathbf{w}^i}{\operatorname{argmin}} && |\mathbf{w}^i|_{\ell_1} + C\sum_{j=1}^{n} \xi_j \\ &\text{subject to} && \mathbf{b}^i(j)(\mathbf{w}^{i\mathsf{T}}\mathbf{x}_j) \geq 1 - \xi_j,\ j = 1, \dots, n. \end{aligned} \tag{17}$$

Here $\mathbf{x}_j$ is the $j^{th}$ column of $\mathbf{X}$. We solve this optimization efficiently by using LibLinear [4]. After obtaining the optimized $\mathbf{w}$, we zero out the dimensions with small absolute values to reach the sparsity rate of $\alpha$. This is done simply by sorting the absolute values in $\mathbf{w}$ and pick a threshold based on the given sparsity rate (*e.g.* if the sparsity rate is $0.1$, we zero out 90% of the smallest values). Computational complexity of obtaining a binary code for a point $\mathbf{x}$ is $\mathcal{O}(\alpha dk)$ where $\alpha$ is the sparsity rate of $\mathbf{W}$. Our experimental results show that even for very sparse projections with $\alpha = 0.01$, the accuracy drop is insignificant. As discussed earlier, for high dimensional data, generating short binary codes will lead to much less number of operations than BITQ and CBE as explained on Figure 2. In the next subsection we discuss a case, where the binary codes are not given during the training phase and we should optimize for both $\mathbf{B}$ and $\mathbf{W}$.

### 3.2. Joint Optimization

So far, binary codes were supplied in the training phase. These binary codes could be obtained from the output of any compelling binary coding method (*e.g.* ITQ). As there is no guarantee that the exact codes can be reconstructed by sparse mapping, we need to incorporate the search for binary codes into the main objective. Similar to the Equation 12, we aim to minimize the quantization error while maintaining the low $\ell_1$-norm. In contrast to Equation 12, $\mathbf{B}$ is an unknown variable.

$$(\mathbf{W}^*, \mathbf{B}^*) = \underset{\mathbf{W},\mathbf{B}}{\operatorname{argmin}}\{\|\mathbf{W}^\mathsf{T}\mathbf{X} - \mathbf{B}\|_F + \lambda|\mathbf{W}|_{\ell_1}\} \tag{18}$$

To solve this optimization problem, one might consider an EM like iterative update of variables; 1- fix $\mathbf{B}$ and solve for $\mathbf{W}$ as described in Section 3.1, 2- fix $\mathbf{W}$ and solve for $\mathbf{B}$ which is $\mathbf{B} = \operatorname{sign}(\mathbf{W}^\mathsf{T}\mathbf{X})$. However, solving this optimization does not entail a desirable solution. The reason is, there is no control over dependency of bits of binary codes, *i.e.* there is no constraint on $\mathbf{B}$. Usually, low correlation can be achieved by an orthogonality constraint over the parameters of the mapping function. In our case, it would be equivalent to have $\mathbf{W}^\mathsf{T}\mathbf{W} = \mathbf{I}$ as a constraint in the objective. However, having this constraint along with the $\ell_1$ penalty complicates the optimization problem, since orthogonality and sparsity of the mapping function are two competing constraints on minimizing the quantization error. Therefore we employ the iterative optimization procedure explained above, but this time considering only one of the constraints at each step.

To solve the optimization of Equation 18, we replace the matrix $\mathbf{B}$ with an explicit sign function of an orthogonal projection of the data as follows:

$$\begin{aligned} (\mathbf{W}^*, \mathbf{P}^*) =& \underset{\mathbf{W},\mathbf{P}}{\operatorname{argmin}}\{\|\mathbf{W}^\mathsf{T}\mathbf{X} - \operatorname{sign}(\mathbf{P}^\mathsf{T}\mathbf{X})\|_F + \lambda|\mathbf{W}|_{\ell_1}\} \\ &s.t. \quad \mathbf{P}^\mathsf{T}\mathbf{P} = \mathbf{I} \end{aligned} \tag{19}$$

where $\mathbf{P} \in \mathbb{R}^{d\times k}$ is an orthogonal matrix. This orthogonal projection ensures the low correlation between the bits, and also in contrast to Equation 18, it provides an update for binary codes which is not directly dependent on $\mathbf{W}$. Again, we employ the iterative two steps block-coordinate descend technique, where we iterate between solving for $\mathbf{W}$ and $\mathbf{P}$. When $\mathbf{P}$ is fixed, the problem would be the same as what we had in Section 3.1; the optimal $\mathbf{W}$ can be obtained via linear $\ell_1$-SVMs. When $\mathbf{W}$ is fixed, the remaining optimization is as follows (let $\mathbf{M} = \mathbf{W}^\intercal\mathbf{X}$):

$$\begin{aligned} \mathbf{P}^* &= \underset{\mathbf{P}}{\text{argmin}}\{\|\mathbf{M} - \text{sign}(\mathbf{P}^\intercal\mathbf{X})\|_F\} \\ s.t. &\quad \mathbf{P}^\intercal\mathbf{P} = \mathbf{I} \end{aligned} \tag{20}$$

The optimized solution for $\mathbf{P}$ in Equation 20 is not unique. Let's define $\mathcal{P}$ as the optimal solution set for $\mathbf{P}^*$. Solving this optimization is intractable due to the non-linearity of the sign function. Instead of solving 20, we claim that using the following optimization gives an optimal $\mathbf{P}$:

$$\begin{aligned} \mathbf{P}^{**} &= \underset{\mathbf{P}}{\text{argmin}}\{\|\,\text{sign}(\mathbf{M}) - \mathbf{P}^\intercal\mathbf{X}\|_F\} \\ s.t. &\quad \mathbf{P}^\intercal\mathbf{P} = \mathbf{I} \end{aligned} \tag{21}$$

The above optimization(21) is an orthogonal procrustes problem and has a closed form solution. $\mathbf{P}^{**} = {\mathbf{V}_{(1:k,:)}}^\intercal\mathbf{U}$ where $(\mathbf{U},\mathbf{D},\mathbf{V}) = \text{svd}(\text{sign}(\mathbf{M})\mathbf{X}^\intercal)$ [1]. It can be shown that $\mathbf{P}^{**} \in \mathcal{P}$. Lets consider a single element in $\mathbf{M}$ as $\mathbf{M}_{(i,j)}$. In 20 we would have :

$$\begin{aligned} \mathbf{P}^*_{(:,i)} &= \underset{\mathbf{P}_{(:,i)}}{\text{argmin}}\{\|\mathbf{M}_{(i,j)} - \text{sign}({\mathbf{P}_{(:,i)}}^\intercal\mathbf{X}_{(:,j)})\|\} \\ s.t. &\quad {\mathbf{P}_{(:,i)}}^\intercal\mathbf{P}_{(:,i)} = 1 \end{aligned} \tag{22}$$

if $\mathbf{M}_{(i,j)} \geq 0$, the optimal solution set

$$\mathcal{P}_{(:,i)} = \{\forall\mathbf{p} \in \mathbb{R}^d |\ \mathbf{p}^\intercal\mathbf{X}_{(:,j)} \geq 0,\ \mathbf{p}^\intercal\mathbf{p} = 1\} \tag{23}$$

The optimal solution from Equation 21 is $\mathbf{P}^{**}_{(:,i)} = \text{sign}(\mathbf{M}_{(i,j)})\frac{\mathbf{X}_{(:,j)}}{\|\mathbf{X}_{(:,j)}\|}$. Replacing $\mathbf{P}^{**}_{(:,i)}$ with $\mathbf{p}$ in Equation 23 proves that $\mathbf{P}^{**}_{(:,i)} \in \mathcal{P}_{(:,i)}$. Analogously, we can prove the same solution when $\mathbf{M}_{(i,j)} < 0$. Therefore, an optimal solution for 21 would also be an optimal solution for Equation 20.

In Algorithm 1 we show all the steps in our joint optimization for binary code learning. The joint optimization has an advantage over original ITQ formulation: in contrast to ITQ, the mapping function is directly learned over the original feature space, not on the PCA reduced space. Our experiments verify that the proposed method can outperform ITQ while being at least ten times faster.

In the next section we show an extensive set of experiments to evaluate all the parts of our proposed method in comparison to the other state-of-the-art methods.

[1]The (:) notation is the same as being used in MATLAB

**Algorithm 1:** Sparse Binary Embedding

```
Input: $\mathbf{X} \in \mathbb{R}^{d\times n}$, $k$(number of bits), $\lambda, \alpha$, $niter$.
Output: $\mathbf{W} \in \mathbb{R}^{d\times k}$, $\mathbf{B} \in \{1,-1\}^{k\times n}$.
 1: $\mathbf{P} \leftarrow random\ orthogonal\ matrix^{d\times k}$
 2: repeat
 3:     $\mathbf{L} \leftarrow \text{sign}(\mathbf{P}^\intercal\mathbf{X})$
 4:     $\mathbf{W} \leftarrow \ell_1 - \text{LinSVMs}(\text{train data} = \mathbf{X}, \text{train labels} =$
        $\mathbf{L}, \text{hyperparameter} = 1 - \lambda)$
 5:     $\mathbf{M} \leftarrow \mathbf{W}^\intercal\mathbf{X}$
 6:     $(\mathbf{U},\mathbf{D},\mathbf{V}) \leftarrow \text{svd}(\text{sign}(\mathbf{M})\mathbf{X}^\intercal)$
 7:     $\mathbf{P} \leftarrow {\mathbf{V}_{(1:k,:)}}^\intercal\mathbf{U}$
 8:     $Objective \leftarrow \|\mathbf{M} - \mathbf{L}\|$
 9: until convergence on $Objective$
10: $\mathbf{W}$ ← zero out values in columns of$\mathbf{W}$ to reach the sparsity rate $\alpha$
11: $\mathbf{B} \leftarrow \text{sign}(\mathbf{W}^\intercal\mathbf{X})$
```

## 4. Experiments

In this section, we measure the accuracy and computational cost of our method on different datasets for image retrieval. The sparsity of our mapping function also proves to be beneficial when the data is noisy. Therefore, we also present an evaluation on retrieval from noisy data.

### 4.1. Experimental Setting

**Datasets:** We consider three datasets in our experiments; ImageNet[3], GIST-1M[7], and SUN14k[16]. Each dataset is randomly partitioned into three sets: *train, query* and *base*. The parameters (mapping functions) are trained on the training set and we use each sample in the query set to find its nearest neighbor in the base set. ImageNet includes 1000 categories of images and has 1281167 images in total. We specifically used the ISLVRC2012 benchmark dataset. 100K, 1K, 1180K samples were picked for training, query and base respectively. We used CNN features extracted by Caffe [8] with 4096 dimensions. Gist1M [7] contains 1M images and their 960 dimensional GIST features. This dataset has frequently been used for approximate nearest neighbor search in the computer vision community. It has a standard partitioning of training(500K), query(100) and base(1M) sets. SUN14K[16] is a dataset of 700 categories of images, which is mainly used for scene recognition. We used this dataset because of its high dimensional features (19080 dimensions). This dataset has been used for attribute based recognition and dual-view hashing by Patterson and Hays [16] and Rastegari et al. [18]. It has 14K images and the visual features extracted by Patterson and Hays [16], which is a concatenation of GIST, PHOG and BoW. The reason that we chose these datasets are: **ImageNet:** to evaluate our method on the state-of-the-art ConvNet (CNN) features that showed great potential for recognition[9], **GIST1M:** to evaluate the accuracy of our method on a standard image retrieval dataset, and **SUN14K:** because of its feature set that allows us to evaluate the efficiency of our method on very high dimensional spaces. In order to have a fair comparison with other baselines, it is very important to pre-process the data by mean centering and normalizing them to unit hyper-sphere [25, 19].

**Methods:** To best of our knowledge, there are two other works that focus on fast binary embedding [25, 5]. Similar to [25], we use ITQ [6] as a baseline method which is not an efficient method but it gives the best accuracy among other binary embedding techniques [19]. Circulant Binary Embedding (**CBE**)[25] and Bilinear Iterative Quantization (**BITQ**) are used as competitor methods in terms of efficiency and accuracy. These methods have two versions: one with random paramaters (**CBE-rand**, **BITQ-rand**) and another one with optimized parameters (**CBE-opt, BITQ-opt**). Here we only use the optimized versions for comparison. Our method is abbreviated by (**SBE**) which stands for Sparse Binary Embedding. We distinguish between the optimization in Section 3.1 by (**SBE-B**) and the joint optimization in Section 3.2 by (**SBE-opt**). In most experiments, we measure our method with a sparsity rate of $\alpha = 0.01$, however, we also explore the sparsity rates 0.1 and 0.001 in Section 4.4. For these comparisons, we used the ITQ, BITQ and CBE software that is available online.

## 4.2. Nearest Neighbor Retrieval

In this section, we demonstrate the accuracy of the nearest neighbor retrieval task on the benchmark datasets. We present both recall rate and precision in the form of area-under-curve (AUC) (Subsection 4.4). Similar to [25], the steps of this experiment for each method is as follows: 1- Learn a model using the training set. 2- Create the binary codes for all the samples in the base set. 3-Pick one sample from query set and find its $K$-nearest neighbors in the base set by using Euclidean distance in the original features space and consider this as ground-truth nearest neighbor set. 4- Generate the binary code for the query sample and retrieve $N$ nearest neighbors from base set by calculating the Hamming distances on the binary codes. 5- Compare the retrieval from binary codes with the ground-truth retrieval and measure the recall rates. This process is repeated over all samples in the query set and the average recall rate is computed. By varying $N$ at $K = 100$, we measure recall rate at different amount of retrieval tasks and plot a curve using these values. Figure 4, 5 and 6 show the recall rate as we change the number of retrieval in different code length (bit). The curve of SBE-opt is on top of all the competitor methods most of the times. It even out-performs the method **ITQ**, which is the most accurate out of all the baseline methods.

## 4.3. Analysis of the objective in joint optimization

Here we present a comparison of the optimization functions proposed in Section 3.2. We refer to Equation 18 as the *Naive Update*. This optimization produces highly correlated bits which may have high quantization error. The new formulation presented in Equation 19 that handles the orthogonality constraints is referred to as *Orthogonal Update*. In Figure 7, the first row demonstrates the correlation between the bits of the binary codes generated from the two optimization functions. As it can be seen, in contrast to Orthogonal Update, the bits resulting from Naive Update are highly correlated. In Figure 7, the second row shows the quantization error after each iteration. The quantization error, as expected, goes up for Naive Update but for the Orthogonal Update it goes down.

Another claim was that Naive Update should converge much faster than the Orthogonal Update, since the latter has

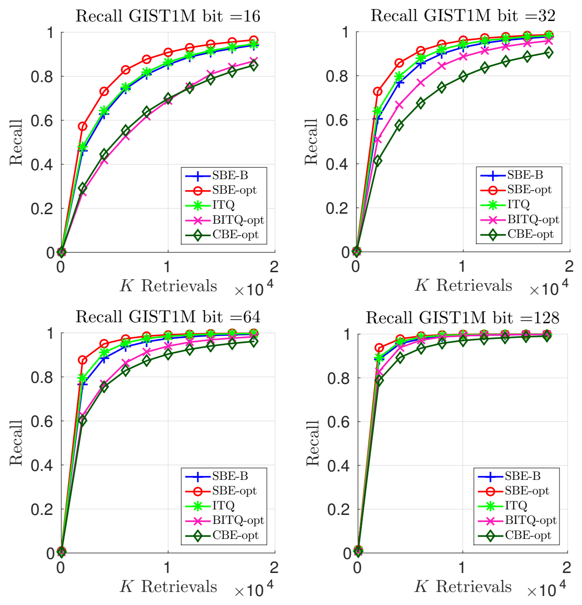

Figure 4: Recall curves for GIST1M dataset

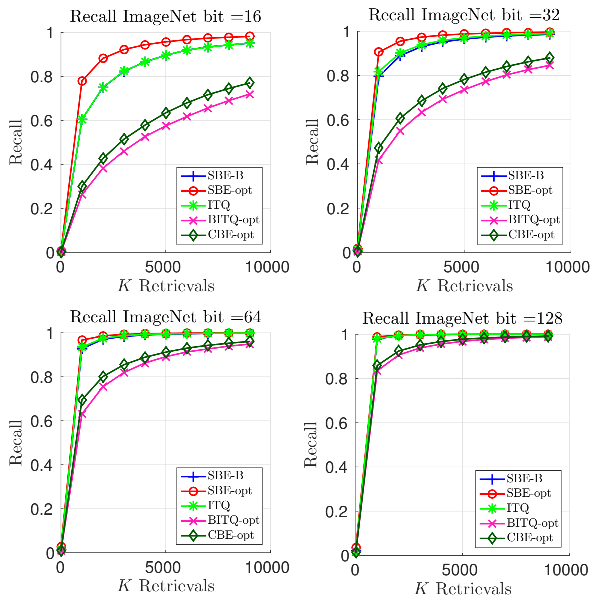

Figure 5: Recall curves ImgeNet dataset

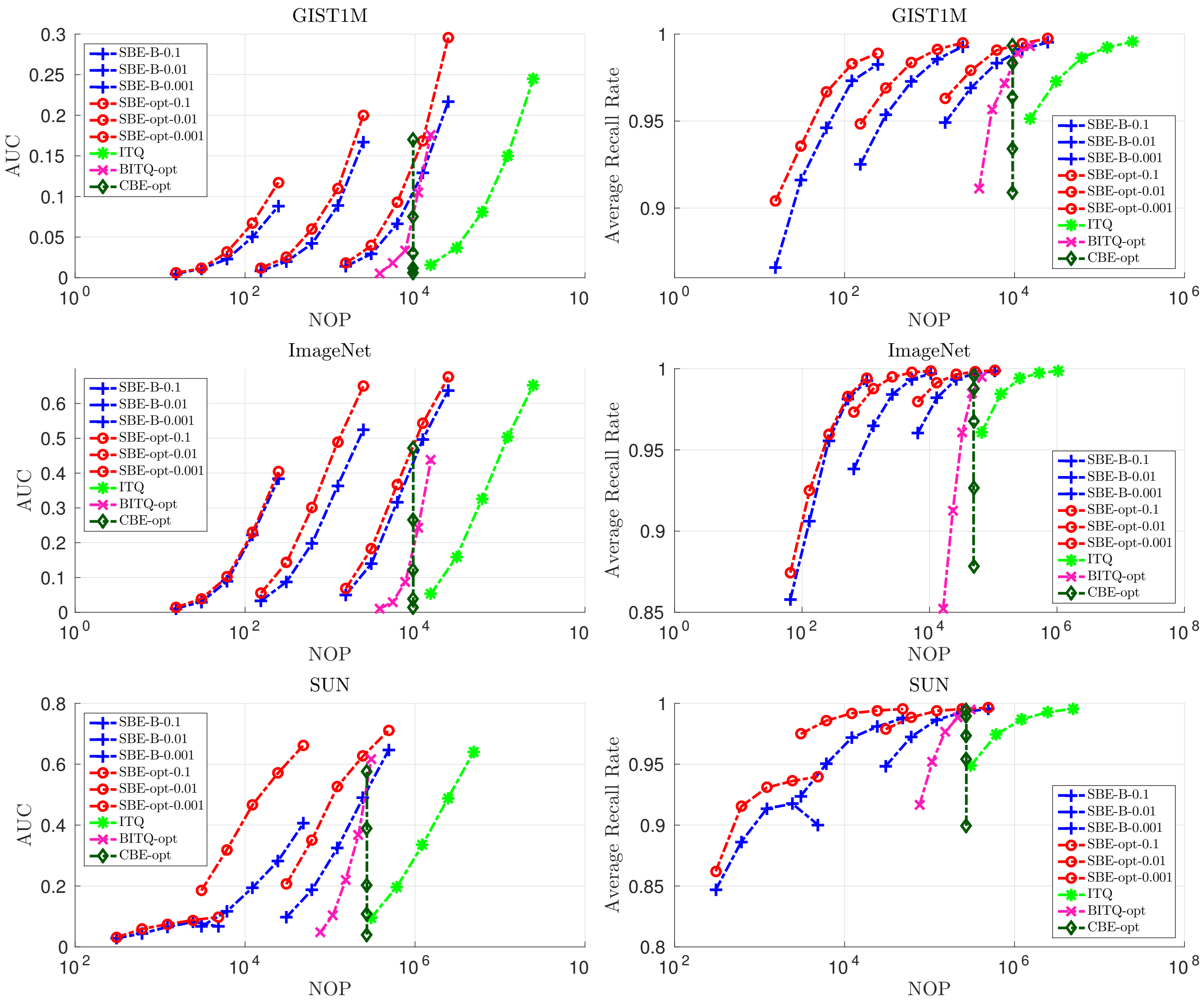

Figure 3: **Number of operations vs. Accuracy (AUC, Recall ):** In this plot each curve represents a method and each point on the curve corresponds to a code length (number of bits) which change from left to right as follows: [16, 32 64, 128, 256]

two competing constraints. This is shown in Figure 7, third row.

## 4.4. Coding Efficiency vs. Accuracy

In this section we compare the efficiency of our method with other competitor methods. In order to present a fair comparison, instead of reporting the running time, we report the number of arithmetic operations (**NOP**) used for each method. Figure 3 shows an evaluation over NOP and accuracy which is measured by AUC and average recall rate. Each curve in Figure 3 depict one method and each point on the curve corresponds to a code length (number of bits) which is in the range of [16, 32, 64, 128, 256]. Evidently, in ImageNet SBE can achieve comparable accuracy with 128 bits while being 1000 times faster than other methods. The dark green curve corresponds to CBE is growing vertically in the plot. This is because the complexity of CBE $d\log(d)$ is independent of the number of bits and only depends on the number of dimensions in the original feature space.

## 4.5. Retrieval Running Time Analysis

Since the main application for our fast binary encoding method is retrieval, we compared it with $k$d-tree as a baseline for a nearest neighbor retrieval task. We used multiple

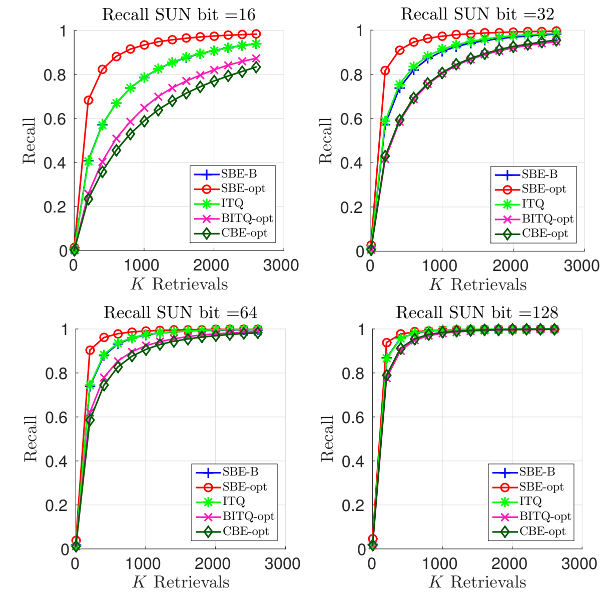

Figure 6: Recall curves for SUN dataset

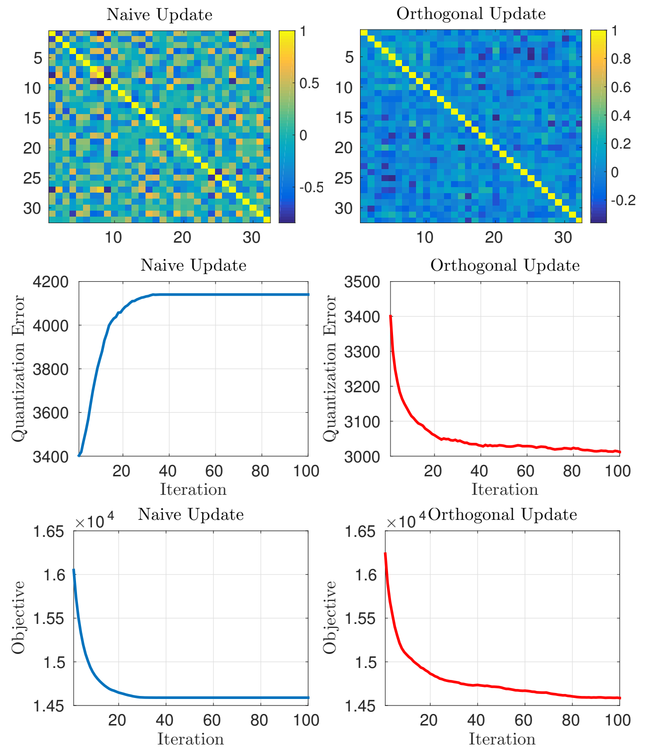

Figure 7: Analysis on the optimization of the objective function on sparse binary coding

index hashing as a retrieval technique on top of our binary codes. For each eight consequent bits, we generate a hash table to index the samples in the database. At query time, we perform $\frac{b}{8}$ look-ups. $b$ is the number of bits in our binary code. We score the collected indexes by counting the number of time that they have been retrieved from the hash tables. Then, we pick the $k$ indices with the highest score as the $k$ nearest neighbors. We vary the number bits by [16, 32, 64, 128, 256] and compute the AUC and running time. In $k$d-tree we vary the bucket size by [$2^2$, $2^4$, $2^8$, $2^{10}$, $2^{12}$] and compute the AUC and running time. The results on GIST1M are depicted in Figure 1. We used the built in $k$d-tree toolbox in MATLAB for this comparison.

## 4.6. Retrieval from Noisy Data

The sparsity of the projection matrix in our method makes it possible to have accurate retrieval with noisy data. In Figure 8 we show the performance of our method in comparison with others on the noisy data in ImageNet. Noisy data is created by randomly selecting some dimensions of each sample and change their values to either 0 , 1 or -1. As it can be seen our method can reach higher accuracy compared to ITQ with high noise ratio.

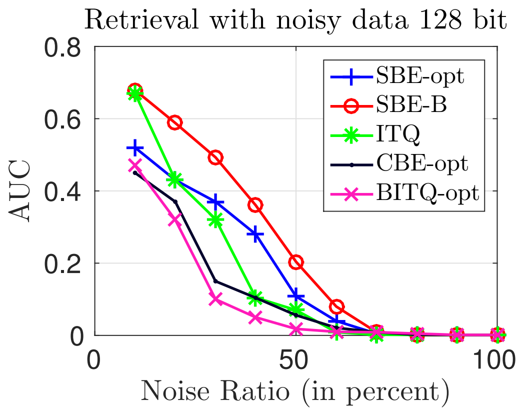

Figure 8: Retrieval with noisy feature on ImageNet

## 5. Conclusion

We proposed a fast technique for binary encoding that can be used for efficient high dimensional image retrieval tasks. Our method uses a fixed computational budget to learn a projection matrix that generates binary codes. We proposed a novel joint optimization and presented a proof of convergence. We evaluated our technique for image retrieval tasks using three different datasets and compared the results with the state-of-the-art methods. Our method consistently shows on-par or higher accuracy, and much faster running time than other methods. We also compared our method with the well-known $k$d-tree algorithm for an image retrieval task.

As future work, we will focus on conditioning our sparse hash functions on each other in the form of a tree to reduce the complexity of the classification problem at each step.

## References

[1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Foundations of Computer Science, 2006. FOCS'06. 47th Annual IEEE Symposium on*, pages 459–468. IEEE, 2006. 2

[2] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the Twentieth Annual Symposium on Computational Geometry*, SCG '04, 2004. 1, 2

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009. 5

[4] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *The Journal of Machine Learning Research*, 9:1871–1874, 2008. 4

[5] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 484–491. IEEE, 2013. 2, 6

[6] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Proceedings of the 2011 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '11, 2011. 2, 6

[7] H. Jégou, M. Douze, C. Schmid, et al. Searching with quantization: approximate nearest neighbor search using short codes and distance estimators. 2009. 5

[8] 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 ACM International Conference on Multimedia*, pages 675–678. ACM, 2014. 5

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 5

[10] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2130–2137. IEEE, 2009. 2

[11] S. Maji, A. C. Berg, and J. Malik. Efficient classification for additive kernel svms. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2013. 2

[12] M. Muja and D. G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV)*, pages 404–410, 2012. 1

[13] M. Muja and D. G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014. 1

[14] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, 2011. 2

[15] M. Norouzi and A. Pournaji. Fast search in hamming space with multi-index hashing. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, 2012. 2

[16] G. Patterson and J. Hays. Sun attribute database: Discovering, annotating, and recognizing scene attributes. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2751–2758. IEEE, 2012. 5

[17] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1986. 1

[18] M. Rastegari, J. Choi, S. Fakhraei, D. Hal, and L. Davis. Predictable dual-view hashing. In *Proceedings of The 30th International Conference on Machine Learning*, pages 1328–1336, 2013. 5

[19] M. Rastegari, S. Fakhraei, J. Choi, D. W. Jacobs, and L. S. Davis. Comparing apples to apples in the evaluation of binary coding methods. *CoRR*, 2014. 6

[20] M. Rastegari, A. Farhadi, and D. A. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV (6)*, 2012. 2

[21] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. *Pattern Analysis and Machine Intellingence*, 34(3), 2011. 2

[22] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3424–3431. IEEE, 2010. 2

[23] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *Computer Vision - ECCV 2012 - 12th European Conference on Computer Vision, Florence, Italy, October 7-13, 2012, Proceedings, Part V*, 2012. 2

[24] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *Computer Vision–ECCV 2012*, pages 340–353. Springer, 2012. 2

[25] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. *arXiv preprint arXiv:1405.3162*, 2014. 2, 6

[26] L. Zheng, S. Wang, and Q. Tian. Coupled binary embedding for large-scale image retrieval. *Image Processing, IEEE Transactions on*, 2014. 1