Sparse Projections for High-Dimensional Binary Codes

Yan Xia[1*] Kaiming He[2] Pushmeet Kohli[2] Jian Sun[2]

[1]University of Science and Technology of China
[2]Microsoft Research

Abstract

This paper addresses the problem of learning long binary codes from high-dimensional data. We observe that two key challenges arise while learning and using long binary codes: (1) lack of an effective regularizer for the learned high-dimensional mapping and (2) high computational cost for computing long codes. In this paper, we overcome both these problems by introducing a sparsity encouraging regularizer that reduces the effective number of parameters involved in the learned projection operator. This regularizer not only reduces overfitting but, due to the sparse nature of the projection matrix, also leads to a dramatic reduction in the computational cost. To evaluate the effectiveness of our method, we analyze its performance on the problems of nearest neighbour search, image retrieval and image classification. Experiments on a number of challenging datasets show that our method leads to better accuracy than dense projections (ITQ [11] and LSH [16]) with the same code lengths, and meanwhile is over an order of magnitude faster. Furthermore, our method is also more accurate and faster than other recently proposed methods for speeding up high-dimensional binary encoding.

1. Introduction

Learning efficient representations of data is an important task for many fields of computer science, such as computer vision and bio-informatics that work with high dimensional inputs. In computer vision, methods for generating binary representations have been successfully used for problems like image retrieval [31, 21, 11], image classification [28, 10, 32], and descriptor matching [17]. While methods for binary encoding have yielded good results, the binary codes were kept short (tens of bits) [31, 21, 11] due to computational and algorithmic reasons.

Recent work on representation learning using deep neural networks (DNN) [20, 7, 13] has shown that features of thousands of dimensions or even more are useful for various recognition tasks such as object classification and image retrieval. Unlike traditional hand-crafted features [24, 25, 19, 26], these high-dimensional features are learned from large-scale data [27] and thus are not structured. On the other hand, it has been noticed that for input signals of dimensionality d, the code-length b of the binary code required to achieve reasonable accuracy (compared with no encoding) is usually $O(d)$ [28, 10, 32]. Based on these reasons, we consider the problem of learning long binary codes for high-dimensional arbitrary input signals.

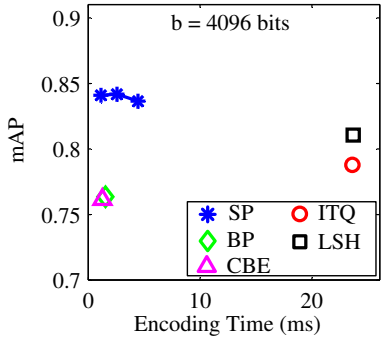


Figure 1. ANN accuracy *vs.* encoding time on a dataset of one million 4096-d DNN features. Our sparse projection (SP) method, with 5%, 10%, and 15% non-zero elements, is more accurate and about 10× faster than ITQ [11] and LSH [16], and is more accurate than BP [10] and CBE [32]. More details are in Figure. 2.

We observe that there are two key challenges in learning and using long binary codes: (1) lack of an effective regularizer for the learned high-dimensional mapping and (2) expensive cost for computing the long codes. Consider a simple and most popular way of generating a binary code: project the d-dimensional data by a b-by-d matrix and then binarize by thresholding. When d is large and $b \sim O(d)$, the projection could involve millions or even billions of parameters. A model in this large scale can easily tend to overfit the data, if without an effective regularizer. It is also computationally expensive in terms of both time and memory.

To overcome these problems, in this paper we incorporate a sparse regularizer in an objective function which learns the projection matrix. This regularizer restricts the

*This work is done when Yan Xia is an intern at Microsoft Research.

the number of non-zero coefficients in the projection matrix. In doing so it reduces the effective number of parameters and reduces overfitting. In addition, the sparsity leads to a dramatic reduction in computational cost.

Comprehensive experiments show that our method not only leads to better accuracy than competitive dense projection methods (Iterative Quantization (ITQ) [11] and Locality-Sensitive Hashing (LSH) [16]) for high-dimensional data with the same code lengths, but is also over one order of magnitude faster. Our method is more accurate than two recent methods (bilinear projections (BP) [10] and circulant binary embedding (CBE) [32]) that are designed for fast high-dimensional binary encoding. Fig. 1 shows a comparison on 4096-d DNN features.

It has traditionally been thought that the ITQ encoding method is only applicable when the length of the desired code is smaller than the dimensionality of the original signal, *i.e.*, $b \leq d$. As a by-product of our work, we show that our method and ITQ [11] can naturally be applied for generating binary codes whose code lengths are *larger* than the data dimensionality ($b > d$). We find that the "procrustes solution" [29, 12], which is a key optimization technique for orthogonal problems, is valid for $b \geq d$. So both our method and ITQ can be simply applied to generate longer codes. The accuracy of both our method and ITQ increases with code lengths in experiments.

2. Related Work

A number of recent studies have considered the problem of high-dimensional binary encoding [10, 32]. In the bilinear projections for binary coding [10], a data vector is projected by two smaller matrices instead of a single large matrix, based on the assumption that the data vectors are formulated by reshaping matrices. This assumption is valid for many hand-crafted and structured features like SIFT [24], GIST [25], VLAD [19], and Fisher Vectors [26], but is not true for learned features such as those learned by DNNs [20, 7]. In circulant binary embedding [32], a circulant matrix is introduced to enable computation in a way like Fast Fourier Transform. Though both [10] and [32] have shown promising speedup, their accuracy is inferior to dense projections (like ITQ [11]) using the same code length.

3. Sparse Projections for Binary Encoding

We first introduce our objective function and regularizer for binary encoding. We use $\mathbf{x} \in \mathbb{R}^d$ to denote a data point, and $\mathrm{X} \in \mathbb{R}^{d \times n}$ to denote a matrix whose each column is a datum. Our target is to learn a projection matrix $\mathrm{R} \in \mathbb{R}^{b \times d}$ for producing b-bit binary codes by $\mathrm{B} = \mathrm{sign}(\mathrm{RX}) \in \{-1, 1\}^{b \times n}$. A widely considered objective function for binary encoding involves minimizing the distortion, which is adopted in ITQ [11] and its variants [10, 32]. More formally, ITQ solves the following problem:

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

Here $\|\cdot\|_{\mathrm{F}}$ denotes the Frobenius norm (the sum of the squares of matrix elements), and I is a d-by-d identity matrix. The above problem is only regularized by the orthogonality constraint $\mathrm{R}^{\mathrm{T}}\mathrm{R} = \mathrm{I}$, which is a weak regularizer.

To further regularize the projection of high-dimensional data, we introduce a sparsity constraint. Our objective function is:

$$\min_{\mathrm{R,B}} \|\mathrm{RX} - \mathrm{B}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \mathrm{R}^{\mathrm{T}}\mathrm{R} = \mathrm{I}, \quad \text{and} \quad |\mathrm{R}|_0 \leq m. \tag{2}$$

Here $|\cdot|_0$ denotes the number of non-zero elements of the matrix[1], and m is a parameter that directly controls the sparsity of the projection matrix R. This is known as the m-sparsity constraint [3]. Note that this problem is not convex due to the orthogonal and the l_0 constraints.

Another possible alternative to above l_0-regularization is l_1-regularization: $|\mathrm{R}|_1 \leq m$, where $|\cdot|_1$ denotes the sum of the magnitudes of the elements. We choose l_0-regularization mainly due to two reasons. First, the l_0-regularization directly controls the number of non-zero elements and so the time/memory complexity of the projection, while the l_1 case does not have this easy control. Second, as we will show, the l_0-regularization leads to a simple hard-thresholding solver during the optimization (Sec. 4.1). So we focus on l_0-regularization in this paper.

4. Optimization

The sparsity constraint in (2) makes it challenging to solve for R directly. To find a feasible solution, we adopt the variable-splitting and penalty techniques in optimization [5, 30]. We introduce an auxiliary variable $\bar{\mathrm{R}}$. We put the orthogonality constraint on $\bar{\mathrm{R}}$ and put the sparse constraint on R, and meanwhile penalize the difference between $\bar{\mathrm{R}}\mathrm{X}$ and RX. With this idea, we relax the problem as the following form:

$$\min_{\mathrm{R},\bar{\mathrm{R}},\mathrm{B}} \|\bar{\mathrm{R}}\mathrm{X} - \mathrm{B}\|_{\mathrm{F}}^2 + \beta\|\bar{\mathrm{R}}\mathrm{X} - \mathrm{RX}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \bar{\mathrm{R}}^{\mathrm{T}}\bar{\mathrm{R}} = \mathrm{I}, \quad \text{and} \quad |\mathrm{R}|_0 \leq m, \tag{3}$$

where β is a penalty weight. Relaxation in this way is similar to Half-Quadratic Splitting [30]. By introducing an auxiliary variable, the original problem can be separated into feasible sub-problems, and the solution to (3) will converge to that of (2) when $\beta \to \infty$ [30]. We solve (3) in an alternating way: updating one variable with others fixed.

[1] More strictly, $|\mathrm{R}|_0$ should be written as $|\mathrm{vec}(\mathrm{R})|_0$. We use $|\mathrm{R}|_0$ for the ease of presentation.

4.1. Fix B and $\bar{\mathrm{R}}$, update R.

This sub-problem is:

$$\min_{\mathrm{R}} \|\mathrm{RX} - \mathrm{Z}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad |\mathrm{R}|_0 \leq m, \tag{4}$$

where $\mathrm{Z} = \bar{\mathrm{R}}\mathrm{X}$ is fixed. This sub-problem looks like a sparse coding problem under the m-sparsity constraint [3]. But in commonly studied sparse coding problems [4, 23, 3] the variable R is a vector and RX is a vector-matrix multiplication. In the following we develop a solution to the matrix variable R used here.

Instead of directly optimizing (4), we solve this problem:

$$\min_{\mathrm{R},\mathrm{S}} \phi(\mathrm{R}, \mathrm{S}) \tag{5}$$

where the new objective function ϕ is defined as:

$$\phi(\mathrm{R}, \mathrm{S}) = \|\mathrm{RX} - \mathrm{Z}\|_{\mathrm{F}}^2 + \|\mathrm{X}\|_{\mathrm{F}}^2\|\mathrm{R} - \mathrm{S}\|_{\mathrm{F}}^2 - \|\mathrm{RX} - \mathrm{SX}\|_{\mathrm{F}}^2, \tag{6}$$

and S is a matrix of the same size as R. Due to the sub-multiplicativity of Frobenius norm [15], the relation $\|\mathrm{X}\|_{\mathrm{F}}^2\|\mathrm{R} - \mathrm{S}\|_{\mathrm{F}}^2 \geq \|\mathrm{RX} - \mathrm{SX}\|^2$ always holds, and thus $\phi(\mathrm{R}, \mathrm{S})$ is no less than the objective function in (4) for any S. So the solution to (5) gives the solution to (4). ϕ is known as a surrogate objective function [22]. We minimize (5) by an alternating algorithm as in [22]: fixing S and solving for R, and vice versa.

(i) Fix S, update R. We expand ϕ and rewrite it as:

$$\phi(\mathrm{R}) = \|\mathrm{X}\|_{\mathrm{F}}^2 \mathrm{tr}(\mathrm{R}^{\mathrm{T}}\mathrm{R}) - 2\mathrm{tr}(\mathrm{Q}^{\mathrm{T}}\mathrm{R}) + const, \tag{7}$$

where $\mathrm{Q} = \mathrm{ZX}^{\mathrm{T}} + \|\mathrm{X}\|_{\mathrm{F}}^2\mathrm{S} - \mathrm{SXX}^{\mathrm{T}}$ and $\mathrm{tr}(\cdot)$ denotes the trace. The constant does not depend on R. Based on the definition of trace, we have an objective function in vector forms:

$$\min_{\mathrm{R}} \|\mathrm{X}\|_{\mathrm{F}}^2 \mathbf{r}^{\mathrm{T}}\mathbf{r} - 2\mathbf{q}^{\mathrm{T}}\mathbf{r}, \quad \text{s.t.} \quad |\mathbf{r}|_0 \leq m \tag{8}$$

where the matrices R and Q are reshaped into vectors $\mathbf{r}$ and $\mathbf{q}$. This problem can be easily solved by $\mathbf{r} = thr_m\left(\mathbf{q}/\|\mathrm{X}\|_{\mathrm{F}}^2\right)$, where thr_m is an operator that keeps the largest (in magnitude) m entries and sets the rest as zero. In matrix form the solution is:

$$\mathrm{R} = thr_m\left(\mathrm{S} + \frac{1}{\|\mathrm{X}\|_{\mathrm{F}}^2}(\bar{\mathrm{R}} - \mathrm{S})\mathrm{XX}^{\mathrm{T}}\right), \tag{9}$$

where we have substituted $\mathbf{q}$ and Z by their definitions, and thr_m keeps the largest m entries in the matrix.

(ii) Fix R, update S. Based on (6), because $\|\mathrm{X}\|_{\mathrm{F}}^2\|\mathrm{R} - \mathrm{S}\|_{\mathrm{F}}^2 - \|\mathrm{RX} - \mathrm{SX}\|_{\mathrm{F}}^2 \geq 0$ and the equality is satisfied when $\mathrm{S} = \mathrm{R}$, the solution to this sub-problem is $\mathrm{S} = \mathrm{R}$.

Combining **(i)** and **(ii)**, we obtain an iterative solution to the problem (4):

$$\mathrm{R}_{t+1} = thr_m\left(\mathrm{R}_t + \frac{1}{\|\mathrm{X}\|_{\mathrm{F}}^2}(\bar{\mathrm{R}} - \mathrm{R}_t)\mathrm{XX}^{\mathrm{T}}\right), \tag{10}$$

where t is the iteration index. We run 30 iterations in our algorithm. The solution in this form is known as Iterative Hard Thresholding [3]. But the algorithm in [3] is developed for a vector variable, and here is for a matrix variable. Following [3], we can further prove that the iterative solution in (10) converges to a local minimum.

A natural initialization to (10) is to let $\mathrm{R}_0 = \bar{\mathrm{R}}$. Then the first iteration of (10) gives a simple result as:

$$\mathrm{R} = thr_m\left(\bar{\mathrm{R}}\right). \tag{11}$$

This one-step solution has a clear intuition: the projection matrix R is updated by hard-thresholding the auxiliary matrix $\bar{\mathrm{R}}$, keeping its largest m entries. In experiments, we will show results using either the iterative solution (10) or the one-step solution (11). We find that the one-step solution gives comparable accuracy, but is much faster for training.

4.2. Fix B and R, update $\bar{\mathrm{R}}$.

With R fixed, the sparsity constraint is ignored in this sub-problem. The two terms in (3) are both quadratic on $\bar{\mathrm{R}}$, so the problem can be shown as equivalent to:

$$\min_{\bar{\mathrm{R}}} \|\bar{\mathrm{R}}\mathrm{X} - \mathrm{Y}\|_{\mathrm{F}}^2 \quad \text{s.t.} \quad \bar{\mathrm{R}}^{\mathrm{T}}\bar{\mathrm{R}} = \mathrm{I}, \tag{12}$$

where $\mathrm{Y} = (\mathrm{B} + \beta\mathrm{RX})/(1 + \beta)$ is fixed. This problem is known as the orthogonal procrustes problem [29, 12] and is recently widely involved in encoding [11, 9].

This procrustes problem is solvable if $b \geq d$, according to [12] as follows. First compute the SVD of the matrix XY^{T} as $\mathrm{XY}^{\mathrm{T}} = \mathrm{U}\Sigma\mathrm{V}^{\mathrm{T}}$, where U is a d-by-d orthogonal matrix, Σ is a d-by-d diagonal matrix, and V is a b-by-d column-orthogonal matrix. Then let $\bar{\mathrm{R}} = \mathrm{VU}^{\mathrm{T}}$.

It is worth noting that according to modern studies of procrustes problems [12], the above solution is valid for all $b \geq d$ (see Chapter 5.7 of [12]), *i.e.*, the code length is *larger* than the data dimensionality. Thus our method and ITQ are both naturally applicable for generating *higher*-dimensional binary codes ($b \geq d$). There have been limited understandings[2] that ITQ is applicable only when $b \leq d$, but we find this is not true. In the experiment section, we will show the results of ITQ and our method even when $b \geq d$.

On the contrary, the procrustes solution [12] is not valid when $b < d$. Actually, this is because $\bar{\mathrm{R}}^{\mathrm{T}}\bar{\mathrm{R}} = \mathrm{I}$ is not a valid constraint if $b < d$, because $\mathrm{rank}(\bar{\mathrm{R}}^{\mathrm{T}}\bar{\mathrm{R}}) \leq \min(b, d)$

[2] In [11], it states that ITQ "*cannot use more bits than the original dimension of the data*".

while rank(I) is d. To handle the case of $b < d$ in our solver, we reduce X to b-dimensional by $\mathrm{X}' = \mathrm{P}\mathrm{X}$, where P is a b-by-d PCA projection matrix corresponding to the largest eigenvalues. Then we use X' in place of X in the sub-problem (12) for solving a b-by-b matrix $\bar{\mathrm{R}}'$. Then $\bar{\mathrm{R}}$ is given by $\bar{\mathrm{R}}'\mathrm{P}$. Note we do not pre-project the data X by PCA in the main problem (3), because the PCA projection matrix P is a dense matrix, so using P as pre-processing will ruin the sparsity of the total projection.

4.3. Fix R and $\bar{\mathrm{R}}$, update B.

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

4.4. Algorithm Summary

We iteratively solve the three sub-problems as in Sec. 4.1-4.3. We pre-process the data by subtracting their mean (but no projection). We initialize $\mathrm{R} = \bar{\mathrm{R}}$ by a random orthogonal matrix, and start from the step of updating B. The algorithm is run 50 iterations and finally a sparse matrix R is produced for projecting data.

In theory, we should start from a small β in (3) and gradually increase it to infinity [30]. But in experiments we find that simply using a fixed β can lead to comparable accuracy, and the accuracy is insensitive to the choice of fixed β (we tried 0.1 to 100). So we fix $\beta = 1$ for simplicity for all experiments in this paper.

5. Experiments

We conduct experiments on three tasks: approximate nearest-neighbor (ANN) search, image retrieval, and image classification. We refer to our method as 'sparse projections (**SP**)' for binary codes, and compare it with the following methods:

- Iterative Quantization (**ITQ**) [11]: this is one of the state-of-the-art binary encoding methods. In the original paper [11], ITQ is only applicable when $b \leq d$. We generalize it to higher-dimensional cases ($b > d$) using the procrustes solution described in Sec. 4.2.

- Locality-Sensitive Hashing (**LSH**) [16, 6, 2]: this method simply uses a random projection matrix. Despite its simplicity, it has been proven [2] that the generated Hamming distance asymptotically approaches the Euclidean distance for codes that have sufficiently large lengths. LSH has been shown [11, 10, 32] to have competitive performance when $b \sim O(d)$.

- Bilinear Projections (**BP**) [10] and Circulant Binary Embedding (**CBE**) [32]: these methods are designed to speed-up projections for high-dimensional data. They are generally only applicable for the case when $b \leq d$ as their objective functions involve terms like ITQ. However, we are unaware of how to generalize the procrustes solution of $b > d$ to these methods, so we can only show their performance in the case $b \leq d$.

We use the implementations of ITQ, BP and CBE that were released by the authors. The training step of all binary encoding methods is run in Matlab, while the evaluation (encoding and ranking) is implemented in C++. All experiments are on a server with an Intel Xeon E5-2650 CPU (2.00Ghz) and 128 GB memory. The running time is evaluated using single-thread implementation.

We have publicly released our Matlab code[3].

5.1. Approximate Nearest Neighbor Search

Experiments on DNN features

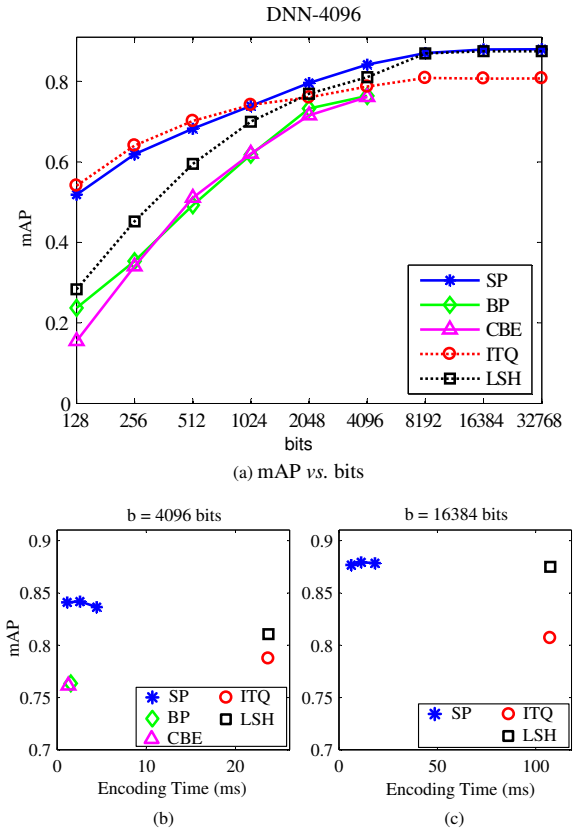
Recent researches have demonstrated the effectiveness of the use of deep learning features as image representations [20, 33]. We first run experiments on such features. We constructed a dataset that contains one million images which are randomly sampled from the recently published dataset of one hundred million Flickr images[4]. We used a CNN model to extract image-wise features for images in the dataset. Zeiler and Fergus's (ZF) [33] "fast" model, which contains five convolutional layers and two 4096-d fully-connected (fc) layers, is used in our experiments. We train this model on ImageNet 2012 [27], following the common guidelines [20]. Using this network, we extract 4096-d outputs of the second fc layer as image features. Each image is resized so that its smaller dimension is 256, and the center 224×224 region is used to compute features. We refer to this dataset as **DNN-4096**. An extra 1000 random samples are used as queries. Note that each 4096-d raw feature (floating number) requires 16384 bytes (131,072 bits).

For comparison, we adopt the evaluation protocol used by authors of ITQ [11]. The average distance to the 50th nearest neighbors of images in the dataset is used to define the true positive samples of a query. Given a query, images in the dataset are ranked according to their Hamming distances to the query, based on their binary codes. This is known as Hamming ranking. We then evaluate the mean Average Precision (mAP), *i.e.*, the mean area under the precision-recall curve.

In Fig. 2 (a), we show how mAP changes with code length b. This is an important comparison as code length b directly determines the memory (storage) requirement of the database and also the time taken to compute the Hamming ranking. However, the time taken for computing the

[3] http://research.microsoft.com/en-us/um/people/kahe/cvpr15spbe

[4] http://labs.yahoo.com/news/yfcc100m/.



(a) mAP *vs.* bits

(b) (c)

Figure 2. Comparisons on the **DNN-4096** dataset. (a) mAP *vs.* the number of bits. In this sub-figure, SP has 10% non-zero elements in its projection matrix. (b, c) mAP *vs.* encoding time when code length is fixed to 4096 and 16384 bits respectively. SP in these two sub-figures are with 5%, 10%, and 15% non-zero elements. Note that each 4096-d raw feature (float) requires 16384 bytes (**131,072** bits). The database of raw features requires ∼16GB memory, and the database of 4096-bit codes requires only ∼500 MB.

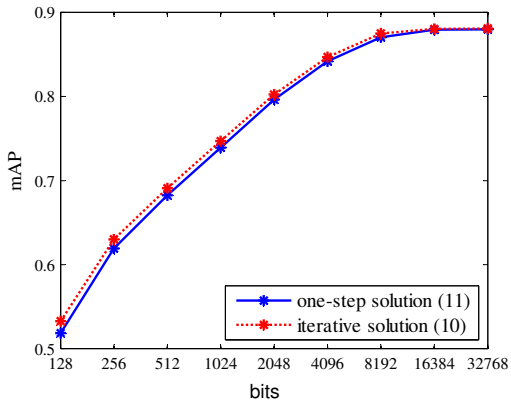


Figure 3. Comparison between the one-step solution (11) and the iterative solution (10) in optimizing the problem of Sec. 4.1. Note that the entire solution is still iterative in either case. We show the results when there are 10% non-zero elements in the projection matrix. The mAP is evaluated on the DNN-4096 dataset.

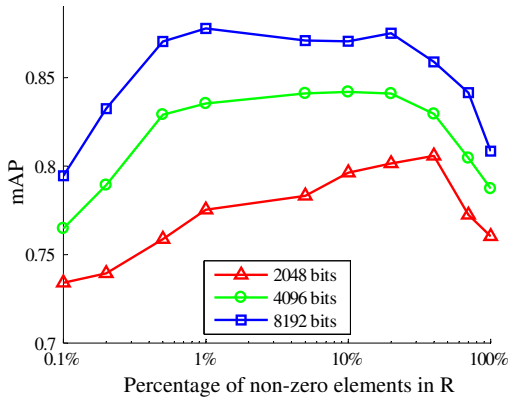


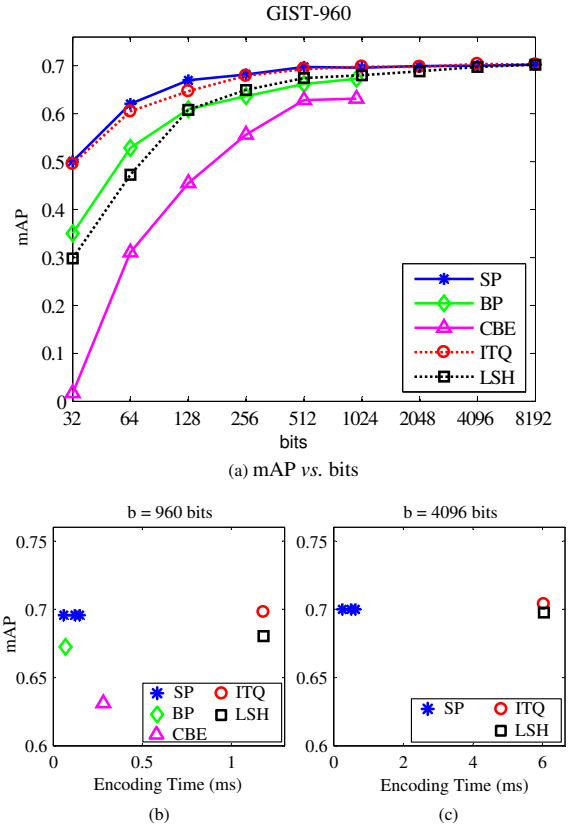
Figure 4. The mAP of our SP method with different sparsity, defined as the percentage of non-zero elements in the projection matrix. This experiment is evaluated on the DNN-4096 dataset. To the extreme when 100% elements are non-zero, the method degrades to ITQ.

binary code, as we will discuss later, does not only depend on b. In Fig. 2 (a), the SP method has 10% non-zero elements in the projection matrix. This figure shows that our SP method is considerably more accurate than BP and CBE, and is able to generate longer codes ($b > d$). Our method also significantly outperforms ITQ on long codes (*e.g.*, $b > 1024$), and is slightly worse on the short codes. Note that longer codes in general have better accuracy, so can be more useful in practice. On the other hand, our method consistently outperforms LSH, except that both methods perform nearly the same when $b > 1024$. Since LSH is guaranteed to make the Hamming distance asymptotically approach the Euclidean distance, this results shows that our method is also very accurate.

Though ITQ and LSH are competitively accurate in Fig. 2 (a), they lead to computationally expensive coding operations. We show the mAP *vs*. encoding time (per query) using 4096 bits (Fig. 2 (b)) and 16384 bits (Fig. 2 (c)). Here SP has 5%, 10%, and 15% non-zero elements in the projection matrix. This means SP is 20×, 10×, and 6.7× faster than ITQ and LSH for encoding, while is still more accurate. In Fig. 2 (b), BP and CBE can achieve speedup ratios similar to SP, but have poorer accuracy. BP and CBE are unavailable for the longer codes ($b > d$) in Fig. 2 (c).

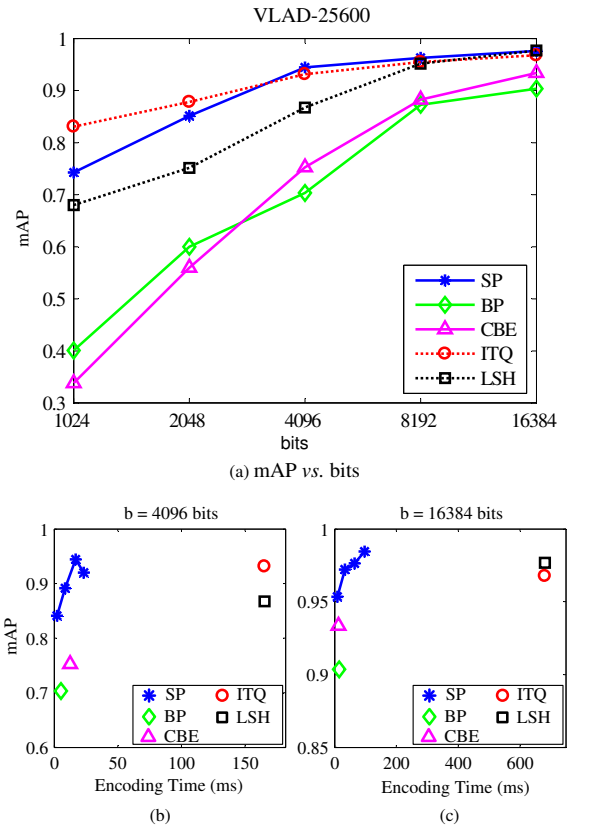
In Fig. 3, we show the results of SP using the one-step solution (11) or the iterative solution (10) for the problem in Sec. 4.1. Fig. 3 shows that they produce comparable results. But the one-step solution is much faster. Using the one-step solution, SP has the same training complexity as the ITQ algorithm (as in Sec. 4.1 and 4.2), and the iterative solution is 30 times slower for training. In the rest part of this paper, we report results using the one-step solution.

To investigate the impact of sparsity to accuracy, in Fig. 4 we show mAP *vs*. the number of non-zero elements in the



(a) mAP *vs.* bits (b) (c)

Figure 5. Comparisons on the **GIST-960** dataset. (a) mAP *vs.* the number of bits. In this sub-figure, SP has 10% non-zero elements in its projection matrix. (b, c) mAP *vs.* encoding time when code length is fixed to 960 and 4096 bits respectively. SP in (b,c) has 5%, 10%, and 15% non-zero elements.



(a) mAP *vs.* bits (b) (c)

Figure 6. Comparisons on the **VLAD-25600** dataset. (a) mAP *vs.* the number of bits. In this sub-figure, SP has 10% non-zero elements in its projection matrix. (b, c) mAP *vs.* encoding time when code length is fixed to 4096 and 16384 bits respectively. SP in (b,c) has 1%, 5%, 10%, and 15% non-zero elements.

projection matrix. In the case of all non-zero coefficients ("100%"), SP becomes equivalent to ITQ. We see that there are some "sweet points" near the sparsity of 10% non-zero elements. This indicates a dense matrix can overfit the data, and the sparsity constraint is an effective regularizer. Actually, a dense matrix in this figure has millions or hundreds of millions of parameters and results in overfitting. Fig. 4 also shows that accuracy remains reasonable even when the projection matrix has only 1% or even 0.1% non-zero elements, with encoding time 100× to 1000× faster.

Experiments on structured features

We also evaluated our method on two datasets of traditional "hand-crafted" features that are structured. The first dataset is **GIST-960** [18], which contains one million 960-dimensional GIST features [25] and 10,000 queries. The second dataset is **VLAD-25600**. The VLAD features [19] are extracted from 100 thousands images randomly sampled from the INRIA image set [17]. The 25600-d VLAD features are generated by encoding 128-d SIFT vectors [24] to a 200-center codebook. An extra random subset of 1000 samples are used as queries in this dataset.

Fig. 5 and Fig. 6 show the results on these datasets, using the same protocol as in Fig. 2. On these structured features, our SP method still achieves competitive accuracy with the dense projection methods (ITQ and LSH), but with substantially faster encoding. Further, SP is faster and more accurate than BP and CBE.

The experiments on DNN features and structured features show that our method is robust to arbitrary features, indicating the sparsity constraint as a regularizer is insensitive to how the signal is generated. Meanwhile, the l_0-sparsity can easily control the number of non-zero elements in the projection, and thus the time/memory consumption.

5.2. Image Retrieval

In Krizhevsky *et al.*'s work [20], the responses of the second fc layer in the CNN model are used as holistic image features for semantic image retrieval. We evaluate the performance of binary encoding methods for this task, on the "Holidays + 1M Flickr" dataset [17]. This dataset con-

		Storage (relative)	*mAP*	*Encoding time* (ms)
4096-d (float)		1	67.1	-
1024 bits	BP	1/128	62.8	0.49
	CBE	1/128	63.7	1.26
	SP	1/128	**64.3**	**0.29**
4096 bits	BP	1/32	65.8	1.53
	CBE	1/32	66.1	1.26
	SP	1/32	**66.4**	**1.14**
8192 bits	**SP**	1/16	**66.8**	2.89
16384 bits	**SP**	1/8	**67.0**	6.28

Table 1. **Image retrieval** performance on **Holidays+1M**. SP has 5% non-zero elements in the projection matrix. Each 4096-d raw feature requires 16384 bytes (131,072 bits), so the database of raw features requires ∼16GB memory.

		100-class (top-1 acc.)	*1000-class* (top-1 acc.)
4096-d (float)		77.1 ± 1.5	65.0
1024 bits	BP	72.9 ± 1.3	58.1
	CBE	73.0 ± 1.3	59.2
	SP	$\mathbf{73.8 \pm 1.3}$	**60.1**
4096 bits	threshold [1]	73.5 ± 1.4	59.1
	BP	76.0 ± 1.5	63.2
	CBE	75.9 ± 1.4	63.0
	SP	$\mathbf{76.3 \pm 1.5}$	**63.3**
8192 bits	**SP**	$\mathbf{76.8 \pm 1.4}$	**64.2**
16384 bits	**SP**	$\mathbf{77.1 \pm 1.6}$	**64.5**

Table 2. **Classification accuracy** on **ImageNet 2012** dataset (measured by top-1 accuracy) when features are encoded to binary codes. SP has 5% non-zero elements in the projection matrix.

tains 1,419 images in 500 different scenes, with an extra one million Flickr images as distracters. 500 query images are provided along with their ground-truth neighbors under the same scenes. We represent each image by a 4096-d deep learning feature as introduced in the above experiments.

Following previous practices [18, 10, 32], we treat image retrieval as an ANN search problem of the encoded features, while the ground-truth neighbors are defined by semantic labels. Given a query image, we perform Hamming ranking and evaluate mAP using the semantic ground-truth.

Table 1 shows the results on the Holidays+1M set. We have the following observations. (1) As a baseline of no encoding, the mAP of 4096-d deep learning features is 67.1%, which is significantly higher than using 64,000-d VLAD features (39.0% as reported in [10]). (2) We compare our method with BP and CBE using 1024 and 4096 bits. Our method leads to better mAP, while the encoding time is smaller. For example, with 1024 bits, our encoding time is only about 1/4 of CBE. (3) Our method can be used for higher-dimensional codes ($b > d$) when better accuracy is desired. With 16384 bits, our method has almost no degradation (67.0% mAP) with respect to the use of the deep learning features. (4) Table 1 also lists the storage required for the database. Without encoding, the storage is 16 GB for 4096-d floating number features. The binary encoding methods can significantly reduce this cost.

5.3. Image Classification

We further evaluate the binary codes as compact features for image classification, as in [10, 32]. For a baseline, we use the above ZF model [33] to extract 4096-d features from the second fully connected layer. Similar to [20], we extract ten views (center, four corners ,and their horizontal flipping) from an image. The ten 4096-d features of these views are extracted, and then averaged as a single 4096-d feature. Following [7, 33, 13], we train a one-vs-rest linear SVM classifier [8] on these features using the ImageNet 2012 [27] training set. On the 1000 classes of test set, this implementation leads to 65.0% accuracy (or equivalently 35.0% top-1 error, comparable to the testing in [13]). We also randomly split the 1000 classes into 10 clusters (100 classes per cluster), and train SVM classifiers on each of them. The average accuracy on 100 classes is 77.1% (see Table 2).

To generate binary image representations, we apply binary encoding methods on the 4096-d features. Besides BP and CBE, we also compare with a thresholding scheme in [1] that binarizes by thresholding each dimension at zero. For all methods, after binary codes are generated, linear SVM classifiers are trained on these new representations.

Table 2 shows the comparisons. The learning-based methods (BP, CBE and SP) are more accurate than simple thresholding (which is only valid when $b = d$). Our method is more accurate than BP and CBE with the same number of bits. And our method can generate higher-dimensional binary codes ($b > d$), which can further improve the accuracy. In the case of 100-class and 16384 bits, SP shows no degradation (77.1% accuracy) compared with no encoding. In the case of 1000-class, the higher-dimensional codes can still improve the accuracy and approach the no encoding baseline. Note that these representations are still more compact than the floating number representations, requiring only 1/128 to 1/8 storage cost.

5.4. Discussion

In the above experiments (Fig. 2,5,6 and Table 1,2), we find that the binary code length b required to achieve graceful degradation (compared with no encoding) is usually around $b \sim O(d)$. For example, on the DNN-4096 dataset (Fig. 2), the mAP gets saturated near 8192 bits; on the GIST-960 dataset, the mAP gets saturated near 512 bits or 1024 bits. In the case of image retrieval (Table 1), graceful degradation ($<1\%$) is achieved with 4096 bits or more.

These observations of $b \sim O(d)$ justify the requirement of using high-dimensional binary codes for high-dimensional data. Short binary codes (like 128 bits, see Fig. 2) 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 encoding.

6. Conclusion and Future Work

We have proposed an effective sparse regularizer for learning high-dimensional projections. The issue of overfitting have drawn attention in related areas like deep learning. The recent success of deep learning is partially contributed by the progress of advanced regularization techniques, such as dropout [14, 20]. We plan to investigate sparse regularizers for deep models in the future.

Acknowledgement. We thank Tiezheng Ge for valuable discussions.

References

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, pages 329–344. Springer, 2014.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.

[3] T. Blumensath and M. E. Davies. Iterative thresholding for sparse approximations. *Journal of Fourier Analysis and Applications*, 14(5-6):629–654, 2008.

[4] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic decomposition by basis pursuit. *SIAM journal on scientific computing*, pages 33–61, 1998.

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

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, 2004.

[7] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *JMLR*, 2008.

[9] T. Ge, K. He, Q. Ke, and J. Sun. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, 2013.

[10] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik. Learning binary codes for high-dimensional data using bilinear projections. In *CVPR*, 2013.

[11] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *CVPR*, 2011.

[12] J. C. Gower and G. B. Dijksterhuis. *Procrustes problems*, volume 3. Oxford University Press Oxford, 2004.

[13] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *ECCV*, 2014.

[14] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv:1207.0580*, 2012.

[15] R. A. Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1990.

[16] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC*, 1998.

[17] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[18] H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *TPAMI*, 2011.

[19] H. Jegou, M. Douze, C. Schmid, and P. Perez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[20] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[21] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *NIPS*, 2009.

[22] K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics*, 2000.

[23] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *NIPS*, 2006.

[24] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60:91–110, 2004.

[25] A. Oliva and A. Torralba. Modeling the shape of the scene: a holistic representation of the spatial envelope. *IJCV*, 2001.

[26] F. Perronnin and C. Dance. Fisher kernels on visual vocabularies for image categorization. In *CVPR*, 2007.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *arXiv:1409.0575*, 2014.

[28] J. Sánchez and F. Perronnin. High-dimensional signature compression for large-scale image classification. In *CVPR*, 2011.

[29] P. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[30] Y. Wang, J. Yang, W. Yin, and Y. Zhang. A new alternating minimization algorithm for total variation image reconstruction. *SIAM Journal on Imaging Sciences*, 2008.

[31] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[32] F. X. Yu, S. Kumar, Y. Gong, and S.-F. Chang. Circulant binary embedding. In *ICML*, 2014.

[33] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. In *ECCV*, 2014.