This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

A Fast Optimization Method for General Binary Code Learning

Fumin Shen, Xiang Zhou, Yang Yang, Jingkuan Song, Heng Tao Shen and Dacheng Tao

***Abstract*—Hashing or binary code learning has been recognized to accomplish efficient near neighbor search, and has thus attracted broad interests in recent retrieval, vision and learning studies. One main challenge of learning to hash arises from the involvement of discrete variables in binary code optimization. While the widely-used continuous relaxation may achieve high learning efficiency, the pursued codes are typically less effective due to accumulated quantization error. In this work, we propose a novel binary code optimization method, dubbed *Discrete Proximal Linearized Minimization (DPLM)*, which directly handles the discrete constraints during the learning process. Specifically, the discrete (thus nonsmooth nonconvex) problem is reformulated as minimizing the sum of a smooth loss term with a nonsmooth indicator function. The obtained problem is then efficiently solved by an iterative procedure with each iteration admitting an *analytical* discrete solution, which is thus shown to converge very fast. In addition, the proposed method supports a large family of empirical loss functions, which is particularly instantiated in this work by both a supervised and an unsupervised hashing losses, together with the bits uncorrelation and balance constraints. In particular, the proposed DPLM with a supervised ℓ_2 loss encodes the whole NUS-WIDE database into 64-bit binary codes within 10 seconds on a standard desktop computer. The proposed approach is extensively evaluated on several large-scale datasets and the generated binary codes are shown to achieve very promising results on both retrieval and classification tasks.**

***Index Terms*—Binary code learning, Hashing, Discrete optimization**

I. Introduction

Binary coding (also known as hashing) has recently become a very popular research subject in information retrieval [8], [18], [23], [38], computer vision [24], [41], [43], machine learning [16], [36], *etc*. By encoding high-dimensional feature vectors (*e.g*., of documents, images, videos, or other types of data) to short hash codes, an effective hashing method is expected to accomplish efficient similarity search while preserving the similarities among original data to some extent. As a result, using binary codes to represent and search in massive data is a promising solution to handle large-scale tasks, owing to reduced storage space (typically several hundred binary bits per datum) and the low complexity of pairwise distance computations in a Hamming space.

The hashing techniques can be generally divided into two major categories: data-independent and data-dependent methods. Locality-Sensitive Hashing (LSH) [8] represents one large family of data-independent methods [4], [14], [13], [26], which generate hash functions via random projections. Although LSH is ensured to have high collision probability for similar data items, in practice LSH usually needs long hash bits and multiple hash tables to achieve both high precision and recall. The huge storage overhead may restrict its applications.

The other category, data-dependent or learning based hashing methods have witnessed a rapid development in the most recent years, due to the benefit that they can effectively and efficiently index and organize massive data with very compact binary codes. Different from LSH, data-dependent binary coding methods aim to generate short binary codes using the training data. A number of algorithms in this category have been proposed, including the unsupervised Spectral Hashing [36], [35], Binary Reconstructive Embedding (BRE) [12], PCA Hashing [33], Iterative Quantization (ITQ) [9], Circulant Binary Embedding (CBE) [39], Anchor Graph Hashing (AGH) [20], [22], Isotropic Hashing (IsoHash) [11], Inductive Manifold Hashing [29], Neighborhood Discriminant Hashing (NDH) [32], Binary Projection Bank (BPB) [19] *etc.*, and the supervised Minimal Loss Hashing (MLH) [25], Semi-Supervised Hashing (SSH) [33], Kernel-Based Supervised Hashing (KSH) [21], FastHash [17], Graph Cut Coding (GCC [7]), Supervised Discrete Hashing (SDH) [28] *etc*. The literature is comprehensive reviewed in [34] recently.

The binary constraints imposed on the target hash codes make the associated optimization problem very difficult to solve, which are generally NP-hard. To simplify the optimization, most of the methods in the literature adopt the following two-step way: first solve a relaxed problem by discarding the discrete constraints, and then quantize the obtained continuous solution to achieve the approximate binary solution. This two-step scheme significantly simplifies the original discrete optimization. Unfortunately, such an approximate solution is typically of low quality and often makes the resulting hash functions less effective. This is possibly due to the accumulated quantization error, which is especially the case when learning long-length codes. Iterative Quantization (ITQ) [9] is an effective approach to decrease the quantization distortion by applying an orthogonal rotation to projected training data. One limitation of ITQ is that it learns orthogonal rotations over pre-computed mappings (*e.g*., PCA or CCA) and the separate learning procedure usually makes ITQ suboptimal.

F. Shen, X. Zhou and Y. Yang are with School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (E-mail: {fumin.shen, johinfly, dlyyang}@gmail.com). Correspondence should be addressed to F. Shen.

J. Song is with University of Trento, Italy (E-mail: jingkuan.song@unitn.it).

H. T. Shen is School of Information Technology and Electrical Engineering, The University of Queensland, QLD 4072, Australia and with School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (E-mail: shenht@itee.edu.au).

D. Tao is with the Centre for Quantum Computation and Intelligent Systems and the Faculty of Engineering and Information Technology, University of Technology, Sydney, 81 Broadway Street, Ultimo, NSW 2007, Australia (E-mail: dacheng.tao@uts.edu.au).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

It would help generate more effective hashes to directly optimize the binary codes without continuous relaxations. However, the importance of discrete optimization in hashing has been less taken into account by most existing hashing methods. Recent efforts in this direction either lead to intractable optimization or are restricted to specific losses thus not easy to generalize. Very recently, binary optimization was studied in the unsupervised discrete graph hashing (DGH [20]) and promising results were obtained compared to previous relaxed methods. One disadvantage of DGH is that it suffers from an expensive optimization due to the involvement of singular value decomposition in each optimization iteration. In the meantime, supervised discrete hashing (SDH [28]) formulated supervised hashing as a linear classification problem with binary codes, where the associated key binary quadratic program (BQP) was efficiently solved by the discrete cyclic coordinate descent (DCC). However, DCC is limited to solving the standard BQP problem and it is still unclear how to apply DCC to other hashing problems with different objectives. For instance, DCC is not ready to optimize with the uncorrelation and balance constraints, which are widely-used in the hashing literature [36].

To overcome these problems, in this work, we propose a fast discrete optimization method for the general binary code learning problem. Our main contributions are summarized as follows:

1) The general binary code learning problem with discrete constraints is rewritten as an unconstrained minimization problem with an objective comprising two parts: a smooth loss function and a nonsmooth indicator function. The smooth function characterizes the learning loss of target binary codes with training data, while the nonsmooth one indicates the binary domain of the optimizing codes. The simple reformulation greatly simplifies the nonconvex nonsmooth binary code optimization problem.
2) We propose a novel discrete optimization method, termed **Discrete Proximal Linearized Minimization (DPLM)**, to learn binary codes in an efficient iterative way. In each optimization iteration, The corresponding subproblem admits an analytical solution by directly investigating the binary code space. As such, a high-quality discrete solution without resort to the continuous relaxation can eventually be obtained in an efficient computing manner, therefore enabling to tackle massive datasets.
3) Different from other discrete optimization solvers in the hashing literature, the proposed method supports a large family of empirical loss functions. In this work, this method is particularly instantiated by the supervised ℓ_2 loss and unsupervised graph hashing loss. The well-known bits uncorrelation and balance constraints are also investigated in the proposed optimization framework.
4) Comprehensive evaluations are conducted on several representative retrieval benchmarks, and the results consistently validate the superiority of the proposed methods over the state-of-the-art in terms of both efficiency and efficacy. In addition, we also show that the binary codes generated by our algorithm perform very well on the image and scene classification problems.

The rest of the paper is organized as follows. Section II elaborates the details of the proposed DPLM method, which is instantiated by both a supervised and an unsupervised hashing objective in Section III, followed by the exploration of bits uncorrelation and balance constraints. In Section IV, we analyze the proposed discrete algorithm with comparison to the relaxed method and other optimization approach. In Section V, we evaluate our algorithm on several real-world large-scale datasets for both retrieval and classification tasks, followed by the conclusion of this work in Section VI.

II. Fast Binary Optimization for Hashing

Let us first introduce some notations. We denote matrices as boldface uppercase letters like $\mathbf{X}$, vectors as boldface lowercase letters like $\mathbf{x}$ and scalars as x. The $r \times r$ identity matrix is denoted as $\mathbf{I}_r$, and the vector with all ones and zeros as $\mathbf{1}$ and $\mathbf{0}$, respectively. We abbreviate the Frobenius norm $||\cdot||_F$ as $||\cdot||$ in this paper. ∇f denotes the gradient of function $f(\cdot)$. $\mathrm{sgn}(\cdot)$ is the sign function with output $+1$ for positive numbers and -1 otherwise. For binary codes, we use (1, -1) bits for mathematical derivations, and use (1, 0) bits for implementations of all referred binary coding and hashing algorithms.

A. The binary code learning problem

Suppose we have n samples $\mathbf{x}_i \in \mathbb{R}^d, i = 1, \cdots, n$, stored in matrix $\mathbf{X} \in \mathbb{R}^{d \times n}$. For each sample $\mathbf{x}$, we aim to learn its r-bit binary code $\mathbf{b} \in \{-1, 1\}^r$. We consider the following general binary code learning problem

$$\begin{aligned} &\min_{\mathbf{B}} \ \mathcal{L}(\mathbf{B}) \\ &\text{s.t.} \ \ \mathbf{B} \in \{-1, 1\}^{r \times n}. \end{aligned} \tag{1}$$

Here $\mathbf{B}$ is the target binary codes for $\mathbf{X}$ and $\mathcal{L}(\cdot)$ is the smooth loss function. In this work, we aim at a scalable and computationally tractable method which can be applied to a large family of loss functions $\mathcal{L}(\cdot)$.

The binary constraints make problem (1) a mixed-integer optimization problem, which is generally NP-hard. Most previous methods resort to the continuous relaxation by discarding the discrete constraints. As aforementioned, however, this relaxed solution may cause large error accumulation as the code length increases. This is mainly because the discrete constraints have not been treated adequately during the learning procedure, as shown in [20], [28].

B. Discrete Proximal Linearized Minimization

In this section, we shown problem (1) can be solved in an efficient way while keeping the discrete variables in the optimization. To simplify the discrete optimization in problem (1), let us first introduce the following *indicator function*

$$\delta_C(\mathbf{B}) = \begin{cases} 0 & \text{if } \mathbf{B} \in C \\ +\infty & \text{otherwise,} \end{cases} \tag{2}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

where C is a nonempty and closed set. Let $\mathbb{B}$ denotes the binary codes space $\{-1,1\}^{r\times n}$. The function $\delta_{\mathbb{B}}(\mathbf{B})$ yields infinity as long as one entry of $\mathbf{B}$ does not belong to the binary domain $\{-1,1\}$. With the indicator function, we are safe to rewrite problem (1) to an unconstrained minimization problem

$$\min_{\mathbf{B}} \mathcal{L}(\mathbf{B}) + \delta_{\mathbb{B}}(\mathbf{B}). \tag{3}$$

The simple reformulation of problem (1) to (3) greatly simplify the optimization therein, as shown below. The objective of (3) consists of two parts: a smooth function and a nonsmooth one. The smooth function $\mathcal{L}(\mathbf{B})$ models the hashing loss which can be chosen freely according to different problem scenarios, while the nonsmooth function $\delta_{\mathbb{B}}(\mathbf{B})$ indicates the domain of the optimizing hash codes.

Solving problem (3) is still nontrivial due to the involvement of nonsmooth indicator. Inspired by the recent advance in nonconvex and nonsmooth optimization [1], [2], we solve problem (3) with the following iterative procedure. Denote $\mathrm{Prox}_{\lambda}^{f}$ the *proximal operator* with function f and parameter λ:

$$\mathrm{Prox}_{\lambda}^{f}(x) = \arg\min_{y} f(y) + \frac{\lambda}{2}||y-x||^2. \tag{4}$$

Suppose we have obtained the code solution $\mathbf{B}^{(j)}$ at the j_{th} iteration for problem (3). At the $(j+1)_{\mathrm{th}}$ iteration, $\mathbf{B}$ is updated by

$$\mathbf{B}^{(j+1)} = \mathrm{Prox}_{\lambda}^{\delta}\big(\mathbf{B}^{(j)} - \frac{1}{\lambda}\nabla\mathcal{L}(\mathbf{B}^{(j)})\big) \tag{5}$$

$$= \arg\min_{\mathbf{B}} \delta(\mathbf{B}) + \frac{\lambda}{2}||\mathbf{B} - \mathbf{B}^{(j)} + \frac{1}{\lambda}\nabla\mathcal{L}(\mathbf{B}^{(j)})||^2. \tag{6}$$

The optimization procedure with (5) is also known as the forward-backward splitting algorithm [1]. The forward-backward splitting scheme for minimizing the sum of a smooth function $\mathcal{L}(\cdot)$ with a nonsmooth one can simply be viewed as the proximal regularization of $\mathcal{L}(\cdot)$ linearized at a given point $\mathbf{B}^{(j)}$.

By transforming the indicator function back to the binary constraints, solution (6) leads to the following problem

$$\begin{aligned} \min_{\mathbf{B}} \quad & ||\mathbf{B} - \mathbf{B}^{(j)} + \frac{1}{\lambda}\nabla\mathcal{L}(\mathbf{B}^{(j)})||^2, \\ \text{s.t.} \quad & \mathbf{B} \in \{-1,1\}^{r\times n}. \end{aligned} \tag{7}$$

Remark 1. *The derivation of problem* (5) *and* (7) *is the key step of our algorithm. By looking at* (7)*, we can see that the problem actually seeks the projection of* $\mathbf{B}^{(j)} - \frac{1}{\lambda}\nabla\mathcal{L}(\mathbf{B}^{(j)})$ *onto the binary code space. Indeed, for the indicator function* $\delta(\mathbf{B})$ *(of the nonempty and closed set* $\mathbb{B}$*), its proximal map* $\mathrm{Prox}_{\lambda}^{\delta}(\mathbf{X})$ *reduces to the projection operator:*

$$P_{\mathbb{B}}(\mathbf{X}) = \arg\min\{||\mathbf{B}-\mathbf{X}||^2 : \mathbf{B}\in\mathbb{B}\}. \tag{8}$$

It is clear that, problem (7) has the analytical solution

$$\mathbf{B}^{(j+1)} = \mathrm{sgn}(\mathbf{B}^{(j)} - \frac{1}{\lambda}\nabla\mathcal{L}(\mathbf{B}^{(j)})). \tag{9}$$

We term this optimization method as **Discrete Proximal Linearized Minimization (DPLM)** due to the involvement of discrete variables compared to the linearized proximal method [1].

In the following, we show that for problem (1) the algorithm (9) converges to a critical point. First we introduce the convergence theorem from [1] for the nonconvex gradient projection method.

Theorem 2 ([1])**.** *Let* $f:\mathbb{R}^n\to\mathbb{R}$ *be a differentiable function whose gradient is L-Lipschitz continuous, and* C *a nonempty closed subset of* $\mathbb{R}^n$*. Being given* $\epsilon\in(0,\frac{1}{2L})$ *and a sequence of stepsizes* γ_k *such that* $\epsilon < \gamma_k < \frac{1}{L} - \epsilon$*, we consider a sequence* (x^k) *that complies with*

$$x^{k+1} \in P_C(x^k - \gamma_k\nabla f(x^k)), \text{with } x^0\in C.$$

If the function $f+\delta_C$ *is a Kurdyka-Lojasiewicz (KL) function and if* (x_k) *is bounded, then the sequence* (x_k) *converges to a point* $x^\star$ *in* C*.*

Corollary 3. *Assume the loss function* $\mathcal{L}$ *is a* C^1 *(continuously differentiable) semi-algebraic function whose gradient is L-Lipschitz continuous. By choosing a proper sequence of parameters of* λ*, the sequence* $\mathbf{B}^{(j)}$ *generated by the proposed DPLM algorithm with* (9) *converges to a critical point* $\mathbf{B}^\star$*.*

Proof: This assumption ensures that the objective of (3) $\mathcal{L}(\cdot)+\delta_{\mathbb{B}}(\mathbf{B})$ is a KL function [1]. It is obvious that the sequence $\mathbf{B}^{(j)}$ generated by (9) is bounded in $\mathbb{B}$. Based on Theorem 2, by choosing the parameter λ greater than the Lipschitz constant L, the DPLM algorithm converges to some critical point. ■

Remark 4. *The requirement of the presented DPLM method is only mild. The KL assumption of the objective function is very general that* $\mathcal{L}$ *being the smooth polynomial is a typical instance. The empirical convergence of DPLM is referred to Section IV-C.*

Till now, we have presented the key optimization method for learning binary codes with a general loss function. The optimization procedure is outlined in Algorithm 1. Despite its simplicity, the proposed method can obtain very high-quality codes for the retrieval and classification tasks, as shown in our experiments.

In addition, due to the analytical solution at each iteration, this method enjoys very fast optimization. We note that the analytical solution does not depend on a specific loss function. In Section III, we will discuss the application of DPLM to different hashing losses, such as supervised ℓ_2 hashing and unsupervised graph hashing. We will also show the well-known bits uncorrelation balance constraints can be easily incorporated in the binary code optimization with the proposed method.

C. Hash function learning

The above method describes the learning procedure for generating binary codes $\mathbf{B}$ for training data $\mathbf{X}$. For a novel query $\mathbf{x}\in\mathbb{R}^d$, we need a hash function to efficiently encode $\mathbf{x}$ into binary code. We here adopt the simple linear hash function $h(\mathbf{x}) = \mathrm{sgn}(\mathbf{P}^\top\mathbf{x})$, which is learned by solving a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

Algorithm 1 Discrete Proximal Linearized Minimization

Input: Training data $\mathbf{X}$; code length r; maximum iteration number t; parameters λ.

Output: Binary codes $\mathbf{B} \in \{-1, 1\}^{r\times n}$; hash function $h(\mathbf{x})$.

1) Initialize $\mathbf{B}$ by the sign of random Gaussian matrix;
2) Loop until converge or reach maximum iterations:
   - Calculate the gradient $\nabla\mathcal{L}$;
   - Update $\mathbf{B}$ by $\mathrm{sgn}(\mathbf{B} - \frac{1}{\lambda}\nabla\mathcal{L}(\mathbf{B}))$;
3) Compute hash function $h(\mathbf{x}) = \mathrm{sgn}(\mathbf{P}^\top\mathbf{x})$ with obtained $\mathbf{B}$ by (10).

linear regression system with the available training data and codes. That is

$$\min_{\mathbf{P}} ||\mathbf{B} - \mathbf{P}^\top\mathbf{X}||^2, \tag{10}$$

which is clearly solved by $\mathbf{P} = (\mathbf{X}\mathbf{X}^\top)^{-1}\mathbf{X}\mathbf{B}^\top$. This hash function learning scheme has been widely used, such as in [28][42].

III. Case studies of the hashing problems

In this section, we investigate different hashing problems with the proposed DPLM method, where both the supervised and unsupervised losses are studied.

A. Supervised hashing

We adopt the ℓ_2 loss in the supervised setting, where the learned binary codes are assumed to be optimal for linear classification. The learning objective writes,

$$\begin{aligned}\mathcal{L}(\mathbf{B}, \mathbf{W}) &= \frac{1}{2}\sum_{i=1}^{n} ||\mathbf{y}_i - \mathbf{W}^\top\mathbf{b}_i||^2 + \delta||\mathbf{W}||^2 \\ &= \frac{1}{2}||\mathbf{Y} - \mathbf{W}^\top\mathbf{B}||^2 + \delta||\mathbf{W}||^2. \end{aligned}\tag{11}$$

Here $\mathbf{Y} \in \mathbb{R}^{k\times n}$ stores the labels of of training data $\mathbf{X} \in \mathbb{R}^{d\times n}$, with its (i, j)-th entry $\mathbf{Y}_{ij} = 1$ if the j-th sample $\mathbf{x}_j$ belongs to the i-th of the total k classes and 0 otherwise. Matrix $\mathbf{W}$ is the classification matrix which is jointly learned with the binary codes. δ is the regularization parameter. The above simple objective has been shown to achieve very promising results recently [28].

With the ℓ_2 loss, the binary codes can be easily computed by the DPLM optimization method as shown in Section II-B. Given $\mathbf{W}$, the key step is updating $\mathbf{B}$ by (9) with the following gradient

$$\nabla\mathcal{L}(\mathbf{B}) = \mathbf{W}\mathbf{W}^\top\mathbf{B} - \mathbf{W}\mathbf{Y}. \tag{12}$$

With $\mathbf{B}$ obtained, the classification matrix $\mathbf{W}$ is efficiently computed by $\mathbf{W} = (\mathbf{B}\mathbf{B}^\top)^{-1}\mathbf{B}\mathbf{Y}^\top$. The whole optimization alternatively runs over variable $\mathbf{B}$ and $\mathbf{W}$. In practice, we simply initialize $\mathbf{B}$ by the sign of random Gaussian matrix, and $\mathbf{W}$ and $\mathbf{B}$ are then updated accordingly. The optimization typically converges within 5 iterations.

B. Unsupervised graph hashing

For the unsupervised setting, we investigate the well-known graph hashing problem [36], which has been extensively studied in the literature [36][22][29]. The unsupervised graph hashing optimizes the following objective

$$\mathcal{L}(\mathbf{B}) = \frac{1}{2}\sum_{i,j=1}^{n} ||\mathbf{b}_i - \mathbf{b}_j||^2\mathbf{A}_{ij} \tag{13}$$

$$= \frac{1}{2}\mathrm{tr}(\mathbf{B}\mathbf{L}\mathbf{B}^\top), \tag{14}$$

where $\mathbf{A}$ is the affinity matrix computed with $\mathbf{A}_{ij} = \exp(-||\mathbf{x}_i - \mathbf{x}_j||^2/\sigma^2)$ and σ is the bandwidth parameter. $\mathbf{L}$ is the associated Laplacian matrix $\mathbf{L} = \mathrm{diag}(\mathbf{A}\mathbf{1}) - \mathbf{A}$.

To tackle this challenging problem, Spectral Hashing [36] additionally assumes that data are sampled from a uniform distribution, which leads to a simple analytical eigenfunction solution of 1-D Laplacians. However, the strong assumption can hardly be true in practice. AGH [22] employs the anchor graph to facilitate constructing affinity matrix and learning hash functions analytically. IMH [29] learns Laplacian eigenmaps on a small data subset and the hash codes are thus inferred with a linear combination of the base points. All these methods apply spectral relaxation to simplify the optimization by discarding the binary constraints.

Different from these methods, we optimize the graph hashing problem by DPLM directly over the binary variables. With the gradient of (13) as

$$\nabla\mathcal{L}(\mathbf{B}) = \mathbf{B}\mathbf{L}, \tag{15}$$

the optimization is performed by updating variable $\mathbf{B}$ in each iteration with

$$\mathbf{B}^{(j+1)} = \mathrm{sgn}(\mathbf{B}^{(j)} - \frac{1}{\lambda}\mathbf{B}^{(j)}\mathbf{L}).$$

Note that the computation of affinity matrix $\mathbf{A}$ dominates the optimization and is $O(dn^2)$ in time complexity. In practice, we adopt the anchor graph to compute $\mathbf{A} = \mathbf{Z}\mathbf{Z}^\top$ with $\mathbf{Z} \in \mathbb{R}^{n\times m}$ as in [22], which is $O(dnm)$ with m anchors. The gradient (15) is thus computed by $\mathbf{B}\mathrm{diag}(\mathbf{A}\mathbf{1}) - \mathbf{B}\mathbf{A}$ which is $O(rmn)$.

C. Bits uncorrelation and balance

The bits uncorrelation and balance constraints have been widely used in previous hashing methods. With these two constraints, problem (1) is rewritten as

$$\begin{aligned}\min_{\mathbf{B}}\ & \mathcal{L}(\mathbf{B}) \\ \text{s.t.}\ & \mathbf{B}\mathbf{B}^\top = n\mathbf{I}_r, \\ & \mathbf{B}\mathbf{1} = \mathbf{0}, \\ & \mathbf{B} \in \{-1, 1\}^{r\times n}.\end{aligned}\tag{16}$$

The first two constraints force the binary codes to be uncorrelated and balanced, respectively. They are two key features of compact binary code learning [36]. A large family of existing hashing algorithms can be seen as the instances of this general model, such as unsupervised graph hashing [36], [22], [20], and supervised hashing [7]. However, these two constraints often make the hashing problem computationally intractable,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

especially with the binary constraints. The recent proposed discrete optimization solver [28] discards these constraints for algorithm feasibility.

We rewrite (16) as follows

$$\min_{\mathbf{B}} \mathcal{L}(\mathbf{B}) + \frac{\mu}{4}||\mathbf{B}\mathbf{B}^\top||^2 + \frac{\rho}{2}||\mathbf{B}\mathbf{1}||^2 \quad \text{s.t. } \mathbf{B} \in \{-1, 1\}^{r\times n}. \tag{17}$$

Note that $||\mathbf{B}\mathbf{B}^\top - n\mathbf{I}_r||^2 = ||\mathbf{B}\mathbf{B}^\top||^2 + const$. With sufficiently large parameters $\mu > 0$ and $\rho > 0$, problems (16) and (17) will be equivalent. Denoting the objective of (17) as $g(\mathbf{B})$, its gradient is given by

$$\nabla g(\mathbf{B}) = \nabla \mathcal{L}(\mathbf{B}) + \mu\mathbf{B}\mathbf{B}^\top\mathbf{B} + \rho\mathbf{B}\mathbf{1}\mathbf{1}^\top. \tag{18}$$

With this, the binary optimization is conducted by updating variable $\mathbf{B}$ in each iteration with

$$\mathbf{B}^{(j+1)} = \text{sgn}(\mathbf{B}^{(j)} - \frac{1}{\lambda}\nabla g(\mathbf{B}^{(j)}). \tag{19}$$

In Section IV, we will explore the impact of these two binary codes properties for both the supervised and unsupervised hashing problems studied in this work.

D. Complexity study

In this part, we discuss the computational complexity of our algorithm. For the supervised method in Section III-A, the main step is updating $\mathbf{B}$ by computing its gradient $\mathbf{W}\mathbf{W}^\top\mathbf{B} - \mathbf{W}\mathbf{Y}$, for which the time complexity is $O(r^2k + r^2n + rkn)$, thus making the total time complexity of updating $\mathbf{B}$ be $O(t(r^2k + r^2n + rnk))$, where t is the maximum iteration number during the $\mathbf{B}$ updating step. The complexity of updating $\mathbf{W}$ is $O(r^3 + rkn + r^2k)$. Therefore, the total time complexity of the supervised algorithm is $O(T(tr^2n + rkn)$ with T iterations (updating $\mathbf{W}$ and $\mathbf{B}$).

The unsupervised algorithm in Section III-B comprises two components: the anchor graph construction and binary code learning. As mentioned in III-B, the first part costs $O(dnm)$ time and the second part costs $O(trmn)$ with t iterations.

The time complexity of computing the hash functions with equation (10) is $O(d^3 + 2d^2n)$. As for these algorithms with bit uncorrelation and balance constraints, the additional computation is due to the computing of $\mu\mathbf{B}\mathbf{B}^\top\mathbf{B} + \rho\mathbf{B}\mathbf{1}\mathbf{1}^\top$ in (18) in each iteration of updating $\mathbf{B}$, which is $O(r^2n + 2rn)$ in time.

To summarize, the training time complexities for the proposed supervised and unsupervised algorithms are both linear as the data size n. For a novel query, the predicting time with hash function $h(\mathbf{x})$ is $O(dr)$ which is independent of n.

IV. Algorithm Analysis

In this section, we evaluate the proposed method from the following aspects: the impact of bits uncorrelation and balance, the optimization performance of DPLM compared to other solvers.

A. The impact of bits uncorrelation and balance

We first evaluate the impact of the two well-known constraints on binary codes: bits uncorrelation and balance. Both the supervised loss and unsupervised loss are evaluated. The performance of our method with or without each of the constraints is shown in Table I. The database of NUS-WIDE and CIFAR-10 are used for evaluation. As we can see, the constraints play important roles in binary code learning. For the two hashing losses, better results are obtained by imposing both these constraints than discarding them or keeping only one in most cases. The ability to incorporate these two constraints into binary code optimization is one of the advantages of our method over other discrete methods such as DCC [28]. We also show in details the impacts of parameters μ and ρ with both the ℓ_2 supervised loss (11) and unsupervised graph hashing loss (13). The MAP results with varying μ and ρ on NUS-WIDE are are shown in Figure 1.

B. DPLM vs. the relaxed method

One may be interested in how the proposed DPLM method performs compared to the relaxed method: first relaxing the original discrete problem to a continuous one and then rounding the resultant solution. In this part, we compare DPLM and the relaxed method for both the supervised ℓ_2 loss and unsupervised graph hashing loss. The mAP and Precision500 results are shown in Figure 2 with code length varying from 32 to 128 bits.

From Figure 2, large performance gains are clearly observed with DPLM over the relaxed approach for both of the two hashing losses. The superior results demonstrate the effectiveness of our optimization method and the importance of discrete optimization for binary code learning or hashing problems. That is, it would be preferred to directly pursue the discrete codes in the binary space without continuous relaxations, provided that scalable and tractable solvers are accessible. In the next part, we will evaluate the convergence speed of DPLM.

C. DPLM vs. DCC

One main contribution of this work is the fast binary optimization method for hashing. In this part, we compare the optimization speed between the proposed Discrete Proximal Linearized Minimization (DPLM) and the recent discrete cyclic coordinate descent (DCC) method [28]. To be fair, we omit the uncorrelation and balance constraints which DCC cannot handle. That is, both DPLM and DCC both minimize the following objective function and are compared according to the obtained optimal solutions:

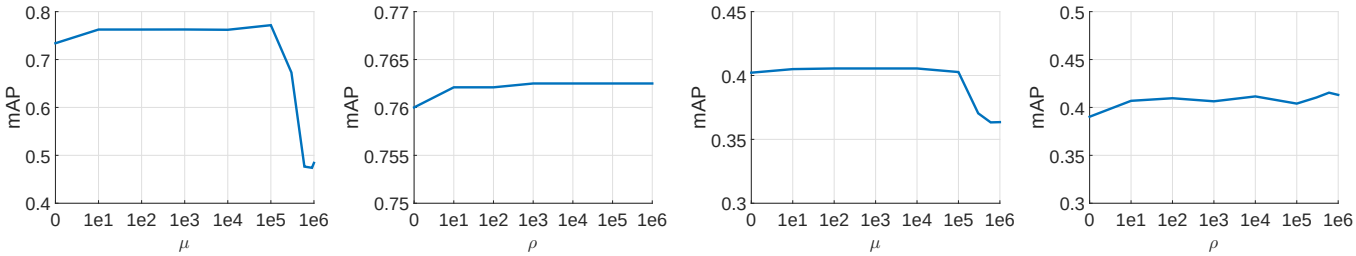
$$\min_{\mathbf{B},\mathbf{W}} \frac{1}{2}||\mathbf{Y} - \mathbf{W}^\top\mathbf{B}||^2 + \delta||\mathbf{W}||^2 \quad \text{s.t. } \mathbf{B} \in \{-1, 1\}^{r\times n}.$$

The objective value as a function of optimization time is shown in Figure 3. We can clearly see that both these two methods converge to similar objective value. However, DPLM obtains much faster convergence than DCC. DPLM only costs

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

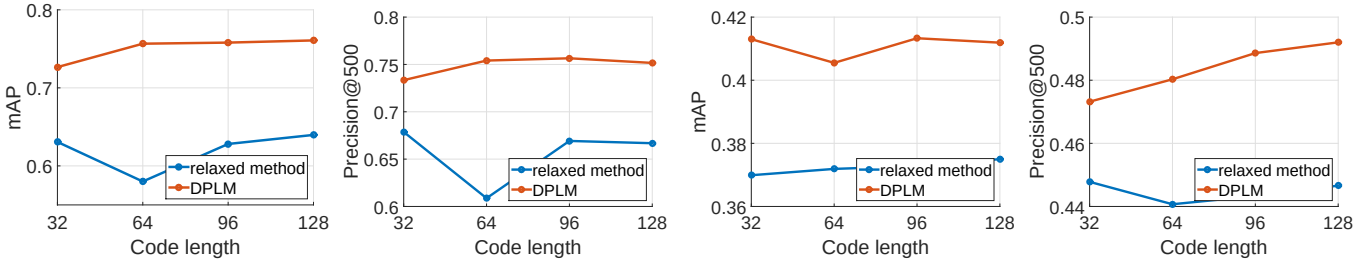
TABLE I: Evaluation of our method with/without the constraint of balance or uncorrelation on the image retrieval task. −: this operation is not applied; ✓: applied. Both the supervised and unsupervised losses are tested. The results are reported in mAP and Precision of top 500 retrieved samples with 64 and 128 bits. The database of NUS-WIDE and CIFAR-10 are used, where the descriptions can be found in Section V-A and [28], respectively.

NUS-WIDE									
Balance	Uncorrelation	Supervised				Unsupervised			
		mAP		Precision@500		mAP		Precision@500	
		64 bits	128 bits	64 bits	128 bits	64 bits	128 bits	64 bits	128 bits
−	−	0.6955	0.7167	0.6079	0.6041	0.3719	0.3750	0.4407	0.4466
✓	−	0.7527	0.7600	0.7524	0.7543	0.3903	0.3849	0.4414	0.4506
−	✓	0.734	0.7580	0.7523	**0.7598**	0.4022	0.4087	0.4596	0.4752
✓	✓	**0.7603**	**0.7610**	**0.7545**	0.7554	**0.4055**	**0.4119**	**0.4803**	**0.4920**
CIFAR-10									
Balance	Uncorrelation	Supervised				Unsupervised			
		mAP		Precision@500		mAP		Precision@500	
		64 bits	128 bits	64 bits	128 bits	64 bits	128 bits	64 bits	128 bits
−	−	0.6809	0.7029	0.6129	0.6286	0.1786	0.1920	0.2377	0.2670
✓	−	0.6842	0.7012	0.6133	0.6298	0.1872	0.2112	0.2596	0.3285
−	✓	0.6821	0.7039	0.6140	0.6302	0.1820	0.1927	0.2441	0.2738
✓	✓	**0.6852**	**0.7046**	**0.6155**	**0.6365**	**0.1960**	**0.2117**	**0.2724**	**0.3303**



(a) Supervised hashing (b) Unsupervised hashing

Fig. 1: The function of MAP w.r.t. parameters μ and ρ with (a) the ℓ_2 supervised loss (11) and (b) unsupervised graph hashing loss (13), respectively. The evaluation is performed on NUS-WIDE. 64 bits are used.



(a) Supervised hashing (b) Unsupervised hashing

Fig. 2: Comparison of DPLM and relaxed optimization with (a) the ℓ_2 supervised loss (11) and (b) unsupervised graph hashing loss (13), respectively. The evaluation is performed on NUS-WIDE in terms of MAP and Precision@500.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

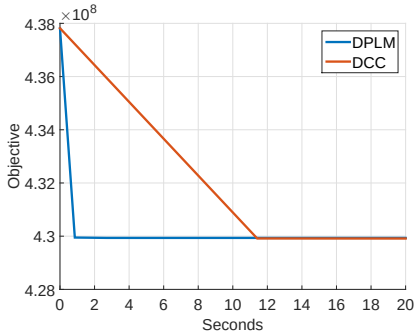


Fig. 3: Objective value as a function of binary code training time for DPLM and DCC on NUS-WIDE. We use 64 bits in this experiment.

about 1 second to achieve the convergence. This is mainly because DPLM updates all bits at the same time in each iteration. In contrast, DCC computes the codes in a bit-by-bit manner, where each bit is computed based on the previously updated bits. In the next section, we will extensively compare the two algorithms on both the retrieval and classification tasks.

The advantages of DPLM over DCC is summarized as the following points: 1) DPLM is much more efficient than DCC for the binary optimization problem, as shown in Figure 3; 2) DPLM is developed to solve the general binary code learning problem while DCC can only solve the BQP problem and cannot handle the bit uncorrelation constraints; 3) By adopting the bit balance and uncorrelation constraints, DPLM can achieve better performance than DCC, as shown in Table II.

V. Experiments

In this section, extensive experiments are conducted to evaluate the proposed hashing methods in both computational efficiency and retrieval or classification performance. We test our method on three large-scale public databases, i.e., SUN397 [37], NUS-WIDE [3] and ImageNet [5]. The detailed descriptions of these databases are introduced in the corresponding subsection. Several state-of-art hashing methods are taken into comparison, including the supervised MLH [25], CCA-ITQ [9], KSH [21], FastHash [17], SDH [28] and the unsupervised SH [36], AGH [22], PCA-ITQ [9], and IMH [29]. For these methods, we employ the implementations and suggested parameters provided by the authors. For our method, since the bits uncorrelation and balance constraints help produce better codes, we impose these two constraints in our algorithm. We empirically set $\lambda = 0.1$, $\mu = 1e3$ and $\rho = 1e2$. Since CCA-ITQ, SDH and our methods can efficiently handle large data during training, we use all the available training data for training. For KSH, MLH and FastHash, we learn these models with 50k training samples due to the large computational costs of these methods.

For the retrieval experiments, we report the compared results in terms of mean average precision (mAP), mean precision of the top 500 retrieved neighbors (Precision@500) and the precision and recall curves. Note that we treat a query a false case if no point is returned when calculating precisions. Ground truths are defined by the category information from the datasets. For computational efficiency, we compare the training and testing time of the evaluated methods. For the compared supervised hashing approaches, we also test the performance of these methods on the classification task, where the metric of classification accuracy is used. If not otherwise specified, the experiments are conducted with MATLAB implementations on a standard PC with an Intel 6-core 3.50GHz CPU and 64G RAM.

A. NUS-WIDE: Retrieval with multi-labeled data

The NUS-WIDE database contains about 270,000 images collected from Flickr. The images in NUS-WIDE are associated with 81 concepts, with each image containing multiple semantic labels. We define the true neighbors of a query as the images sharing at least one labels with the query image. The provided 500-dimensional Bag-of-Words features are used. we collect the 21 most frequent label for test. For each label, 100 images are uniformly sampled for the query set and the remaining images are for the training set. The results in terms of retrieval performance (mAP and Precision@500) and training/testing time efficiency are reported in Table II.

It is clear from Table II that our approach achieves the best results in terms of both mAP and precision among all the compared supervised methods. In particular, with 64 bits, our method outperforms the best of all other methods (obtained by SDH) by more than 6% and 15% in terms of mAP and precision, respectively. The precision-recall and precision curves of these compared methods with 32 to 128 bits are shown in Figure 4. Our method consistently outperforms all other methods by large margins in all situations.

We also evaluate these methods in terms of training and testing efficiencies. We can clearly see from Table II that, our method costs less training time than all other compared methods. Specifically, DPLM only consumes only about 8.3 seconds to train the hashing model on the NUS-WIDE database. CCA-ITQ also has a high computational efficiency, which is much faster than other methods. In terms of testing time (encoding a query image into binary code), our method together with SDH and CCA-ITQ run very fast on the same scale while FastHash suffers from a slow encoding speed.

B. ImageNet: Retrieval with large-scale high dimensional features

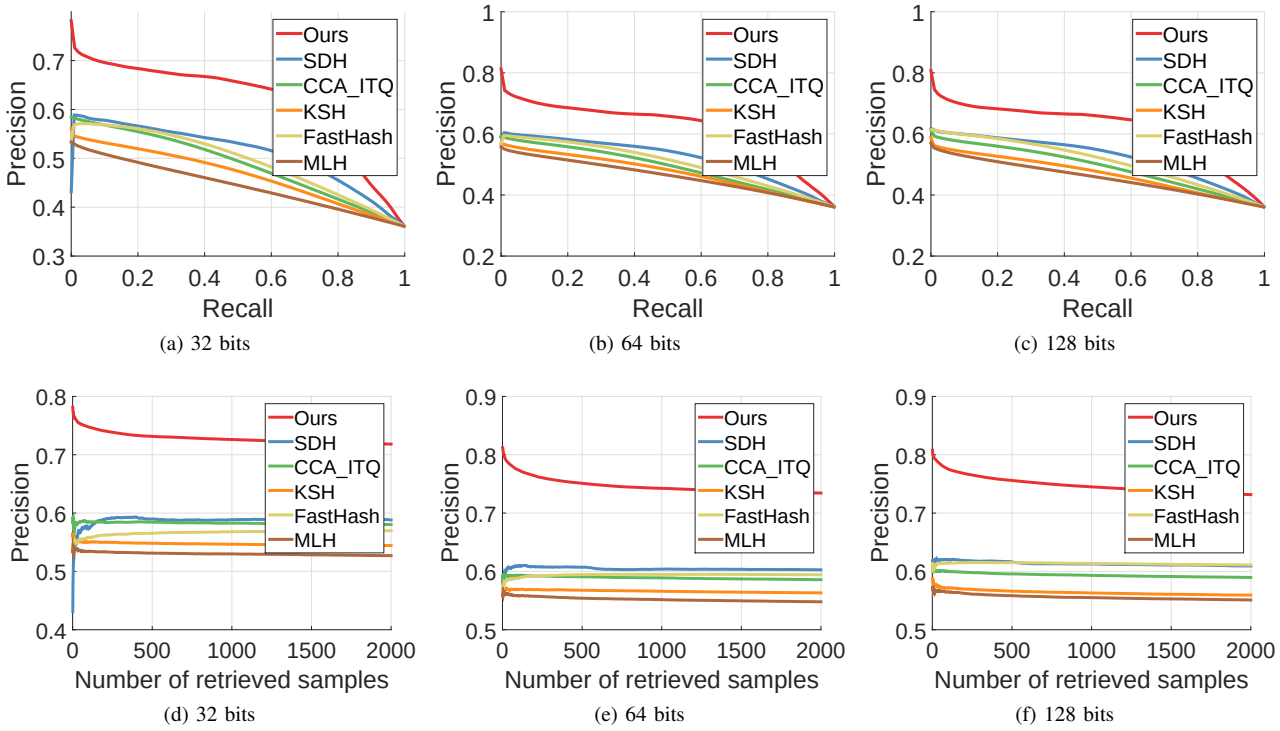
As a subset of ImageNet [5], the large dataset ILSVRC 2012 contains over 1.2 million images of totally 1,000 categories. We use the provided training set as the retrieval database and 50,000 images from the validation set as the query set. We extract the feature for each image by the convolutional neural networks (CNN) model as a 4096D vector. The results are reported in the Table III.

As in the last section, similar results are observed from Table III that our method obtains the best results. On this large dataset our method is slightly better than SDH, while both of them outperforms other methods by even larger gaps than on the relatively smaller NUS-WIDE database. These results

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

TABLE II: Results in term of mAP and mean precision of the top 500 retrieved neighbors (precision@500) of the compared supervised methods on the **NUS-WIDE** database with 64 and 128 bits, respectively. The training and testing time are reported in seconds.

Method	mAP		Precision@500		Training time (s)		Testing time (s)	
	64 bits	128 bits	64 bits	128 bits	64 bits	128 bits	64 bits	128 bits
Ours	**0.7603**	**0.7610**	**0.7545**	**0.7547**	**8.32**	**13.97**	1.31e-6	**2.47e-6**
SDH	0.6955	0.7167	0.6079	0.6041	34.72	119.8	1.65e-6	2.55e-6
CCA-ITQ	0.6232	0.6239	0.5919	0.5962	8.93	19.71	**1.08e-6**	3.03e-6
KSH	0.6091	0.6129	0.5638	0.5659	2092.3	4384.3	5.84e-6	1.33e-5
FastHash	0.5346	0.5507	0.6013	0.6197	3486.16	7091.99	1.31e-2	2.86e-2
MLH	0.4726	0.4689	0.5540	0.5583	8413.4	13791	2.23e-5	3.79e-5



(a) 32 bits (b) 64 bits (c) 128 bits

(d) 32 bits (e) 64 bits (f) 128 bits

Fig. 4: (Top) Precision-Recall and (Bottom) Precision curves with top 2000 retrieved images of the compared methods on **NUS-WIDE**.

demonstrate the importance of discrete optimization for binary code learning.

In addition, our method demonstrates clearer advantages on ImageNet in training efficiency. For example, our method trains on the whole dataset with only about 5.5 minutes with 128 bits, while SDH costs more than 1 hour and KSH, FastHash and MLH runs even slower. From these experiments, it is clear that *our discrete hashing method can generate more effective binary codes with much less learning time*.

C. SUN397: Scene classification with binary codes

In this part, we test the compared hashing methods on the classification task by feeding the classifier with the generated binary feature with these methods. The LIBLINEAR implementation of linear SVM is used here. The proposed approach is compared with several other supervised hashing methods including SDH, CCA-ITQ, KSH, FastHash and MLH. SUN397 is a widely-used scene classification benchmark, which contains about 108,000 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 [10]. In this experiment, 100 images are sampled uniformly randomly from each of the 18 largest scene categories to form a test set of 1,800 images and the rest for training set. The results are reported in Table IV.

As can be clearly seen, the proposed approach obtains the highest classification accuracies on this dataset. Clear advantage of our method is shown over SDH especially with short code length. With long code lengths (128 bits), our method achieves very close results with SDH, while outperforms other methods by more than 5% accuracies.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

TABLE III: Results in term of mAP and mean precision of the top 500 retrieved neighbors (precision@500) of the compared methods on the **ImageNet** database with 64 and 128 bits, respectively. The training and testing time are reported in seconds. The experiments are conducted on a workstation with an Intel 6-core 2.10GHz CPU and 188G RAM.

Method	mAP		Precision@500		Training time (s)		Testing time (s)	
	64 bits	128 bits	64 bits	128 bits	64 bits	128 bits	64 bits	128 bits
Ours	**0.3235**	**0.4310**	**0.3039**	**0.3996**	**258.82**	**336.84**	**2.80e-6**	**6.93e-6**
SDH	0.3225	0.4261	0.3016	0.3987	1568.66	4015.20	3.01e-6	8.24e-6
CCA-ITQ	0.1086	0.1694	0.1651	0.2461	372.52	468.00	3.42e-6	7.64e-6
KSH	0.0897	0.1351	0.1702	0.2381	6953.48	13897.53	6.18e-5	9.67-05
FastHash	0.1062	0.1827	0.1944	0.2598	3486.16	7091.99	1.59e-2	1.31e-2
MLH	0.0739	0.1111	0.1493	0.2069	20864.12	43494.35	1.18e-5	2.74e-5

TABLE IV: Classification accuracy (%) on **SUN397** with the produced binary codes by different hashing methods. The code length varies from 32 to 128 bits.

Method	Accuracy (%)			
	32 bits	64 bits	96 bits	128 bits
Ours	**66.83**	**76.72**	**77.67**	**78.72**
SDH	63.00	75.28	77.44	78.22
CCA-ITQ	60.06	70.50	73.50	73.67
KSH	63.06	68.83	70.44	73.17
FastHash	64.95	71.04	74.17	75.38
MLH	56.56	65.67	68.78	71.44

TABLE V: Classification accuracy (%) on **ImageNet** with the produced binary codes by different hashing methods. The code length varies from 32 to 128 bits. The experiments are conducted on a workstation with an Intel 6-core 2.10GHz CPU and 188G RAM.

Method	Accuracy (%)			
	32 bits	64 bits	96 bits	128 bits
Ours	**18.28**	**30.50**	**36.59**	**39.89**
SDH	18.19	30.12	36.03	39.85
CCA-ITQ	10.66	18.31	23.84	27.42
KSH	18.07	27.58	31.62	32.69
FastHash	17.69	28.04	34.36	36.72
MLH	18.00	27.83	33.02	35.70

D. ImageNet: Image classification with binary codes

In this subsection, we test the classification performance of the learned binary codes on the ImageNet benchmark [5]. The same training/test setting is used as in Section V-B. The classification accuracies on this dataset are reported in Table V. Our method performs slightly better than SDH on this dataset (with much lower learning cost however), while much better than all other methods. The results in Table IV and Table V clearly show that the binary codes generated by our methods work very well on the classification problem as well as the retrieval task.

E. Comparison with unsupervised methods

In this part, we evaluate our method in the unsupervised setting by performing DPLM with the unsupervised graph hashing loss. Other representative methods of graph hashing including SH, AGH (with one or two layers), IMH with Laplacian Eigenmaps (denoted as IMH_LE) and the well known LSH and ITQ are taken into comparison. We denote AGH with one and two layers by AGH_1 and AGH_2, respectively. For AGH, IMH and our method, we use k-means to generate 1,000 cluster centers for anchor or subset points.

The comparison is conducted on the ImageNet dataset, where we form the retrieval and training database by the 100 largest classes with total 128K images from the provided training set, and 50,000 images from the validation set as the query set. The retrieval results of these unsupervised methods are reported in Table VI with code lengths from 32 to 128 bits. Consistent with the supervised experiments, the proposed method outperforms all other methods in both mAP and precision. The advantage of our method is further illustrated by the detailed precision-recall curves and precision curves on top 2,000 retrieved images, as shown in Figure 5.

VI. Conclusions and Discussion

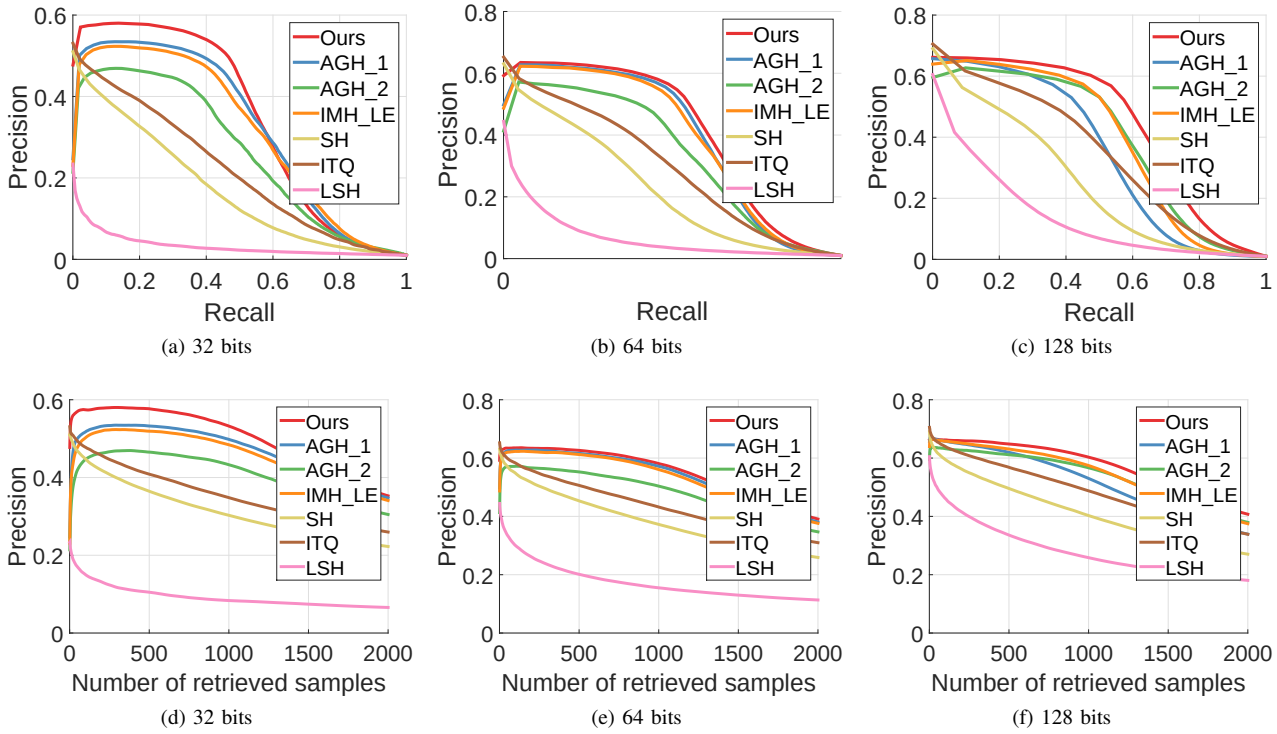
This paper investigated discrete optimization in the general binary code learning problem. To tackle the difficult optimization problem over binary variables, we proposed an effective discrete optimization algorithm, dubbed Discrete Proximal Linearized Minimization (DPLM). Profiting from the analytical solution at each iteration, DPLM led very fast optimization. Compared with existing discrete methods, the proposed method supported a large family of empirical loss functions and constraints, which was instantiated by the supervised ℓ_2 loss and unsupervised graph hashing loss. Several large benchmark datasets were used for evaluation and the results clearly demonstrated the superiority of both our supervised and unsupervised approaches over many other state-of-the-art methods, in terms of both retrieval precision and classification accuracy.

Deep learning based hashing. Deep learning (DL) has become one of the most effective feature learning approach for vision applications. For image hashing, DL also show its promising performance for the image retrieval task ([6], [15], [44]). We note that, however, in the test phase DL

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TIP.2016.2612883, IEEE Transactions on Image Processing

TABLE VI: Results in term of mAP and mean precision of the top 500 retrieved neighbors (precision@500) of the compared unsupervised methods on the **ImageNet** database with 32 to 128 bits.

Method	mAP				Precision@500			
	32 bits	64 bits	96 bits	128 bits	32 bits	64 bits	96 bits	128 bits
Ours	**0.4985**	**0.5445**	**0.5720**	**0.5784**	**0.5854**	**0.6269**	**0.6450**	**0.6485**
IMH	0.4469	0.5172	0.5261	0.5163	0.5191	0.6121	0.6256	0.6311
AGH-1	0.4587	0.5243	0.5176	0.4537	0.5325	0.6159	0.6346	0.6197
AGH-2	0.3927	0.4645	0.5011	0.5222	0.4653	0.5525	0.5919	0.6114
SH	0.2418	0.3066	0.3243	0.3310	0.3647	0.4532	0.4813	0.4956
ITQ	0.3001	0.3839	0.4244	0.4412	0.4086	0.5063	0.5490	0.5683
LSH	0.0485	0.1010	0.1462	0.1922	0.1050	0.2013	0.2731	0.3359



(a) 32 bits (b) 64 bits (c) 128 bits

(d) 32 bits (e) 64 bits (f) 128 bits

Fig. 5: (Top) Precision-Recall and (Bottom) Precision curves with top 2000 retrieved images of the compared methods on **ImageNet**.

based hashing methods need to forward an image through a deep neural network (usually with many layers of projections), which costs much more time than non-DL hashing algorithms including ours (with only one projection). Therefore, with the same input (*e.g.*, raw intensity or GIST feature), the proposed approach can provide more efficient binary code encoding. Another advantage of this work is that, we derive an efficient algorithm for the general binary optimization problem, which effectively handles discrete constraints. It has been shown that directly optimizing with the discrete constraints can produce higher quality codes than the ones with continuous relaxations. In contrast, current DL based hashing algorithms usually resort to a continuous relaxation (*e.g.*, by the sigmoid function).

Potential applications To remedy the aforementioned limitation of current DL based hashing methods, a reasonable solution for high-quality binary code learning will be incorporating the proposed DPLM binary optimization technique into the deep hash function learning process. This will be a challenging yet valuable research direction deserving further studies.

The proposed DPLM method is developed for general bianry optimization, threrefore another potential application of DPLM will be its deployment with different hashing sennarios. For instance, DLPM could be applied to boost the performance of current hashing algorithms with pairwise supervised information (*e.g.*, [21], [7]), multi-model hashing (*e.g.*, [30], [31]), where discrete optimization is supposed to produced higher quality hashes.

In addition to hashing, DPLM is also potentialy applied to other binary optimization problems, such as inner product binarizing [27] and collaborative filtering [45], [40].

REFERENCES

[1] H. Attouch, J. Bolte, and B. F. Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2013.

[2] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, 146(1-2):459–494, 2014.

[3] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval*, 2009.

[4] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. ACM Symposium Comput. geometry*, 2004.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 248–255, 2009.

[6] V. Erin Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou. Deep hashing for compact binary codes learning. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2475–2483, 2015.

[7] T. Ge, K. He, and J. Sun. Graph cuts for supervised binary coding. In *Proc. Eur. Conf. Comp. Vis.*, pages 250–264, 2014.

[8] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proc. Int. Conf. Very Large Databases*, pages 518–529, 1999.

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

[10] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *Proc. Eur. Conf. Comp. Vis.*, pages 392–407. Springer, 2014.

[11] W. Kong and W.-J. Li. Isotropic hashing. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1646–1654, 2012.

[12] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Proc. Adv. Neural Inf. Process. Syst.*, 2009.

[13] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *Proc. IEEE Int. Conf. Comp. Vis.*, 2009.

[14] B. Kulis, P. Jain, and K. Grauman. Fast similarity search for learned metrics. *IEEE Trans. Pattern Anal. Mach. Intell.*, 31(12):2143–2157, 2009.

[15] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3270–3278, 2015.

[16] X. Li, G. Lin, C. Shen, A. van den Hengel, and A. Dick. Learning hash functions using column generation. In *Proc. Int. Conf. Mach. Learn.*, 2013.

[17] G. Lin, C. Shen, Q. Shi, A. van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1971–1978, 2014.

[18] L. Liu, M. Yu, and L. Shao. Multiview alignment hashing for efficient image search. *IEEE Trans. Image Proc.*, 24(3):956–966, 2015.

[19] L. Liu, M. Yu, and L. Shao. Projection bank: From high-dimensional data to medium-length binary codes. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 2821–2829, 2015.

[20] W. Liu, C. Mu, S. Kumar, and S.-F. Chang. Discrete graph hashing. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 3419–3427, 2014.

[21] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 2074–2081, 2012.

[22] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *Proc. Int. Conf. Mach. Learn.*, pages 1–8, 2011.

[23] X. Liu, C. Deng, B. Lang, D. Tao, and X. Li. Query-adaptive reciprocal hash tables for nearest neighbor search. *IEEE Trans. Image Proc.*, 25(2):907–919, 2015.

[24] J. Lu, V. E. Liong, and J. Zhou. Cost-sensitive local binary feature learning for facial age estimation. *IEEE Trans. Image Proc.*, 24(12):5356–5368, 2015.

[25] M. Norouzi and D. M. Blei. Minimal loss hashing for compact binary codes. In *Proc. Int. Conf. Mach. Learn.*, 2011.

[26] M. Raginsky and S. Lazebnik. Locality-sensitive binary codes from shift-invariant kernels. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1509–1517, 2009.

[27] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. Tao Shen. Learning binary codes for maximum inner product search. In *Proc. IEEE Int. Conf. Comp. Vis.*, pages 4148–4156, 2015.

[28] F. Shen, C. Shen, W. Liu, and H. T. Shen. Supervised discrete hashing. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 37–45, 2015.

[29] F. Shen, C. Shen, Q. Shi, A. van den Hengel, Z. Tang, and H. T. Shen. Hashing on nonlinear manifolds. *IEEE Trans. Image Proc.*, 24(6):1839–1851, 2015.

[30] J. Song, Y. Yang, Z. Huang, H. Shen, and J. Luo. Effective multiple feature hashing for large-scale near-duplicate video retrieval. *IEEE Trans. Multimedia*, 15:1997–2008, 2013.

[31] J. Song, Y. Yang, Y. Yang, Z. Huang, and H. T. Shen. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proc. ACM Conf. Management Of Data*, pages 785–796, 2013.

[32] J. Tang, Z. Li, M. Wang, and R. Zhao. Neighborhood discriminant hashing for large-scale image retrieval. *IEEE Trans. Image Proc.*, 24(9):2827–2840, 2015.

[33] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012.

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

[35] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *Proc. Eur. Conf. Comp. Vis.*, pages 340–353, 2012.

[36] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 1753–1760, 2008.

[37] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 3485–3492, 2010.

[38] Y. Yang, F. Shen, H. T. Shen, H. Li, and X. Li. Robust discrete spectral hashing for large-scale image semantic indexing. *IEEE Trans. Big Data*, 1(4):162–171, 2015.

[39] F. Yu, S. Kumar, Y. Gong, and S.-f. Chang. Circulant binary embedding. In *Proc. Int. Conf. Mach. Learn.*, pages 946–954, 2014.

[40] H. Zhang, F. Shen, W. Liu, X. He, H. Luan, and T.-S. Chua. Discrete collaborative filtering. In *Proc. ACM Conf. Inf. Ret.*, 2016.

[41] L. Zhang, H. Lu, D. Du, and L. Liu. Sparse hashing tracking. *IEEE Trans. Image Proc.*, 25(2):840–849, 2016.

[42] P. Zhang, W. Zhang, W.-J. Li, and M. Guo. Supervised hashing with latent factor models. In *Proc. ACM Conf. Inf. Ret.*, pages 173–182, 2014.

[43] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. *IEEE Trans. Image Proc.*, 24(12):4766–4779, 2015.

[44] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pages 1556–1564, 2015.

[45] K. Zhou and H. Zha. Learning binary codes for collaborative filtering. In *Proc. ACM Conf. Knowl. Disc. Data Min.*, pages 498–506, 2012.