Classification by Retrieval: Binarizing Data and Classifier

Fumin Shen[1], Yadong Mu[2], Yang Yang[1], Wei Liu[3], Li Liu[4], Jingkuan Song[1], Heng Tao Shen[1]*
[1]Center for Future Media and School of Computer Science and Engineering, University of Electronic Science and Technology of China [2] Institute of Computer Science and Technology, Peking University
[3]Tencent AI Lab [4]Malong Technologies Co., Ltd

ABSTRACT

This paper proposes a generic formulation that significantly expedites the training and deployment of image classification models, particularly under the scenarios of many image categories and high feature dimensions. As the core idea, our method represents both the images and learned classifiers using binary hash codes, which are simultaneously learned from the training data. Classifying an image thereby reduces to retrieving its nearest class codes in the Hamming space.

Specifically, we formulate multi-class image classification as an optimization problem over binary variables. The optimization alternatingly proceeds over the binary classifiers and image hash codes. Profiting from the special property of binary codes, we show that the sub-problems can be efficiently solved through either a binary quadratic program (BQP) or linear program. In particular, for attacking the BQP problem, we propose a novel bit-flipping procedure which enjoys high efficacy and local optimality guarantee. Our formulation supports a large family of empirical loss functions and is here instantiated by exponential/linear losses. Comprehensive evaluations are conducted on several representative image benchmarks. The experiments consistently observe reduced computational and memory complexities of model training and deployment, without sacrifice of accuracies.

CCS CONCEPTS

•**Information systems** →*Similarity measures;* •**Computing methodologies** →*Supervised learning by classification;*

KEYWORDS

Hashing; binary codes

ACM Reference format:
Fumin Shen[1], Yadong Mu[2], Yang Yang[1], Wei Liu[3], Li Liu[4], Jingkuan Song[1], Heng Tao Shen[1]*. 2017. Classification by Retrieval: Binarizing Data and Classifier. In *Proceedings of SIGIR '17, Shinjuku, Tokyo, Japan, August 07-11, 2017,* 11 pages.
DOI: http://dx.doi.org/10.1145/3077136.3080767

*Corresponding author: Heng Tao Shen

1 INTRODUCTION

In recent years, large-scale visual recognition problem has attracted tremendous research enthusiasm from both academia and industry owing to the explosive increase in data size and feature dimensionality [42–44]. Classifying an image into thousands of categories often entails heavy computations by using a conventional classifier, exemplified by k nearest neighbor (k-NN) and support vector machines (SVM), on a commodity computer. For the image recognition problem with many categories, the computational and memory overhead primarily stems from the large number of classifiers to be learned. The complexities can be high at the stages of both training and deploying these classifiers. Considering a classification task with C different classes and D-dimensional feature representation, even the simplest linear models are comprised of $D \times C$ parameters. As an inspiring example to our work in this paper, the ImageNet dataset [7] contains annotated images from 21,841 classes in total. When experimenting with some state-of-the-art visual features (*e.g.*, 4096-dimensional deep neural networks feature), a huge number of 80 million parameters need to be learned and stored, which clearly indicates slow training and low efficacy at the deployment phase. Real-world applications (such as industrial image search engine) often require a near-real-time response. The conventional ways of training multi-class image classifiers thus have much space to be improved.

Compact binary hash codes [9] have demonstrated notable empirical success in facilitating large-scale similarity-based image search, referred to as image hashing in the literature. In a typical setting of supervised learning, the hash codes are optimized to ensure smaller Hamming distances between images of the same semantic kind. In practice, image hashing techniques have been widely utilized owing to its low memory footprint and guaranteed scalability to large data.

Though the hashing techniques for image search has been a well-explored research area, its application on large-scale optimization still remains a nascent topic in the fields of machine learning and computer vision. Intuitively, one can harness the hash codes for the image classification task through naive methods such as k-NN voting. Both the training and testing images are indexed with identical hashing functions. A new image is categorized by the majority semantic label within the hashing bucket where it is projected into. However, since the hash codes are initially optimized for image search purpose, such a naive scheme does not guarantee high accuracy for image recognition.

The most relevant works to ours are approximately solving non-linear kernel SVM via hashing-based data representation [19, 20, 27]. These methods first designate a set of hashing functions that transform the original features into binary codes. The original non-linear kernels (*e.g.*, RBF kernel) are theoretically proved to be approximated by the inner product between binary hash bits. Prominent advantages of such a treatment are two-folds: the required hash bits only weakly hinge on the original feature dimensionality, and meanwhile the non-linear optimization problem is converted into a linear alternative. As a major drawback, these works still rely on the regular real-valued based classifiers upon the binary features. Though it enables the direct application of linear solvers, the potential advantage of binary codes is not fully utilized.

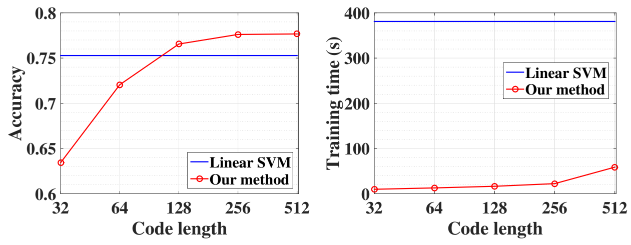
Our work is a non-trivial extension of the aforementioned line of research. We further elevate the efficacy of classification models by binarizing both the features and the weights of classifiers. In other words, our goal is to develop a generic multi-class classification framework. The classifier weights and image codes are simultaneously learned. Importantly, both of them are represented by binary hash bits. This way the classification problem is transformed to an equivalent and simpler operation, namely searching the minimal Hamming distance between the query and the C binary weight vectors. This can be extremely fast by using the built-in XOR and POPCOUNT operations in modern CPUs. We implement this idea by formulating the problem of minimizing the empirical classification error with purely binary variables. The overview of the proposed method is illustrated in Figure 2.

The major technical contributions of this work are summarized as below:

(1) We define a novel problem by binarizing both classifiers and image features and simultaneously learning them in a unified formulation. The prominent goal is to accelerate large-scale image recognition. Our work represents an unexplored research direction, namely extending hashing techniques from fast image search to the new topic of hashing-accelerated image classification.

(2) An efficient solver is proposed for the binary optimization problem. We decouple two groups of binary variables (image codes and binary classifier weights) and adopt an alternating-minimizing style procedure. Particularly, we show that the sub-problems are in the form of either binary quadratic program (BQP) or linear program. An efficient bit-flipping scheme is designed for the BQP sub-problem. Profiting from the special traits of binary variables, we are able to specify the local optimality condition of the BQP.

(3) Our formulation supports a large family of empirical loss functions and is here instantiated by exponential/linear losses. In our quantitative evaluations, both variants are compared with key competing algorithms, particularly a highly-optimized implementation of the SVM algorithm known as LibLinear [8]. Our proposed method demonstrates significant superiority in terms of train/test CPU time and classification accuracy, as briefly depicted by Figure 1.



(a) Classification accuracy. (b) Training complexity.

Figure 1: Comparison of our method with the LibLinear implementation of Linear SVM for classification on the SUN dataset with 108K images from 397 scene categories. By coding both the image feature and learned classifiers with a small number of hash bits, our method achieves better results than Linear SVM, even with much smaller model training complexity.

We also discuss a sparse binary coding model, which will further reduce the computational and memory cost of our method when applied to high-dimensional images.

The rest of this paper is organized as follows. After discussing the related literature in Section 2, we present the details of our proposed model in Section 3. Section 4 shows the experimental results of our method, followed by the conclusions of this work in Section 5.

2 RELATED WORK

Let us first review the related works which strongly motivate ours. They can be roughly cast into two categories:

2.1 Hashing for fast image search and beyond

Learning compact hash codes [41, 46] recently becomes a hot research topic in information retrieval [9, 28] and computer vision [2, 51]. The proliferation of digital photography has made billion-scale image collections a reality, and efficiently searching a similar image to the query is a critical operation in such image collections.

The seminal work of LSH [9] sheds a light on fast image search with theoretic guarantee. In a typical pipeline of hash code based image search [40, 48], a set of hashing functions are generated either in an unsupervised manner or learned by perfectly separating similar / dissimilar data pairs on the training set. The recent studies on the former category, unsupervised hashing, are more focused on the learning based methods. That is, different from the random LSH algorithm, these methods, a.k.a, *learning to hash* try to learn hash functions from the provided training data, which is shown to achieve promising performance with much more compact codes. Representative methods in this category include Spectral Hashing (SH [46]), Iterative Quantization (ITQ [10]), Manhattan Hashing [16], Anchor Graph Hashing (AGH [26]), Inductive Manifold Hashing (IMH [36, 37]), Discrete Graph Hashing (DGH [24]), *etc.*.

The latter category is known as supervised hashing since it often judges the similarity of two samples according to their semantic labels. For an unseen image, it finds the most similar images in the database by efficiently comparing the corresponding hash codes. It can be accomplished in sub-linear time using hash buckets [9]. Representative methods in this line include Binary Reconstructive Embedding (BRE [18]), Minimal Loss Hashing (MLH [30]), Semi-supervised Hashing (SSH [40]), LDA hashing [2], Kernel-Based Supervised Hashing (KSH [25]), CCA based Iterative Quantization (CCA-ITQ [10]), FastHash [21], Latent Factor Hashing (LFH [54]), Supervised Discrete Hashing (SDH [35]), Zero-shot Hashing (ZSH [50]), Discrete Semantic Ranking Hashing (DSeRH [23]) *etc.*. Recently, Shen *et al.*, [38] present a simple discrete optimization method which can substantially improve the performance of existing hashing methods.

The success of hashing technique is indeed beyond fast image search. For example, Dean et al. [6] used hash tables to accelerate the dot-product convolutional kernel operator in large-scale object detection, achieving a speed-up of approximately 20,000 times when evaluating 100,000 deformable-part models. Recently, the hashing technique was successfully applied to sketch-based image retrieval [22], action recognition [31, 32], image classification [52]. The hashing technique has been recently applied to collaborative filtering based recommendation systems [53, 55]. Particularly, the proposed model in this work can also be potentially applied to the closely related matrix factorization [3, 14], tag prediction [29, 45] and collaborative filtering [12, 13] problem.

2.2 Hashing for large-scale optimization

Noting the Hamming distance is capable of faithfully preserve data similarity, it becomes a natural thought to extend it for approximating non-linear kernels. Optimizing with non-linear kernels generally require more space to store the entire kernel matrix, which prohibits its scalability to large data. Real vectors based explicit feature mapping [39] partially remedies above issue, approximating kernel functions by the inner product between real vectors. However, they typically require high dimension towards an accurate approximation, and is thus beyond the scope of most practitioners. A more recent strand of research instead approximates non-linear kernel with binary bits, of which the prime examples can be found in [19, 20, 27]. In particular, Mu et al. developed a random subspace projection which transforms original data into compact hash bits. The inner product of hash code essentially plays the role of kernel functions. Consequently, the non-linear kernel SVM as their problem of interest can be converted into a linear SVM and resorts to efficient linear solvers like LibLinear [8].

The philosophy underlying all aforementioned works can be summarized as binarizing the features and harnessing the compactness of binary code. We here argue that the potential of hashing technique in the context of large-scale classification has not been fully explored yet. Related research is still in its embryonic stage. For example, a recent work by Shen et al. [35] proposed a Supervised Discrete Hashing (SDH) method under the assumption that good hash codes were optimal for linear classification. However, similar to other methods, SDH still classified the learned binary codes by real-valued weights. Thus the test efficiency for binary codes is still not improved compared to real-valued features.

The work [45] shared a similar idea with ours, which encoded both images and tags into binary codes for fast image tagging. However, [45] aimed to reconstruct the observed tags by the similarity estimated from the binary codes, which was very different our method which is formulated in binary codes classification. A related work is [34], which learned two set of asymmetric binary codes to reconstructed the similarity by feature inner products. Recently, a few work [4, 15, 33] proposed to accelerate deep neural networks using binary weights and activations.

After surveying related literature, we are motivated to advocate in this paper the extreme binary learning model, wherein both image features and classifiers shall be represented by binary bits. This way the learned models get rid of real-valued weight vectors and can fully benefit from high-optimized hash bit operators such as XOR.

3 THE PROPOSED MODEL

Suppose that we have generated a set of binary codes $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^n \in \{-1,1\}^{r\times n}$, where $\mathbf{b}_i$ is the r-bit binary code for original data $\mathbf{x}_i$ from the training set $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n$. For simplicity, we assume a linear hash function

$$\mathbf{b} = \mathrm{sgn}(\mathbf{P}^\top \mathbf{x}), \tag{1}$$

where $\mathbf{P} \in \mathbb{R}^{d\times r}$. In the context of linear classification, the binary codes $\mathbf{b}$ is classified according to the maximum of the score vector

$$\mathbf{W}^\top \mathbf{b} = [\mathbf{w}_1^\top \mathbf{b}, \cdots, \mathbf{w}_C^\top \mathbf{b}]^\top, \tag{2}$$

where $\mathbf{w}_c \in \{-1,1\}^r$ is the binary parameter vector for class $c \in [1,\cdots,C]$. Taking advantage of the binary nature of both $\mathbf{w}_c$ and $\mathbf{b}$, the inner product $\mathbf{w}_c^\top \mathbf{b}$ can be efficiently computed by $r - 2\mathbb{D}_{\mathbb{H}}(\mathbf{w}_c, \mathbf{b})$, where $\mathbb{D}_{\mathbb{H}}(\cdot,\cdot)$ is the Hamming distance. Thereby the standard classification problem is transformed to searching the minimum from C Hamming distances (or equivalently the maximum of binary code inner products).

Following above intuition, this paper proposes a multi-class classification framework, simultaneously learning the binary feature codes and classifier. Suppose $\mathbf{b}_i$ is the binary code of sample $\mathbf{x}_i$ and it shall be categorized as class c_i. Ideally, it expects the smallest Hamming distance (or largest inner product) to $\mathbf{w}_{c_i}$, in comparison with other classifier $\mathbf{w}_c$, $c \neq c_i$. An intuitive way of achieving this is through optimizing the inter-class "margin". Formally, we can minimize the loss $\ell\big(-(\mathbf{w}_{c_i}^\top \mathbf{b}_i - \mathbf{w}_c^\top \mathbf{b}_i)\big)$, $\forall c$, where $\ell(\cdot)$ is a generic loss function. We re-formulate multi-class classification problem as below:

$$\begin{aligned} \min_{\mathbf{W},\mathbf{B},\mathbf{P}} \quad & \sum_{i=1}^{n}\sum_{c=1}^{C} \ell\big(-(\mathbf{w}_{c_i}^\top \mathbf{b}_i - \mathbf{w}_c^\top \mathbf{b}_i)\big) \\ \text{s.t.} \quad & \mathbf{b}_i = \mathrm{sgn}(\mathbf{P}^\top \mathbf{x}_i),\ \forall i,\ \mathbf{w}_c \in \{-1,1\}^r,\ \forall c. \end{aligned} \tag{3}$$

We instantiate $\ell(\cdot)$ with the exponential loss (Section 3.1) and linear loss (Section 3.2). In fact, the loss function in Problem (3) can be broadly defined. Any proper loss function $\ell(\cdot)$ can be applied as long as it is monotonically increasing.

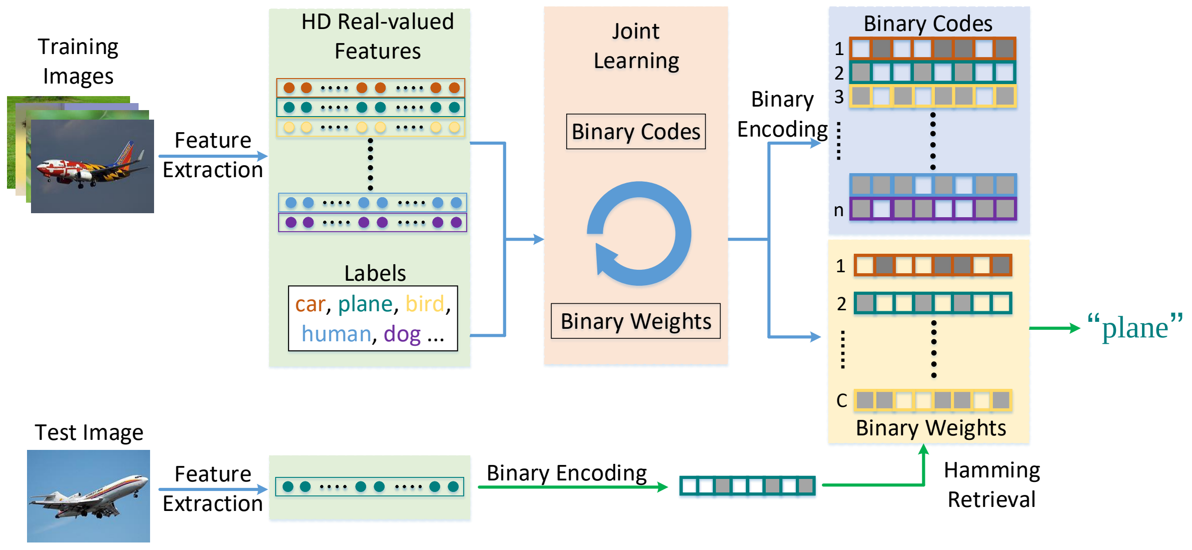


Figure 2: Overview of the proposed method. Both training images and classifiers are encoded by compact binary codes, and classifying a test image is thereby conducted by searching its neareast class codes by Hamming distances.

3.1 Learning with exponential loss

Using the exponential loss function, we have the following formulation:

$$\min_{\mathbf{W},\mathbf{B},\mathbf{P}} \sum_{i=1}^{n}\sum_{c=1}^{C} \exp\left[-(\mathbf{w}_{c_i}^\top \mathbf{b}_i - \mathbf{w}_c^\top \mathbf{b}_i)\right] \tag{4}$$

$$\text{s.t. } \mathbf{b}_i = \operatorname{sgn}(\mathbf{P}^\top \mathbf{x}_i),\ \forall i, \tag{5}$$

$$\mathbf{w}_c \in \{-1,1\}^r,\ \forall c.$$

We tackle problem (4) by alternatingly solving the sub-problems with $\mathbf{W}$, $\mathbf{B}$ and $\mathbf{P}$, respectively.

Learning binary classification weights Assume $\mathbf{B}$ is known. We iteratively update $\mathbf{W}$ row by row, i.e., one bit each time for $\mathbf{w}_c, c = 1, \cdots, C$, while keep all other $r-1$ bits fixed. Let $\mathbf{w}(k)$ denote the k-th entry of $\mathbf{w}$ and $\mathbf{w}(\backslash k)$ the vector which zeros its k-th element. We then have

$$\exp(\mathbf{w}^\top \mathbf{b}) = \exp\left[\mathbf{w}(\backslash k)^\top \mathbf{b}\right] \cdot \exp\left[\mathbf{w}(k)\mathbf{b}(k)\right]. \tag{6}$$

Denoting $u = \frac{e^{-1}+e^{1}}{2}$ and $v = \frac{e^{-1}-e^{1}}{2}$, it can be verified that

$$\exp\left[\mathbf{w}(k)\mathbf{b}(k)\right] = u - v \cdot \mathbf{b}(k)\mathbf{w}(k). \tag{7}$$

Equation (7) clearly shows the exponential function of the product of two binary bits equals to a linear function of the product. By applying (6) and (7), we write the loss term in (4) as:

$$\begin{aligned}&\exp\left[-(\mathbf{w}_{c_i}^\top \mathbf{b}_i - \mathbf{w}_c^\top \mathbf{b}_i)\right] \\ =&\gamma_{ick} \cdot \left[u - v \cdot \mathbf{b}_i(k)\mathbf{w}_c(k)\right] \cdot \left[u + v \cdot \mathbf{b}_i(k)\mathbf{w}_{c_i}(k)\right],\end{aligned} \tag{8}$$

where the constant $\gamma_{ick} = \exp\left[\left(\mathbf{w}_c(\backslash k) - \mathbf{w}_{c_i}(\backslash k)\right)^\top \mathbf{b}_i\right]$.

After merging terms with the same orders, optimizing problem (4) with regard to $\mathbf{w}^k = [\mathbf{w}_1(k); \cdots; \mathbf{w}_C(k)]$ becomes

$$\mathbf{w}^k \leftarrow \arg\min_{\mathbf{w}^k} \frac{1}{2}(\mathbf{w}^k)^\top \mathbf{H}^k \mathbf{w}^k + (\mathbf{w}^k)^\top \mathbf{g}^k, \tag{9}$$

where

$$\begin{aligned}\mathbf{H}^k &= -2v^2 \mathbf{Y}\mathbf{\Gamma}^k, \\ \mathbf{g}^k &= uv\mathbf{Y}(\mathbf{b}^k \odot \mathbf{\Gamma}\mathbf{1}) - uv\Gamma^\top \mathbf{b}^k.\end{aligned} \tag{10}$$

Here $\mathbf{\Gamma}^k \in \mathbb{R}^{n\times C}$ includes its entries γ_{ick}, $\mathbf{b}^k$ is the n-dimension vector including the k_{th} binary bits of training data. $\mathbf{Y} \in \mathbb{R}^{C\times n}$ is the label matrix whose entry y_{ci} at coordinate (c,i) equals to 1 if sample $\mathbf{x}_i$ belongs to class c and 0 otherwise. $\odot$ denotes the element-wise product. $\mathbf{1}$ is the vector with all ones.

Problem (9) is a binary quadratic program (BQP) and can be efficiently solved by a sequential bit flipping operation. A local optimum can be guaranteed. We solve problem (9) by investigating the local optimality condition. Intuitively, for any local optimum, flipping any of its hash bits will not decrease the objective value of problem (9). Let $\mathbf{H}_{*,c}, \mathbf{H}_{c,*}$ denote the column or row vector indexed by c respectively. $\mathbf{g}(c)$ and $\mathbf{w}(c)$ represents the c_{th} element of $\mathbf{g}$ and $\mathbf{w}$, respectively. In problem (9), collecting all terms pertaining to $\mathbf{w}(c)$ obtains

$$f(\mathbf{w}^k(c)) = \frac{1}{2}\left(\mathbf{H}_{*,c}^{k\top} + \mathbf{H}_{c,*}^k\right)\mathbf{w}^k \cdot \mathbf{w}^k(c) + \mathbf{g}^k(c) \cdot \mathbf{w}^k(c).$$

By flipping $\mathbf{w}(c)$, the gain of the objective function of problem (9) is

$$\Delta_{\mathbf{w}^k(c)\to -\mathbf{w}^k(c)} = f(-\mathbf{w}^k(c)) - f(\mathbf{w}^k(c)). \tag{11}$$

Regarding the local optimality condition of problem (9), we have the observation below:

Algorithm 1 Sequential bit flipping

```
1: while local optimality condition does not hold do
2:     Calculate the bit-flipping gain $\Delta_{\mathbf{w}(c)\to-\mathbf{w}(c)}$ for $c = 1,\dots,C$;
3:     Select $\hat{c} = \arg\min_c \Delta_{\mathbf{w}(c)\to-\mathbf{w}(c)}$ and $\Delta_{min} = \min_c \Delta_{\mathbf{w}(k)\to-\mathbf{w}(c)}$;
4:     if $\Delta_{min} < 0$ then
5:         Set $\mathbf{w}(c) \leftarrow -\mathbf{w}(c)$;
6:     else
7:         Exit;
8:     end if
9: end while
```

Proposition 3.1. ***(Local optimality condition):*** *Let $\mathbf{w}^*$ be a solution of problem (9). $\mathbf{w}^*$ is a local optimum when the condition $\Delta_{\mathbf{w}(c)\to-\mathbf{w}(c)} \geq 0$ holds for $c = 1,\dots,C$.*

Proof. The conclusion holds by a simple application of proof of contradiction. Recall that $\mathbf{w}$ is a binary vector. Flipping any bit of $\mathbf{w}$ will incur a specific change of the objective function as described by $\Delta_{\mathbf{w}(i)\to-\mathbf{w}(i)}$. When the changes incurred by all these flipping operations are all non-negative, $\mathbf{w}^*$ is supposed to be locally optimal. Otherwise, we can flip the bit with negative $\Delta_{\mathbf{w}(i)\to-\mathbf{w}(i)}$ to further optimize the objective function. □

With the above analysis, we summarize our algorithm for updating the C-bit $\mathbf{w}^k$ as in Algorithm 1.

Binary code learning With $\mathbf{W}$ fixed, we have the following problem

$$\min_{\mathbf{B},\mathbf{P}} \sum_{i=1}^{n}\sum_{c=1}^{C} \exp\left[-(\mathbf{w}_{c_i}^\top \mathbf{b}_i - \mathbf{w}_c^\top \mathbf{b}_i)\right] \tag{12}$$
$$\text{s.t. } \mathbf{b}_i = \operatorname{sgn}(\mathbf{P}^\top \mathbf{x}_i),\ \forall i,\ \mathbf{w}_c \in \{-1,1\}^r,\ \forall c.$$

Technically, we could reformulate this constrained problem into a joint learning problem regularized by the fidelity between binary codes and hash function outputs, as in [35]. However, in practice, we find this solution does not improve the performance than the two-step method: first learning binary codes followed by learning the hash prediction matrix. In this work, we adopt the two-step method.

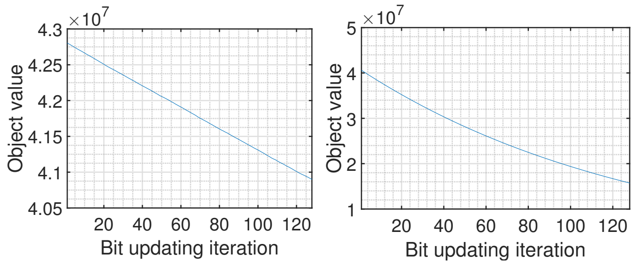
Similar to the optimization procedure for $\mathbf{W}$, we solve for $\mathbf{B}$ by a coordinate descent scheme. In particular, at each iteration, all the rest $r-1$ hash bits are fixed except for the k-th hash bit $\mathbf{b}^k = [\mathbf{b}_1(k);\cdots;\mathbf{b}_n(k)]$. Let $\mathbf{b}_i(\backslash k)$ denote the vector which zeros its k-th element $\mathbf{b}_i(k)$. We rewrite equation (8) w.r.t $\mathbf{b}_i(k)$ as

$$\begin{aligned}&\exp\left[-(\mathbf{w}_{c_i}^\top \mathbf{b}_i - \mathbf{w}_c^\top \mathbf{b}_i)\right] \\ =&z_{ick}\cdot \exp\left[(\mathbf{w}_c(k) - \mathbf{w}_{c_i}(k))\mathbf{b}_i(k)\right]\end{aligned} \tag{13}$$

where $z_{ick} = \exp\left[(\mathbf{w}_c - \mathbf{w}_{c_i})^\top \mathbf{b}_i(\backslash k)\right]$.

Denote $u' = \frac{e^{-2}+e^2}{2}$, $v' = \frac{e^{-2}-e^2}{2}$. Similar as (7), we have

$$\exp\left[(\mathbf{w}_c(k) - \mathbf{w}_{c_i}(k))\mathbf{b}_i(k)\right] = \begin{cases} 0, & \mathbf{w}_{c_i}(k) = \mathbf{w}_c(k) \\ u' + v'\cdot \mathbf{b}_i(k), & \mathbf{w}_{c_i}(k) = 1, \mathbf{w}_c(k) = -1 \\ u' - v'\cdot \mathbf{b}_i(k), & \mathbf{w}_{c_i}(k) = -1, \mathbf{w}_c(k) = 1. \end{cases}$$



(a) Updating W. **(b) Updating B.**

Figure 3: Object value as a function of bit updating iteration k in optimizing W and B on the dataset of SUN397.

We can see that, the non-linear exponential loss term becomes either a constant or linear function with regard to $\mathbf{b}_i(k)$.

Let matrix $\mathbf{Z}^k \in \mathbb{R}^{n\times C}$ include its entry at the coordinate (i,c) as z_{ick} if $\mathbf{w}_{c_i}(k) = 1, \mathbf{w}_c(k) = -1$ and 0 otherwise; similarly let matrix $\bar{\mathbf{Z}}^k \in \mathbb{R}^{n\times C}$ include its entry at the coordinate (i,c) as z_{ick} if $\mathbf{w}_{c_i}(k) = -1, \mathbf{w}_c(k) = 1$ and 0 otherwise. Then the loss in (4) can be written as w.r.t $\mathbf{b}^k$

$$\begin{aligned}&\sum_{i=1}^{n}\sum_{c=1}^{C} \exp\left[-(\mathbf{w}_{c_i}^\top \mathbf{b}_i - \mathbf{w}_c^\top \mathbf{b}_i)\right] \\ =&\sum_{i=1}^{n}\sum_{c=1}^{C} \mathbf{Z}^k(i,c)\cdot(u' + v'\mathbf{b}_i(k)) + \bar{\mathbf{Z}}^k(i,c)\cdot(u' - v'\mathbf{b}_i(k)) \\ =&\sum_{i=1}^{n} \mathbf{z}^k(i)\cdot(u' + v'\mathbf{b}_i(k)) + \bar{\mathbf{z}}^k(i)\cdot(u' - v'\mathbf{b}_i(k)),\end{aligned} \tag{14}$$

where $\mathbf{z}^k(i) = \sum_{c=1}^{C} \mathbf{Z}^k(i,c)$ and $\bar{\mathbf{z}}^k(i) = \sum_{c=1}^{C} \bar{\mathbf{Z}}^k(i,c)$.

Then we have the following optimization problem

$$\min_{\mathbf{b}^k} v'(\mathbf{z}^k - \bar{\mathbf{z}}^k)^\top \mathbf{b}^k, \quad \text{s.t. } \mathbf{b}^k \in \{-1,1\}^r, \tag{15}$$

which has a optimal solution

$$\mathbf{b}^k = -\operatorname{sgn}(v'(\mathbf{z}^k - \bar{\mathbf{z}}^k)). \tag{16}$$

Figure 3 shows the objective value as a function of the bit updating iteration number. As can be seen, with the proposed coordinate descent optimizing procedure for both $\mathbf{W}$ and $\mathbf{B}$, the object value consistently decreases as updating the hash bits in each sub-problem. The optimization for the original problem (4) typically converges in less than 3 iterations of alternatively optimizing $\mathbf{W}$ and $\mathbf{B}$.

3.2 Learning with the linear loss

In this section, we instantiate our model with the simple linear loss:

$$\min_{\mathbf{W},\mathbf{B},\mathbf{P}} \sum_{i=1}^{n}\sum_{c=1}^{C} (\mathbf{w}_c^\top \mathbf{b}_i - \mathbf{w}_{c_i}^\top \mathbf{b}_i) \tag{17}$$
$$\text{s.t. } \mathbf{b}_i = \operatorname{sgn}(\mathbf{P}^\top \mathbf{x}_i),\ \forall i,\ \mathbf{w}_c \in \{-1,1\}^r,\ \forall c.$$

Similar as with the exponential loss, we tackle problem (17) by alternatively solving the sub-problems regarding to $\mathbf{W}$, $\mathbf{B}$

and $\mathbf{P}$, respectively. We write problem (17) as w.r.t. $\mathbf{W}$,

$$\min_{\mathbf{W}} \quad \sum_{i=1}^{n}\sum_{c=1}^{C}(\mathbf{w}_c^\top \mathbf{b}_i - \mathbf{w}_{c_i}^\top \mathbf{b}_i) \tag{18}$$
$$\text{s.t.} \quad \mathbf{w}_c \in \{-1,1\}^r, \forall c.$$

Collecting all terms with $\mathbf{w}_c$, $\forall c$, problem (18) writes

$$\min_{\mathbf{W}} \quad \mathbf{w}_c^\top \big(\sum_{i=1}^{n}\mathbf{b}_i - C\sum_{i=1,c_i=c}^{n}\mathbf{b}_i\big) \tag{19}$$
$$\text{s.t.} \quad \mathbf{w}_c \in \{-1,1\}^r,$$

which has optimal solution

$$\mathbf{w}_c = \mathrm{sgn}\big(C\sum_{i=1,c_i=c}^{n}\mathbf{b}_i - \sum_{i=1}^{n}\mathbf{b}_i\big). \tag{20}$$

For the sub-problem regarding to $\mathbf{B}$, we first let matrix $\mathbf{W}^o$ of size $r \times n$ include its i_{th} column as $\mathbf{w}_i^o = \sum_{c=1}^{C}\mathbf{w}_c - C\mathbf{w}_{c_i}$. Problem (18) writes w.r.t. $\mathbf{B}$ as

$$\min_{\mathbf{B}} \mathrm{trace}(\mathbf{W}^{o\top}\mathbf{B}), \quad \text{s.t.} \;\; \mathbf{B} \in \{-1,1\}^{r\times n}.$$

$\mathbf{B}$ can be efficiently computed by $-\mathrm{sgn}(\mathbf{W}^o)$. It is clear that, both of the two sub-problems associated with $\mathbf{B}$ and $\mathbf{W}$ have closed-form solutions, which significantly reduces the computation overhead of classifier training and code learning.

3.3 Binary code prediction

With the binary codes $\mathbf{B}$ for training data $\mathbf{X}$ obtained, the hash function $h(\mathbf{x}) = \mathrm{sgn}(\mathbf{P}^\top\mathbf{x})$ is obtained by solving a simple linear regression system

$$\mathbf{P} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{B}. \tag{21}$$

Then for a new sample $\mathbf{x}$, the classification is conducted by searching the minimum of C Hamming distances:

$$c^* = \arg\min_{c}\{\mathbb{D}_{\mathbb{H}}\big(\mathbf{w}_c, h(\mathbf{x})\big)\}, \; c = 1, \cdots, C. \tag{22}$$

The binary coding step occupies the main computation in the testing stage, which is $O(dr)$ in time complexity. In some cases, the image feature dimensions d can be very large or the code length r need to be long for satisfying performance. Then the computational cost of binary code prediction and storage cost of the projection $\mathbf{P}$ should not be neglected. This motivates us to devise a sparse model, where most entries of the hash projection matrix $\mathbf{P}$ are zeros. That is, we solve the following ℓ_0 problem:

$$\min_{\mathbf{P}} ||\mathbf{P}^\top\mathbf{X} - \mathbf{B}||^2$$
$$\text{s.t.} \;\; ||\mathbf{P}||_0 \le s, \tag{23}$$

where $||\cdot||_0$ denotes the ℓ_0 norm and t is the number of nonzero entries of $\mathbf{P}$. Problem (23) has been well known as the sparse representation problem, and exactly solving it can be NP-hard due to the involvement of nonconvex and nonsmooth ℓ_0 norm. The typical solution can be attained by relaxing the ℓ_0 norm to the convex ℓ_1, which is much easier to slove while still admitting the sparsity of resultant model [47]. In this work, however, we choose to solve the ℓ_0 problem since the nonzero number (and thus the computation and memory costs) can be exactly controlled.

Inspired by the recent advance of the proximal optimization method [1], we solve problem (23) by the following iterative procedure. First let us introduce the sparse projection operator:

$$T_s(\mathbf{U}) = \arg\min_{\mathbf{V}}\big\{||\mathbf{U}-\mathbf{V}||^2 : ||\mathbf{V}||_0 \le s\big\}. \tag{24}$$

It can be easily observed that the sparse projection operator selects exactly the first s largest elements (in absolute value) of $\mathbf{U}$ and set all other elements zeros. For convenience, let us denote the loss in (23) by $\mathcal{L}(\mathbf{P})$. According to [1], problem (23) can be solved by the proximal alternating linearization minimization (PALM) algorithm, that is, at the tth iteration, $\mathbf{P}$ is updated by

$$\mathbf{P}^{(t)} = T_s\big(\mathbf{P}^{(t-1)} - \frac{1}{\gamma_t}\nabla\mathcal{L}(\mathbf{P})\big), \tag{25}$$

where $\nabla\mathcal{L}(\mathbf{P}) = \mathbf{X}\mathbf{X}^\top\mathbf{P} - \mathbf{X}\mathbf{B}^\top$. The sequence of parameters γ_t is chosen such that γ_t is greater than the Lipschitz modulis of the loss function $\mathcal{L}$, which is $||\mathbf{X}\mathbf{X}^\top||_{\text{fro}}$ here. With this updating procedure, the generated sequence of $\mathbf{P}$ will globally converge to a critical point [1]. The reader is referred to [1] for the details of the PALM algorithm. The work [48] also proposed a sparse projection model for binary coding, which was however derived very differently. Another difference is the work [48] focuses on image retrieval while our method focuses on the classification task.

3.4 Discussion

As mentioned above, the proposed method classifies a new sample $\mathbf{x}$ by first computing its binary codes $(\mathrm{sgn}(\mathbf{P}^\top\mathbf{x}))$ and retrieving its nearest neighbor from $\{\mathbf{w}_c\}_{c=1}^{C}$. Taking advantage of extremely fast bit operations (XOR and POPCOUNT), the computation consumed by the second step can be mostly ignored. In fact, the computational burden of our proposed method in the testing phase mainly comes from projecting a feature vector onto the matrix $\mathbf{P}$, which costs $O(dr)$ in time. With the sparse model, the time complexity can be reduced to $O(D)$ with $D << dr$ the number of nonzero elements in $\mathbf{P}$. For conventional linear classifiers (*e.g.*, linear SVM), the time complexity for classifying a test sample is $O(dC)$.

It is clear that the proposed method is not always relatively more efficient, especially when the number of classes is small. However, we note that our method is more suitable for accelerating large-scale classification problems with a large number of classes. The time complexity does not change much with different numbers of classes. In this sense, our proposed method can be more reasonably treated as a classification method with nearly constant time complexity at test time, insensitive to the number of classes. In addition, we will show in the experiments that the proposed method has lower training complexity even with a small number of classes, as justified in Table 1.

Another advantage of our method is it significantly reduce the storage cost by converting high-dimensional real-valued features and classification models to compact binary codes. Differently, previous hashing algorithms fail to take advantage of the binary nature of obtained hash codes (still treated as real-valued features) when applied for classification.

4 EXPERIMENTS

In this section, we conduct comprehensive evaluations to validate the efficacy of our proposed method in terms of both classification accuracy and computational efficiency.

4.1 Datasets

Three large-scale datasets: SUN397 [49], ImageNet [7] and Caltech-256[2] are used in our experiments.

SUN397 [49] contains about 108K images from 397 scene categories, where each image is represented by a 1,600-dimensional feature vector extracted by PCA from 12,288-dimensional Deep Convolutional Activation Features [11]. We use a subset of this dataset including 42 categories with each containing more than 500 images; 100 images are sampled uniformly randomly from each category to form a test set of 4,200 images. As a subset of **ImageNet** [7], the large dataset ILSVRC 2012 contains over 1.2 million images of totally 1,000 categories. We form the evaluation database by the 100 largest classes with total 128K images from training set, and 50,000 images from validation set as test set. We use the 4096-dimensional features extracted by the convolution neural networks (CNN) model [17]. The **Caltech-256** dataset contains 256 object categories including over 30K images. Similarly, we extract the 4096-dimensional CNN features by the model in [17]. For this dataset, we use half of the images in each category as training data and the the half as test set.

4.2 Compared methods and evaluation metrics

The proposed method with both exponential loss (denoted by Ours-Exponential) and linear loss (denoted by Ours-Linear) are evaluated. The binary code prediction is conducted with (21) by default. We will evaluate the sparse model (23) in Section 4.5. The proposed methods are compared with two popular linear classifiers: one-vs-all linear SVM (OVA-SVM) and multi-class SVM (Multi-SVM [5]), both of which are implemented using the LibLinear software package [8]. For these two methods, we tune the parameter c from the range [1e-3, 1e3] and the best results are reported. Note that during classification, both the features and classifiers are continuous for the SVM algorithms, which is different from our methods. We also compare our methods against several state-of-the-art binary code learning methods including Locality Sensitive Hashing (LSH) implemented by signed random projections, CCA-ITQ [10], KSH [25], FastHash [21] and SDH [35]. Due to the expensive training of KSH and FastHash, we randomly select 5K and 10K samples, respectively, from training data for learning these two models. All available training data are used for learning for other methods. The classification results of these hashing methods are obtained by performing the multi-class linear SVM over the predicted binary codes by the corresponding hash functions. We use the public codes and suggested parameters of these methods from the authors.

We extensively evaluate these compared methods in terms of classification accuracy, computation efficiency (training time and testing time) and storage memory overhead.

4.3 Accuracy and computational efficiency

In this part, we extensively evaluate the proposed two methods with the compared algorithms in classification accuracy and computation time. We set the code length to 128 for our method and the hashing algorithms LSH, CCA-ITQ, KSH, FastHash and SDH.

The main results on SUN397, ImageNet and Caltech-256 are reported in Table 1. Between the algorithms with different loss functions of our proposed model, we can see that the method with the exponential loss performs slightly better than that with the linear loss, while the latter one benefits from a much more efficient training. This is not surprising because Ours-Linear solves two alternating sub-problems both with closed-form solutions. In the following experiments, we use the linear method for its high efficiency.

Compared to other methods, the results in Table 1 clearly show that, our methods achieve very competitive classification accuracies with the full-precision linear SVMs on SUN397 and ImageNet. For example, our method obtains 76.56% accuracy on SUN397 which is better Multi-SVM by 1.28%. While OVA-SVM achieves slightly better results than our method, the training and testing time cost are much higher. On the Caltech-256 dataset, our methods with 128 bits (together with other hashing algorithms) do not perform as well as SVM, which may be because the code length is too short to encode enough information. Figure 5 depicts the impacts of code lengths on the performance of our algorithm.

In the meanwhile, even being constrained to learning binary classification weights, our methods obtain noticeably better results than the state-of-the-art hashing algorithms. Specifically, Ours-Exponential outperforms the best results obtained the hashing algorithms by 2.88%, 2.4% on SUN397 and ImageNet , respectively. On Caltech-256, our two algorithms attain very competitive accuracies with CCA-ITQ and SDH which are much higher than those by LSH, KSH and FastHash.

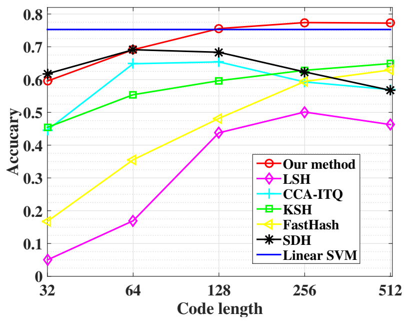
In terms of training time, we can see that our method with the linear loss runs way faster than all other methods on all the evaluated three datasets. In particular, on SUN397, Ours-Linear is 50× and 23× faster than the LibLinear implementations of one-vs-all SVM and multi-class SVM. Compared with the hashing algorithm, our method runs over 28× faster than the fastest LSH followed by Liblinear. For the testing time, the benefit of binary dimension reduction for our methods and other three hashing algorithms is clearly shown on the SUN397 dataset with a large number of categories. Our methods requires less testing time than the hashing based classification methods, due to the extremely fast classification implemented by searching in the Hamming space.

We also evaluate the compared methods with different code lengths, where the detailed results on SUN397, ImageNet

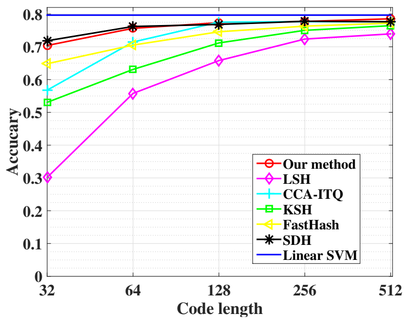
[2] http://www.vision.caltech.edu/Image_Datasets/Caltech256/.

Table 1: Comparative results in terms of test accuracy (%), training and testing time (seconds). Experiments are conducted on a standard PC with a quad-core Intel CPU and 32GB RAM. For LSH, CCA-ITQ, SDH and our methods, 128 bits are used. OVA-SVM and Multi-SVM is performed with LibLinear, where the best accuracies are reported with parameter c chosen from {1e-3, 1e-2, 1e-1, 1, 1e1, 1e2, 1e3}.

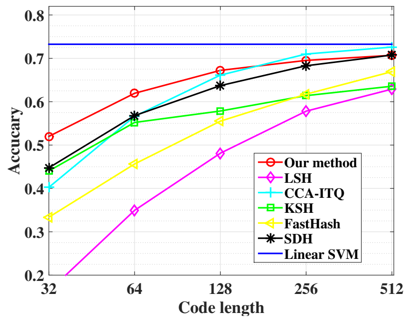
Method	SUN397			ImageNet			Caltech-256		
	acc (%)	train time	test time	acc (%)	train time	test time	acc (%)	train time	test time
OVA-SVM	**77.39**	818.87	1.55e-5	**79.84**	151.02	1.15e-5	**73.31**	47.37	2.70e-4
Multi-SVM	75.28	380.94	1.01e-5	79.48	93.12	1.21e-5	73.25	38.54	2.58e-4
LSH	54.11	417.42	7.75e-6	58.16	107.41	1.32e-5	51.07	15.16	2.78e-5
CCA-ITQ	69.33	452.34	8.78e-6	76.30	142.95	1.25e-5	65.14	15.12	2.86e-5
KSH	59.67	2805.50	5.19e-5	74.38	11217.71	4.96e-5	57.85	2854.31	7.39e-5
FastHash	48.17	3144.80	1.20e-3	72.66	371.65	7.75e-4	55.51	522.71	6.58e-4
SDH	72.56	2522.33	7.43e-6	76.64	1102.21	1.43e-5	**66.90**	132.56	2.83e-5
Ours-Exponential	75.44	772.11	**3.67e-6**	**79.04**	245.14	**6.54e-6**	65.52	69.31	**1.31e-5**
Ours-Linear	**76.56**	**16.45**	**3.86e-6**	77.88	**35.16**	**6.86e-6**	65.48	**11.94**	**1.35e-5**



(a) SUN397



(b) ImageNet

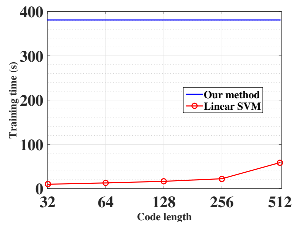


(c) Caltech-256

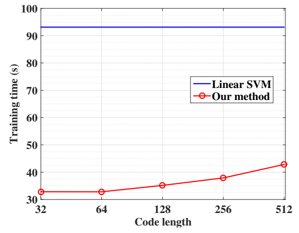
Figure 4: Comparative results of different methods in classification accuracy on SUN397 and ImageNet with code length from 32 to 512.

and Caltech-256 are shown in Figure 4. From Figure 4, it is clear that with a relatively small number of bits (*e.g.*, 256 bits), our method can achieve close classification performance to or even better than the real-valued linear SVM with real-valued features on . We can also observe that our method consistently outperforms other hashing algorithms by noticeable gaps at all code lengths on SUN397. On the ImageNet dataset, our method achieves marginally better results than SDH and CCA-ITQ, while better than KSH and the random LSH algorithm. For the Caltech-256 dataset, our method outperforms all of the compared hashing algorithms with less than 128 bits while CCA-ITQ achieves sightly better accuracies with long code lengths. However, we note that the results by ITQ are obtained by additionally learning a full-precision SVM on ITQ codes while our method classifies the generated binary codes by binary weights.

Figure 5 demonstrates the consumed training time by our method and Linear SVM by the Liblinear solver on two large-scale datasets. The computation efficiency advantage of our method is clearly shown. Our method has a nearly linear training time complexity with the code length, which can facilitate its potential applications in high-dimensional binary vector learning.



(a) SUN397

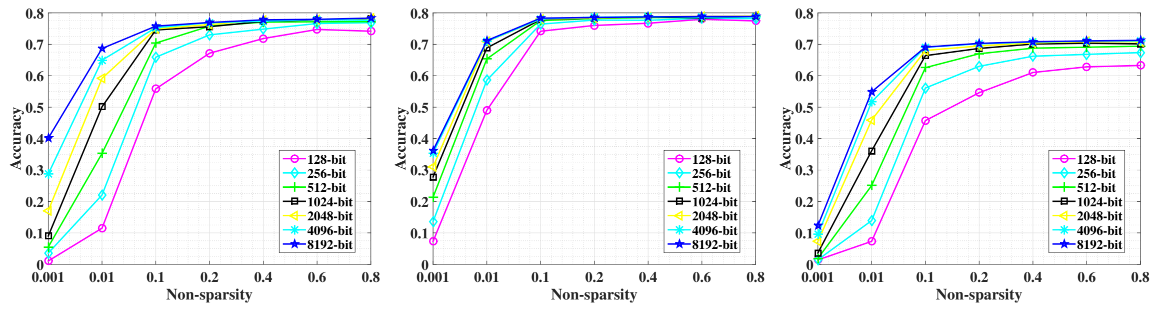


(b) ImageNet

Figure 5: Comparative results of our method and linear SVM in training time on SUN397 and ImageNet with code length from 32 to 512.

4.4 Storage cost

In this subsection, we analyze the storage cost of the image features (real-valued and binary codes) and also the running



(a) SUN397 (b) ImageNet (c) Caltech-256

Figure 6: Classification accuracy vs. non-sparity of P of the proposed sparse model on SUN397, ImageNet and Caltech-256 with various code lengths.

Table 2: Memory overhead (MB) to store the image features and classification model using Linear SVM and our method (128-bit). Note that, for our method, the trained model includes the real-valued hashing matrix (P) and the binary classification weight matrix W.

Dataset	Image features		Classification model	
	Real-valued	Binary	Linear SVM	Ours
ImageNet	3943.91	24.37	30.86	9.92
SUN397	1283.83	2.01	4.79	1.54
Caltech-256	157.04	0.97	7.89	3.95

memory overhead of storing the classification model. For our method, the trained model includes the real-valued hash function matrix (**P**) and the binary classification weight matrix. The results are reported in Table 2. It is clearly shown that our approach requires much less memory than Linear SVM for storing both the image features and classification models. Taking the ImageNet for example, Linear SVM needs over 150 times more space than our method for the dataset, and over 3 times more space for the classification models.

4.5 Evaluation of the sparse model

As discussed above, the image features or the target binary codes dimensionality can be very large, which will make the hash code prediction be expensive in both computation and storage of the hash model. To alleviate this problem, we develop a sparse model in Section 3.3. In this subsection, we evaluate the impact of the sparsity of the hash model **P** on the classification performance. Figure 6 demonstrates the accuracy vs. non-sparsity (the number of nonzeros in **P** over dr) of our method with various code lengths on SUN397, ImageNet and Caltech-256.

We can observe from Figure 6 that the accuracies are consistently improved with the increasing ratio of nonzeros in the hash model. However, the performance becomes stable when the non-sparsity is larger than 0.1 or 0.2 with a long code length being used on all these three datasets. This may provide a way for the trade-off between the performance and computation/memory cost.

5 CONCLUSIONS

This work proposed a novel classification framework, by which classification was equivalently transformed to searching the nearest binary weight code in the Hamming space. Different from previous methods, both the feature and classifier weight vectors were simultaneously learned with binary hash codes. Our framework could employ a large family of empirical loss functions, and we here especially studied the representative exponential and linear loss. For the two sub-problems regarding the binary classifier and image hash codes, a binary quadratic program (BQP) and linear program were formulated, respectively. In particular, for the BQP problem, a novel bit-flipping procedure which enjoys high efficacy and local optimality guarantee were presented. Significant computation overhead reduction of model training and deployment were obtained by our method, while without sacrifice of classification accuracies. We also discussed a sparse binary coding model, which provided a practical way to further reduce the computational and memory costs.

ACKNOWLEDGMENTS

This work was supported in part by the National Natural Science Foundation of China under Project 61502081, Project 61572108, Project 61673299 and Project 61632007, in part by the Fundamental Research Funds for the Central Universities under Project ZYGX2014Z007.

REFERENCES

[1] Jérôme Bolte, Shoham Sabach, and Marc Teboulle. 2014. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming* 146, 1-2 (2014), 459–494.
[2] M. M. Bronstein and P. Fua. 2012. LDAHash: Improved Matching with Smaller Descriptors. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 1 (2012), 66–78.
[3] Da Cao, Liqiang Nie, Xiangnan He, Xiaochi Wei, Shunzhi Zhu, Shunxiang Wu, and Tat-Seng Chua. 2017. Embedding Factorization Models for Jointly Recommending User Generated Lists and Their Contained Items. In *Proc. SIGIR.*
[4] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. 2015. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Proc. NIPS.* 3123–3131.

[5] Koby Crammer and Yoram Singer. 2002. On the algorithmic implementation of multiclass kernel-based vector machines. *J. Mach. Learn. Res.* 2 (2002), 265–292.

[6] Thomas Dean, Mark A. Ruzon, Mark Segal, Jonathon Shlens, Sudheendra Vijayanarasimhan, and Jay Yagnik. 2013. Fast, Accurate Detection of 100,000 Object Classes on a Single Machine. In *Proc. CVPR*. 1814–1821.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Proc. CVPR*. 248–255.

[8] R.E. Fan, K.W. Chang, C.J. Hsieh, X.R. Wang, and C.J. Lin. 2008. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.* 9 (2008), 1871–1874.

[9] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. 1999. Similarity search in high dimensions via hashing. In *Proc. VLDB*. 518–529.

[10] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 12 (2013), 2916–2929.

[11] Yunchao Gong, Liwei Wang, Ruiqi Guo, and Svetlana Lazebnik. 2014. Multi-scale orderless pooling of deep convolutional activation features. In *Proc. ECCV*. Springer, 392–407.

[12] Xiangnan He, Ming Gao, Min-Yen Kan, Yiqun Liu, and Kazunari Sugiyama. 2014. Predicting the popularity of web 2.0 items based on user comments. In *Proc. SIGIR*. 233–242.

[13] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proc. WWW*.

[14] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proc. SIGIR*. 549–558.

[15] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Proc. NIPS*. 4107–4115.

[16] Weihao Kong, Wu-Jun Li, and Minyi Guo. 2012. Manhattan Hashing for Large-scale Image Retrieval. In *Proc. SIGIR*. 45–54.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *Proc. NIPS*. 1097–1105.

[18] B. Kulis and T. Darrell. 2009. Learning to hash with binary reconstructive embeddings. In *Proc. NIPS*. 1042–1050.

[19] Ping Li, Gennady Samorodnitsk, and John Hopcroft. 2013. Sign Cauchy projections and Chi-square kernel. In *Proc. NIPS*. 2571–2579.

[20] Ping Li, Anshumali Shrivastava, Joshua L Moore, and Arnd C König. 2011. Hashing algorithms for large-scale learning. In *Proc. NIPS*. 2672–2680.

[21] Guosheng Lin, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and David Suter. 2014. Fast supervised hashing with decision trees for high-dimensional data. In *Proc. CVPR*. 1971–1978.

[22] Li Liu, Fumin Shen, Yuming Shen, Xianglong Liu, and Ling Shao. 2017. Deep Sketch Hashing: Fast Free-hand Sketch-Based Image Retrieval. In *Proc. CVPR*.

[23] Li Liu, Mengyang Yu, Fumin Shen, and Ling Shao. 2017. Discretely Coding Semantic Rank Orders for Image Hashing. In *Proc. CVPR*.

[24] Wei Liu, Cun Mu, Sanjiv Kumar, and Shih-Fu Chang. 2014. Discrete Graph Hashing. In *Proc. NIPS*. 3419–3427.

[25] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *Proc. CVPR*. 2074–2081.

[26] Wei Liu, Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2011. Hashing with Graphs. In *Proc. ICML*. 1–8.

[27] Yadong Mu, Gang Hua, Wei Fan, and Shih-Fu Chang. 2014. Hash-SVM: Scalable Kernel Machines for Large-Scale Visual Classification. In *Proc. CVPR*. 979–986.

[28] Liqiang Nie, Meng Wang, Zheng-Jun Zha, and Tat-Seng Chua. 2012. Oracle in image search: A content-based approach to performance prediction. *ACM Trans. Inf. Syst.* 30, 2 (2012), 13.

[29] Liqiang Nie, Yi-Liang Zhao, Xiangyu Wang, Jialie Shen, and Tat-Seng Chua. 2014. Learning to recommend descriptive tags for questions in social forums. *ACM Trans. Inf. Syst.* 32, 1 (2014), 5.

[30] Mohammad Norouzi and David M Blei. 2011. Minimal loss hashing for compact binary codes. In *Proc. ICML*. 353–360.

[31] Jie Qin, Li Liu, Ling Shao, Bingbing Ni, Chen Chen, Fumin Shen, and Yunhong Wang. 2017. Binary Coding for Partial Action Analysis with Limited Observation Ratios. In *Proc. CVPR*.

[32] Jie Qin, Li Liu, Ling Shao, Fumin Shen, Bingbing Ni, Jiaxin Chen, and Yunhong Wang. 2017. Zero-Shot Action Recognition with Error-Correcting Output Codes. In *Proc. CVPR*.

[33] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. 2016. Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. ECCV*. Springer, 525–542.

[34] Fumin Shen, Wei Liu, Shaoting Zhang, Yang Yang, and Heng Tao Shen. 2015. Learning Binary Codes for Maximum Inner Product Search. In *Proc. ICCV*. 4148–4156.

[35] Fumin Shen, Chunhua Shen, Wei Liu, and Heng Tao Shen. 2015. Supervised Discrete Hashing. In *Proc. CVPR*. 37–45.

[36] Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, and Zhenmin Tang. 2013. Inductive Hashing on Manifolds. In *Proc. CVPR*. 1562–1569.

[37] Fumin Shen, Chunhua Shen, Qinfeng Shi, Anton van den Hengel, Zhenmin Tang, and Heng Tao Shen. 2015. Hashing on Nonlinear Manifolds. *IEEE Trans. Image Proc.* 24, 6 (2015), 1839–1851.

[38] Fumin Shen, Xiang Zhou, Yang Yang, Jingkuan Song, Heng Tao Shen, and Dacheng Tao. 2016. A Fast Optimization Method for General Binary Code Learning. *IEEE Transactions on Image Processing* 25, 12 (2016), 5610–5621.

[39] Andrea Vedaldi and Andrew Zisserman. 2012. Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 3 (2012), 480–492.

[40] Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-Supervised Hashing for Large Scale Search. *IEEE Trans. Pattern Anal. Mach. Intell.* 34, 12 (2012), 2393–2406.

[41] Jingdong Wang, Ting Zhang, Jingkuan Song, Nicu Sebe, and Heng Tao Shen. 2016. A survey on learning to hash. *arXiv preprint arXiv:1606.00185* (2016).

[42] M. Wang, W. Fu, S. Hao, H. Liu, and X. Wu. 2017. Learning on Big Graph: Label Inference and Regularization with Anchor Hierarchy. *IEEE Trans. Know. Data Engin.* 29, 5 (2017), 1101–1114.

[43] Meng Wang, Weijie Fu, Shijie Hao, Dacheng Tao, and Xindong Wu. 2016. Scalable semi-supervised learning by efficient anchor graph regularization. *IEEE Trans. Know. Data Engin.* 28, 7 (2016), 1864–1877.

[44] Meng Wang, Xueliang Liu, and Xindong Wu. 2015. Visual Classification by ℓ_1-Hypergraph Modeling. *IEEE Trans. Know. Data Engin.* 27, 9 (2015), 2564–2574.

[45] Qifan Wang, Bin Shen, Shumiao Wang, Liang Li, and Luo Si. 2014. Binary codes embedding for fast image tagging with incomplete labels. In *Proc. ECCV*. Springer, 425–439.

[46] Yair Weiss, Antonio Torralba, and Robert Fergus. 2008. Spectral Hashing. In *Proc. NIPS*. 1753–1760.

[47] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. 2009. Robust face recognition via sparse representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 2 (2009), 210–227.

[48] Yan Xia, Kaiming He, Pushmeet Kohli, and Jian Sun. 2015. Sparse Projections for High-Dimensional Binary Codes. In *Proc. CVPR*. 3332–3339.

[49] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. 2010. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*. 3485–3492.

[50] Yang Yang, Yadan Luo, Weilun Chen, Fumin Shen, Jie Shao, and Heng Tao Shen. 2016. Zero-Shot Hashing via Transferring Supervised Knowledge. In *ACM Multimedia*. 1286–1295.

[51] Yang Yang, Fumin Shen, Heng Tao Shen, Hanxi Li, and Xuelong Li. 2015. Robust discrete spectral hashing for large-scale image semantic indexing. *IEEE Trans. Big Data* 1, 4 (2015), 162–171.

[52] Yang Yang, Hanwang Zhang, Mingxing Zhang, Fumin Shen, and Xuelong Li. 2015. Visual coding in a semantic hierarchy. In *ACM Multimedia*. 59–68.

[53] Hanwang Zhang, Fumin Shen, Wei Liu, Xiangnan He, Huanbo Luan, and Tat-Seng Chua. 2016. Discrete Collaborative Filtering. In *Proc. SIGIR*. 325–334.

[54] Peichao Zhang, Wei Zhang, Wu-Jun Li, and Minyi Guo. 2014. Supervised Hashing with Latent Factor Models. In *Proc. SIGIR*. 173–182.

[55] Ke Zhou and Hongyuan Zha. 2012. Learning Binary Codes for Collaborative Filtering. In *Proc. KDD*. 498–506.