Hadamard Coding for Supervised Discrete Hashing

Gou Koutaki, *Member, IEEE*, Keiichiro Shirai, *Member, IEEE*, and Mitsuru Ambai, *Member, IEEE*

***Abstract*— In this paper, we propose a learning-based supervised discrete hashing (SDH) method. Binary hashing is widely used for large-scale image retrieval as well as video and document searches, because the compact binary code representation is essential for data storage and reasonable for query searches using bit operations. The recently proposed SDH method efficiently solves mixed-integer programming problems by alternating optimization and the *discrete cyclic coordinate descent* (DCC) method. Based on some preliminary experiments, we show that the SDH method can be simplified without performance degradation. We analyze the simplified model and provide a mathematically exact solution thereof; we reveal that the exact binary code is provided by a "Hadamard matrix." Therefore, we named our method *Hadamard coded-SDH* (HC-SDH). In contrast to the SDH, our model does not require an alternating optimization algorithm and does not depend on initial values. The HC-SDH is also easier to implement than the *iterative quantization*. Experimental results involving a large-scale database show that the Hadamard coding outperforms the conventional SDH in terms of precision, recall, and computational time. On the large data sets SUN-397 and ImageNet, the HC-SDH provides a superior mean average of precision (mAP) and top-accuracy compared with the conventional SDH methods with the same code length and FastHash. The training time of the HC-SDH is 170 times faster than the conventional SDH and the testing time including the encoding time is seven times faster than the FastHash which encodes using a binary-tree.**

***Index Terms*— Supervised discrete hashing, Hadamard matrix, binary orthogonal.**

I. Introduction

BINARY hashing is an important technique for pattern recognition, computer vision, machine learning, and large-scale image/video/document retrieval [1]–[7]. Binary hashing enables multi-dimensional feature vectors with integers or floating-point elements to be transformed into short binary codes.

This binary code representation is an important technique since large-scale databases occupy large amounts of storage. Furthermore, it is easy to compare a query in binary code with a binary code in a database because the Hamming distance between them can be computed efficiently by using bitwise operations that are part of the instruction set of any modern CPU [8], [9].

Many binary hashing methods have been proposed in the past two decades. *Locality-sensitive hashing* (LSH) [1] is one of the most popular methods. LSH generates binary codes by using a random projection matrix and thresholding using the sign of the projected data. *Iterative quantization* (ITQ) [2] is another state-of-the-art binary hashing method. ITQ optimizes a projection matrix of the hash function by iterating projection and thresholding procedures according to the given training samples. The binary code of a query is compared with the binary codes in a database, and efficient algorithms for searching for the nearest neighbor code have been developed [10]. Moreover, classifiers such as support vector machine and its improved method [11] are used for the classification.

Binary hashing can be roughly classified into two types: unsupervised hashing [3], [6], [12]–[15] and supervised hashing. Supervised hashing uses label information if it exists. In general, supervised hashing is more accurate than unsupervised hashing; thus, in this study, we target supervised hashing. In addition, some unsupervised methods such as LSH and ITQ can be converted into supervised methods by imposing label information on their feature vectors. For example, canonical correlation analysis (CCA) [16] can transform feature vectors to maximize inter-class variation and minimize intra-class variation according to the label information. Hereafter, we refer to these processes as CCA-LSH and CCA-ITQ, respectively.

Rather than imposing label information on feature vectors, such as in CCA, imposing it directly on hash functions has been proposed. In this case, a cost function is defined by both the label information and feature vectors. *Kernel-based supervised hashing* (KSH) [17] uses spectral relaxation to optimize the cost function through a sign function. The feature vectors are transformed by kernels during preprocessing. KSH has also been improved to *kernel-based supervised discrete hashing* (KSDH) [18]. It relaxes the discrete hashing problem through linear relaxation. *Supervised discriminative hashing* [5] decomposes training samples into inter- and intra-samples. *Column sampling-based discrete supervised hashing* (COSDISH) [19] uses column sampling based on semantic similarity, and decomposes the problem into a sub-problem to make it easier to solve. In these methods, binary code $\mathbf{b}$ is computed as $\mathbf{b} = \mathrm{sign}(\mathbf{P}^\top \mathbf{x})$ by using a sign function and linear matrix operation on the input feature vector $\mathbf{x}$. A main problem of these approaches is to obtain the matrix $\mathbf{P}$. On the other hand, the use of other algorithms to compute the binary code has also been proposed. For example, *fast*

Manuscript received October 26, 2017; revised April 1, 2018 and May 26, 2018; accepted June 18, 2018. Date of publication July 12, 2018; date of current version August 14, 2018. This work was supported by JST PRESTO. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Xiaochun Cao. *(Corresponding author: Gou Koutaki.)*

G. Koutaki is with the Department of Electrical and Computer Engineering, Kumamoto University, Kumamoto 860-8555, Japan (e-mail: koutaki@cs.kumamoto-u.ac.jp).

K. Shirai is with the Faculty of Engineering, Shinshu University, Nagano 380-8553, Japan (e-mail: keiichi@shinshu-u.ac.jp).

M. Ambai is with the Denso IT Laboratory, Inc., Tokyo 150-0002, Japan (e-mail: manbai@d-itlab.co.jp).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2018.2855427

supervised hashing (FastHash) [20] generates a binary code using a boosted decision tree.

The optimization of binary codes leads to a mixed-integer programming problem involving integer and non-integer variables, which is an NP-hard problem in general [7]. Therefore, many methods either discard the discrete constraints or transform the problem into a relaxed problem, *i.e.*, a linear programming problem [21]. This relaxation significantly simplifies the problem, but is known to affect classification performance [7].

Recent research has introduced a type of *supervised discrete hashing* (SDH) [7] that directly learns binary codes without relaxation. SDH is a state-of-the-art method because of its ease of implementation, reasonable computation time for learning, and enhanced performance over other state-of-the-art supervised hashing methods. SDH solves discrete problems by using a *discrete cyclic coordinate descent* (DCC) method, which is an approximate solver of 0-1 quadratic integer programming problems. In the latest work of [22], rather than using the integer programming solver, a simple sign operation to optimize the binary code generates good binary codes of SDH and it is efficient to train samples while maintaining the performance. This algorithm is known as *fast SDH* (FSDH).

Moreover, there are some improvements in supervised hashing and its cost function. *Supervised discrete hashing with relaxation* (SDHR) [23] improved label-loss by adding an offset vector. In spite of the simple modification, the performance can be improved. In [24], $\ell_{p,q}$ mixed norm is introduced as the cost function for robust similarity searches. *Distributed adaptive binary quantization* has been proposed [25] and it optimizes the binary codes associated with the distribution of training samples. Some methods use the structure of samples [26], [27]. A sparse encoding and sparse anchor representation are also useful for general hashing frameworks [28], [29]. Recently, neural network based hashing methods have been developed [30], [31] and have high accuracy.

A. Contributions and Advantages of Our Method

In this study, we first analyze the SDH method and point out that it can be simplified without performance degradation based on some preliminary experiments. We analyze the approximated model and provide a mathematically exact solution thereof. Because the exact solution can be provided by the Hadamard matrix as described later, we name our method *Hadamard coded-supervised discrete hashing* (HC-SDH). The model simplification is validated through experiments involving several large-scale datasets.

The advantages of the proposed method are as follows:

- Unlike SDH or FSDH, HC-SDH does not require alternating optimization or hyper-parameters, and is does not depend on the initial value.
- It is easier to implement than ITQ and is efficient in terms of computational time. In MATLAB, the implementation of HC-SDH only requires three lines.
- High bit scalability: the learning time and performance of HC-SDH do not depend on the code length.
- HC-SDH is superior in terms of precision and recall than other state-of-the-art supervised hashing methods.

B. Related Work

1) PQ Type Hashing: Binary hashing methods such as ITQ and SDH are a kind of quantization hashing. Another type of quantization hashing, product quantization (PQ), is well known [32]–[36]. PQ-type hashing uses vector quantization such as k-means and an input feature vector is represented by the code word of training samples. In binary hashing, a query vector is encoded into a binary code and a symmetric distance such as the Hamming distance is used. On the other hand, in PQ-type hashing, the query vector is not encoded into a binary code and an asymmetric distance is used for searching. Although discussions of PQ-type hashing vs. binary hashing are continuing today, in this study, we focus on binary hashing.

2) Matrix Factorization: As described subsequently, the SDH method poses a matrix factorization problem: $\mathbf{F} = \mathbf{W}^\top \mathbf{B}$. The popular form of this problem is singular value decomposition (SVD) , and when $\mathbf{W}$ and $\mathbf{B}$ are unconstrained, the Householder method is used for computation. When $\mathbf{W} \geq 0$, non-negative matrix factorization (NMF) is used [37]. In the case of the SDH method, $\mathbf{B}$ is constrained to $\{-1, 1\}$ and $\mathbf{W}$ is unconstrained. In a similar problem setting, Slawski *et al.* proposed matrix factorization with binary components [38].

3) Other Applications: Other applications of factorization with binary components exist outside of hashing for pattern recognition and image retrieval. In this regard, [38] reported an application to DNA analysis for cancer research in which $\mathbf{B}$ is constrained to $\{0, 1\}$, and indicates unmethylated/methylated DNA sequences. The factorization was optimized by using CPLEX. Furthermore, a similar model has been proposed for use in display electronics. The work of [39] proposed binary continuous decomposition for multi-view displays. Their model decomposes multiple images $\mathbf{F}$ into binary images $\mathbf{B}$ and a weight matrix $\mathbf{W}$. An image projector projects binary 0-1 patterns through digital mirror devices (DMDs), and the weight matrix corresponds to the transmittance of the LCD shutter. The decomposition was optimized by using particle swam optimization (PSO) and a branch-and-bound method.

The remainder of this paper is organized as follows. Section II contains an overview of the SDH method and Sec. III discusses the technical problems this model presents. Section IV elaborates on the proposed approximated SDH method and analyzes the model, after which the exact solution of the model is provided. Section V describes the experiments in detail, provides the results, and discusses the findings. Finally, the paper is concluded in Sec. VI.

II. Supervised Discrete Hashing (SDH) Model

In this section, we introduce the *supervised discrete hashing* (SDH) model. Let $\mathbf{x}_i \in \mathbb{R}^M$ be a feature vector as an M dimensional column vector, and let us introduce a set of N ($\geq M$) training samples $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_N] \in \mathbb{R}^{M \times N}$. Then, consider binary label information $\mathbf{y}_i \in \{0, 1\}^C$ corresponding

to $\mathbf{x}_i$, where C is the number of classification categories. Setting the k-th element to 1, $[\mathbf{y}_i]_k = 1$, and the other elements to 0 indicates that the i-th vector belongs to class k. By concatenating N $\mathbf{y}_i$ vectors horizontally, a label matrix $\mathbf{Y} := \left[\mathbf{y}_1, \ldots, \mathbf{y}_N\right] \in \{0, 1\}^{C \times N}$ is constructed.

A. Binary Code Assignment to Each Sample

To each sample $\mathbf{x}_i$, an L-bit binary code $\mathbf{b}_i \in \{-1, 1\}^L$ is assigned. By concatenating N $\mathbf{b}_i$ vectors horizontally, a binary matrix $\mathbf{B} := [\mathbf{b}_1, \ldots, \mathbf{b}_N] \in \{-1, 1\}^{L \times N}$ is constructed. The binary code $\mathbf{b}_i$ is computed as

$$\mathbf{b}_i = \operatorname{sign}\left(\mathbf{P}^\top \mathbf{x}_i\right), \tag{1}$$

where $\mathbf{P} \in \mathbb{R}^{M \times L}$ (hence $\mathbf{P}^\top \in \mathbb{R}^{L \times M}$) is a linear transformation matrix and $\operatorname{sign}(\cdot)$ is the sign function. The major aim of SDH is to determine the matrix $\mathbf{P}$ from training samples $\mathbf{X}$. In practice, feature vectors $\{\mathbf{x}_i\}$ are transformed by preprocessing. Thus, we denote the original feature vectors $\mathbf{x}_i^{\text{ori}}$ and the transformed feature vectors $\mathbf{x}_i$.

B. Preprocessing: Kernel Transformation

The original feature vectors of training samples $\mathbf{x}_i^{\text{ori}}$ $(i = 1, \ldots, N)$ are converted into the feature vectors $\mathbf{x}_i \in \mathbb{R}^M$ using the following kernel transformation Φ:

$$\begin{aligned}\mathbf{x}_i &\leftarrow \Phi(\mathbf{x}_i^{\text{ori}}) \\ &= \left[\exp\left(-\frac{\|\mathbf{x}_i^{\text{ori}} - \mathbf{a}_1\|^2}{\sigma}\right), \ldots, \exp\left(-\frac{\|\mathbf{x}_i^{\text{ori}} - \mathbf{a}_m\|^2}{\sigma}\right)\right]^\top,\end{aligned} \tag{2}$$

where $\mathbf{a}_m$ is an anchor vector obtained by randomly sampling the original feature vectors, $\mathbf{a}_m = \mathbf{x}_{\text{rand}}^{\text{ori}}$. Then, the transformed feature vectors are bundled into the matrix form $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_N]$.

C. Classification Model

Following binary coding by (1), we assume that the binary code that classifies the class is effective, and formulate the following simple linear classification model:

$$\widehat{\mathbf{y}}_i = \mathbf{W}^\top \mathbf{b}_i, \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{L \times C}$ is a weight matrix and $\widehat{\mathbf{y}}_i$ is an estimated label vector. As mentioned above, its maximum index, $\arg\min_k [\widehat{\mathbf{y}}_i]_k$, indicates the assigned class of $\mathbf{x}_i$.

D. Optimization of SDH

The SDH problem is defined as the following minimization problem:

$$\min_{\mathbf{B},\mathbf{W},\mathbf{P}} \|\mathbf{Y} - \mathbf{W}^\top \mathbf{B}\|^2 + \lambda \|\mathbf{W}\|^2 + \nu \|\mathbf{B} - \mathbf{P}^\top \mathbf{X}\|^2, \tag{4}$$

where $\|\cdot\|$ is the Frobenius norm, and $\lambda \geq 0$ and $\nu \geq 0$ are balance parameters. The first term includes the classification model explained in Sec. II-C. The second term is a regularizer for $\mathbf{W}$ to avoid overfitting. The third term indicates the fitting errors due to binary coding.

In this optimization, it is sufficient to compute $\mathbf{P}$, *i.e.*, if $\mathbf{P}$ is obtained, $\mathbf{B}$ can be obtained by (1), and $\mathbf{W}$ can be obtained from the following simple least-squares equation:

$$\mathbf{W} = \left(\mathbf{B}\mathbf{B}^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{B}\mathbf{Y}^\top. \tag{5}$$

However, because of the difficulty presented by optimization, the optimization problem of (4) is usually divided into three sub-problems comprising the optimization of $\mathbf{B}, \mathbf{W}$, and $\mathbf{P}$, respectively. Thus, the following alternating optimization is performed:

(i) **Initialization**: $\mathbf{B}$ is initialized, usually randomly.

(ii) **F-Step**: $\mathbf{P}$ is computed by the following simple least-squares method:

$$\mathbf{P} = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1} \mathbf{X}\mathbf{B}^\top. \tag{6}$$

(iii) **W-Step**: $\mathbf{W}$ is computed by (5).

(iv) **B-Step**: After fixing $\mathbf{P}$ and $\mathbf{W}$, Eq. (4) becomes

$$\begin{aligned}&\min_{\mathbf{B}} \ \|\mathbf{Y}\|^2 - 2\operatorname{Tr}(\mathbf{W}^\top \mathbf{Y}^\top \mathbf{B}) + \operatorname{Tr}(\mathbf{B}^\top \mathbf{W}\mathbf{W}^\top \mathbf{B}) \\ &\qquad + \nu\left(\|\mathbf{B}\|^2 - 2\operatorname{Tr}(\mathbf{P}^\top \mathbf{X}\mathbf{B}) + \|\mathbf{P}^\top \mathbf{X}\|^2\right) \\ &\Rightarrow \min_{\mathbf{B}} \ \operatorname{Tr}(\mathbf{B}^\top \mathbf{Q}\mathbf{B} + \mathbf{F}^\top \mathbf{B}),\end{aligned} \tag{7}$$

where

$$\begin{aligned}\mathbf{Q} &:= \mathbf{W}\mathbf{W}^\top \in \mathbb{R}^{L \times L}, \\ \mathbf{F} &:= -2\left(\mathbf{W}\mathbf{Y} + \nu \mathbf{P}^\top \mathbf{X}\right) \in \mathbb{R}^{L \times N}.\end{aligned} \tag{8}$$

Note that $\operatorname{Tr}(\mathbf{B}^\top \mathbf{B}) = LN$. The trace can be rewritten as

$$\min_{\{\mathbf{b}_i\}} \ \sum_{i=1}^{N} \mathbf{b}_i^\top \mathbf{Q}\mathbf{b}_i + \mathbf{f}_i^\top \mathbf{b}_i, \tag{9}$$

where $\mathbf{f}_i \in \mathbb{R}^L$ is the i-th column vector of $\mathbf{F}$. The $\{\mathbf{b}_i\}$ are independent of one another. Therefore, it reduces to the following 0-1 integer quadratic programming problem for each i-th sample:

$$\forall_i \ \min_{\mathbf{b}_i \in \{-1,1\}^L} \ \mathbf{b}_i^\top \mathbf{Q}\mathbf{b}_i + \mathbf{f}_i^\top \mathbf{b}_i. \tag{10}$$

Here, the FSDH method simply computes the binary code as

$$\mathbf{B} = \operatorname{sign}(\mathbf{W}\mathbf{Y} + \nu \mathbf{P}^\top \mathbf{X}). \tag{11}$$

(v) Iterate steps (ii)$\sim$(iv) until convergence is achieved.

III. Discussion of the SDH Method

A. 0-1 Integer Quadratic Programming Problem

1) DCC Method: SDH solves (10) by using a *discrete cyclic coordinate descent* (DCC) method. This method optimizes a one-bit element of $\mathbf{b}_i$ while fixing the other $L - 1$ bits; the l-th bit b_l is optimized as

$$b_l = -\operatorname{sign}\left(2 \sum_{i \neq l} Q_{i,l} b_i + f_l\right). \tag{12}$$

Then, all bits $l = 1, \ldots, L$ are optimized, and this procedure is repeated several times. In addition, the DCC method is prone to converge to a local minimum because of its greediness. To improve this method, Shen *et al.* proposed using a proximal operation of convex optimization [40].

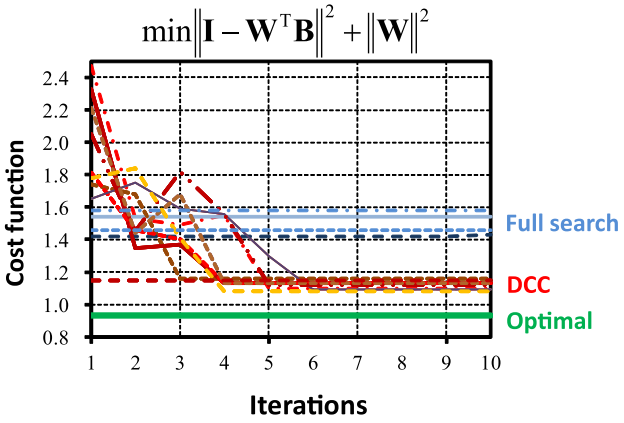

Fig. 1. Convergence of optimizations with several initial conditions. Even though the problem is simple, conventional solvers with alternating optimization (DCC and full search) cannot reach the optimal solution (green line) and converge to local minima.

2) Branch-and-Bound Method: In the case of a large number of bits $L \geq 32$, solving (10) exactly is difficult because this problem is NP-hard. However, a few efficient methods capable of solving the 0-1 integer quadratic programming problem, exist. In [39], a branch-and-bound method is used to solve the problem. Their approach is to expand $\mathbf{b}$ into a binary tree of depth L, and the problem of (10) is divided into a sub-problem by splitting $\mathbf{b} = [\mathbf{b}_1^\top, \mathbf{b}_2^\top]^\top$. At each node, the lower bound is computed and compared with the given best solution; child nodes can be excluded from the search.

The computation of the lower bound depends on the structure of $\mathbf{Q}$, $\mathbf{q}$, and $\mathbf{b}$. In general, a standard method for computing the lower bound is the linear relaxation method, $\mathbf{b} \in \{-1, 1\}^L \Rightarrow \mathbf{b} \in [-1, 1]^L$. In this case, a rough lower bound of the quadratic term in (10) can be provided by the minimum eigenvalues of $\mathbf{Q}$. However, linear relaxation is pointless in the SDH method because $L > C$ in general; hence, the matrix $\mathbf{Q} = \mathbf{W}\mathbf{W}^\top$ is rank deficient and, as a result, the minimum eigenvalue of $\mathbf{Q}$ becomes zero.

Even if we could obtain an efficient algorithm, such as branch-and-bound and a good lower bound, the application of binary hashing would still present computational difficulties because code lengths of $L = 64, 128$, or 256 bits, which are frequently encountered, would remain too long to optimize.

B. Alternating Optimization and Initial Value Dependence

Even if we could perform the binary optimization in (10), the resulting binary codes $\mathbf{B}$ would not always be optimal because they depend on other fixed variables $\mathbf{W}$ and $\mathbf{P}$. In addition, alternating optimization is prone to cause a serious problem: the solution depends on the initial values, and may converge to a local minimum during iteration, even if each step of **F-Step**, **W-Step**, and **B-Step** provided the optimal solution.

Figure 1 shows an example of the optimization result for a simple version of the SDH method in (4) with a small number of bits $(L, C, N) = (16, 10, 10)$. In this case, an exact solution is known and the minimum value is 0.94 (green line in Fig. 1). The DCC (red lines) provides results for 10 randomized initial conditions. The full search (blue lines) provides the results of an exact full search in **B-Step**, where $2^{16} = 65,536$ nodes are searched.

Despite the small size of the problem, the cost function of conventional alternating solvers (DCC and full search) cannot find the exact value, and depend on the initial values. Interestingly, a full search immediately converges to a local minimum, and the results are less accurate than those of DCC.

IV. Proposed SDH Method

We introduce a new hashing model by approximating the SDH method. The new model utilizes the following assumptions:

A1: The number of bits L of the binary code is a power of 2: $L = 2^l$.

A2: The number of bits is greater than the number of classes: $L \geq C$.

Single-labeling

A3: problem.

A4: $\|\mathbf{W}\mathbf{Y}\|^2 \gg \nu \|\mathbf{P}^\top \mathbf{X}\|^2$ in (8).

Note that assumptions **A1**∼**A3** also become the limitations of the proposed model. In **A4**, SDH recommends that the parameter ν is set to a very small value, such as $\nu = 10^{-5}$ [7]. In practice, $\|\mathbf{W}\mathbf{Y}\|^2 \fallingdotseq 31.53$ and $\nu \|\mathbf{P}^\top \mathbf{X}\|^2 \fallingdotseq 0.013$ in the CIFAR-10 dataset. Furthermore, when $\nu = 0$, almost the same results can be obtained for all datasets as shown in the experimental results in Sec. V.

Using the approximation, we solve the following problem for each N-sample $\mathbf{b}_i$ in **B-Step**:

$$\forall_i \min_{\mathbf{b}_i \in \{-1,1\}^L} \mathbf{b}_i^\top \mathbf{Q} \mathbf{b}_i + \mathbf{f}_i^\top \mathbf{b}_i, \tag{13}$$

where $\mathbf{Q} := \mathbf{W}\mathbf{W}^\top$ is a constant matrix and $\mathbf{f}_i$ that is the i-th column vector of $\mathbf{F} := -2\mathbf{W}\mathbf{Y}$ depends on label $\mathbf{y}_i$. By using the single-label assumption in **A3**, the number of kinds of $\mathbf{y}_i$ is limited to C:

$$\mathbf{y}_1 = [1, 0, \ldots, 0]^\top, \ldots, \mathbf{y}_C = [0, 0, \ldots, 1]^\top. \tag{14}$$

Thus, it is sufficient to solve only C of the N integer quadratic programming problems of (13). In general, the number of samples N is larger than the number of classes: $N \gg C$, *e.g.*, $N = 59,000$ and $C = 10$. Thereby, the computational cost of **B-Step** becomes 5,900 times lower. In other words, the approximation proposes the following:

Proposition 1: The approximation by $\nu = 0$ defines the SDH method to assign a binary code to each class.

After obtaining the binary codes of each class $\mathbf{B}' = [\mathbf{b}'_1, \ldots, \mathbf{b}'_C] \in \{-1, 1\}^{L \times C}$, the binary codes of all samples $\mathbf{B}$ can be constructed by lining up $\mathbf{b}'_i$ as

$$\mathbf{B} := \left[\mathbf{b}'_{y_1}, \ldots, \mathbf{b}'_{y_N}\right]. \tag{15}$$

After constructing $\mathbf{B}$, the projection matrix $\mathbf{P}$ can be obtained by (6).

A. Analytical Solutions of the Approximated SDH Method

From Proposition 1, we found that it is sufficient to determine the binary code for each class. Furthermore, we can choose the optimal binary codes under the approximation model as follows:

Lemma 1: If $f(x_i)$ is convex, the solution of

$$\min_{\{x_i\}} \sum_{i}^{N} f(x_i) \quad s.t. \ \sum_{i}^{N} x_i = L \tag{16}$$

is given by the mean value $x_i = L/N$ $(i = 1, \ldots, N)$.

Proof: See Appendix VI-A. □

Theorem 1: An analytical solution of the approximated SDH $\mathbf{B}'$ is obtained as a Hadamard matrix.

Proof: Using the approximation and label representations in (14), the SDH method in (4) becomes

$$\min_{\mathbf{B}',\mathbf{W}} \|\mathbf{I} - \mathbf{W}^\top \mathbf{B}'\|^2 + \lambda \|\mathbf{W}\|^2, \tag{17}$$

where $\mathbf{I} \in \mathbb{R}^{C\times C}$ is an identity matrix. Using the solution of (17), *i.e.*, $\mathbf{W} = (\mathbf{B}'\mathbf{B}'^\top + \lambda \mathbf{I})^{-1}\mathbf{B}'$, and the eigendecomposition of $\mathbf{B}'^\top \mathbf{B}' = \mathbf{U}^\top \mathbf{D}\mathbf{U}$, we denote the eigenvalues as σ_i and denotes a diagonal matrix $\mathrm{diag}(\mathbf{D}) = \{\sigma_i\}_{i=1}^{C}$ and then obtain $\sum_{i=1}^{C} \sigma_i = \mathrm{Tr}(\mathbf{D}) = \mathrm{Tr}(\mathbf{B}'^\top \mathbf{B}') = LC$ as the trace of diagonal values. Then, (17) can be represented simply as

$$\min_{\mathbf{B}'} \ \sum_{i=1}^{C} \frac{\lambda}{\sigma_i + \lambda} \quad s.t. \ \sum_{i=1}^{C} \sigma_i = LC. \tag{18}$$

By Lemma 1, $\sigma_i = L$ $(i = 1, \ldots, C)$. This implies that $\mathbf{B}'$ is an orthogonal matrix with binary elements $\{-1, 1\}$; in other words, $\mathbf{B}' \in \{-1, 1\}^{L\times C}$ can be obtained as a submatrix of the Hadamard matrix $\mathbf{H} \in \{-1, 1\}^{L\times L}$. We named our method *Hadamard coded-supervised discrete hashing* (HC-SDH). □

Corollary 1: The following characteristics can be obtained easily:

- *$\mathbf{B}'$ is independent of regularization parameter λ (i.e., it is λ-invariant).*
- *The optimal weight matrix $\mathbf{W}$ of HC-SDH is given by scaling the binary matrix $\mathbf{B}'$ as $\mathbf{W} = \frac{1}{L+\lambda}\mathbf{B}'$.*
- *The minimum value of* (17) *is given by $\frac{\lambda C}{L+\lambda}$.*

In short, this means it becomes possible to eliminate the **W-Step**, the alternating procedure, and the initial value dependence. Moreover, an exact solution of the HC-SDH method can be obtained independently of the hyper-parameters λ and ν.

B. Implementation of HC-SDH

Algorithm 1 and Figure 2, respectively, show the algorithm of HC-SDH and sample MATLAB code, which is simple and easy to implement. Figure 3 shows an example of $\mathbf{B}'$ and $\mathbf{B}$. A Hadamard matrix of size $2^k \times 2^k$ can be constructed recursively by Sylvester's method [41] as

$$\mathbf{H}_2 := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix},$$
$$\mathbf{H}_{2^k} := \begin{bmatrix} \mathbf{H}_{2^{k-1}} & \mathbf{H}_{2^{k-1}} \\ \mathbf{H}_{2^{k-1}} & -\mathbf{H}_{2^{k-1}} \end{bmatrix} \quad (k \geq 2). \tag{19}$$

Algorithm 1 Hadamard Coded-Supervised Discrete Hashing (HC-SDH)

Input: Pre-processed training data $\mathbf{X}$ and labels $\{y_i\}_{i=1}^{N}$: code length L, number of samples N, number of classes C.
Output: Projection matrix $\mathbf{P}$.
1: Compute Hadamard matrix $\mathbf{H} \in \{-1, 1\}^{L\times L}$
2: Let $[\mathbf{b}'_1, \ldots, \mathbf{b}'_C]$ be C columns of $\mathbf{H}$.
3: Construct $\mathbf{B}$ by $\mathbf{b}_i = \mathbf{b}'_{y_i}$.
4: Compute $\mathbf{P}$ from $\mathbf{B}$ and $\mathbf{X}$ by (6).

```
HA = hadamard(L);  %L-bit Hadamard matrix
B  = HA(y,:);      %y:label array
P  = (X*X')\(X*B');%X:feature vectors
```

Fig. 2. Sample MATLAB code for the main part of HC-SDH, which is implemented in only three lines and is easier to implement than the ITQ algorithm.

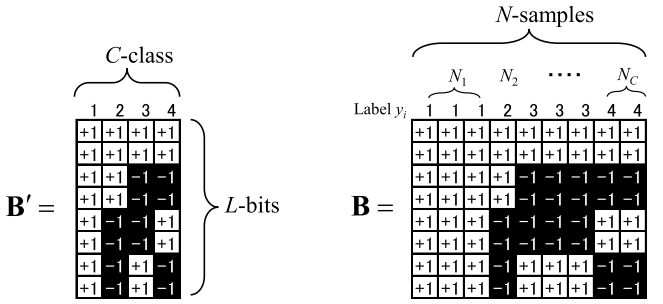

Fig. 3. An example of the construction of $\mathbf{B}'$ and $\mathbf{B}$ with $L = 8$ bits and $C = 4$ classes. After computing $\mathbf{B}'$ from the Hadamard matrix, the binary code $\mathbf{B}$ with N columns is constructed according to label y_i.

Furthermore, Hadamard matrices of orders 12 and 20 were constructed by Hadamard transformation [42]. Fortunately, in applications of binary hashing, because $L = 16$, the numbers of bits that are frequently used are 32, 64, 128, 256, and 512, and Sylvester's method suffices in most cases. Here, the assumption **A1** is not a necessary and sufficient condition because it suffices to obtain the binary orthogonal matrix such as a Hadamard matrix. More precisely, it is known that there exist Hadamard matrices when n is a power of 2; $n = L/12$ or $L/20$. Therefore, for example $L = 24, 48, 96$, and 192 are also available and are sufficient for actual applications.

C. Analysis of Bias Term of HC-SDH

We have already shown that $\mathbf{B}$, which is obtained from the Hadamard matrix, minimizes two terms: $\|\mathbf{Y} - \mathbf{W}^\top \mathbf{B}\|^2 + \lambda \|\mathbf{W}\|^2$. Furthermore, we pay attention to the way in which $\mathbf{B}$ affects the bias term $\|\mathbf{B} - \mathbf{P}^\top \mathbf{X}\|^2$. In this subsection, we continue to analyze its behavior. We suppose that samples are sorted by label y_i. Let $\mathbf{P}^\top = \mathbf{B}\mathbf{X}^\top \left(\mathbf{X}\mathbf{X}^\top\right)^{-1}$ be the bias term:

$$\begin{aligned} \|\mathbf{B} - \mathbf{P}^\top \mathbf{X}\|^2 &= \|\mathbf{B}\,(\mathbf{I} - \mathbf{K})\,\|^2 \\ &= \mathrm{Tr}(\mathbf{B}^\top \mathbf{B}) - \mathrm{Tr}(\mathbf{B}\mathbf{K}\mathbf{B}^\top), \end{aligned} \tag{20}$$

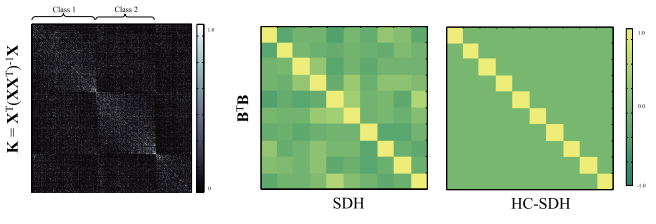

Fig. 4. Visualization of matrix $\mathbf{K}$ (left) and matrix $\mathbf{B}^\top\mathbf{B}$ (right). $\mathbf{B}^\top\mathbf{B}$ of SDH includes a "negative" block in the non-diagonal components, and reduces $\mathrm{Tr}(\mathbf{K}\mathbf{B}^\top\mathbf{B}\mathbf{K})$.

where $\mathbf{K} := \mathbf{X}^\top\left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X} \in \mathbb{R}^{N\times N}$ is a projection matrix. Therefore, to reduce the bias term, $\mathrm{Tr}(\mathbf{B}\mathbf{K}\mathbf{B}^\top)$ should preferably have a large value. Then, using $\mathbf{K} = \mathbf{K}\mathbf{K}$, we can rewrite it as

$$\mathrm{Tr}(\mathbf{B}\mathbf{K}\mathbf{B}^\top) = \mathrm{Tr}(\mathbf{K}\mathbf{B}^\top\mathbf{B}\mathbf{K}), \tag{21}$$

where $\mathbf{B}^\top\mathbf{B}$ is a block-diagonal matrix:

$$\mathbf{B}^\top\mathbf{B} = L\begin{bmatrix}\mathbf{J}_{N_1} & & \mathbf{O}\\ & \ddots & \\ \mathbf{O} & & \mathbf{J}_{N_C}\end{bmatrix}, \tag{22}$$

where $\mathbf{J}_{N_k} \in 1^{N_k\times N_k}$ are matrices with all elements equal to 1, and N_k is the number of samples with label $y_i = k$. Using these values, $\mathrm{Tr}(\mathbf{K}\mathbf{B}^\top\mathbf{B}\mathbf{K})$ in (21) can be expressed as

$$L\sum_{i=1}^{N}\Big[\left(K_{i,1}+\ldots+K_{i,N_1}\right)^2+\left(K_{i,N_1+1}+\ldots+K_{i,N_2}\right)^2 +\ldots+\left(K_{i,N_{C-1}+1}+\ldots+K_{i,N_C}\right)^2\Big], \tag{23}$$

where $\{K_{i,j}\}$ with the same label $y_i = y_j$ are summed. The definition $\mathbf{K} := \mathbf{X}^\top\left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X}$, K_{ij} can be regarded as the normalized correlation of $\mathbf{x}_i$ and $\mathbf{x}_j$. Since samples with the same label must represent a similar feature vector, $\mathrm{Tr}(\mathbf{K}\mathbf{B}^\top\mathbf{B}\mathbf{K})$ is assumed to be a large value.

Figure 4 shows visualizations of matrices $\mathbf{K}$ and $\mathbf{B}^\top\mathbf{B}$ for SDH and HC-SDH. High-correlation areas of $\mathbf{K}$ are partitioned by each class block. $\mathbf{B}^\top\mathbf{B}$ of SDH includes a "negative" block in the non-diagonal components, and reduces $\mathrm{Tr}(\mathbf{K}\mathbf{B}^\top\mathbf{B}\mathbf{K})$. On the other hand, the proposed HC-SDH shows clear blocks; the diagonal blocks take the value L and the non-diagonal blocks 0.

V. Experiments

We tested two experiments named *Experiment I* and *Experiment II*. First, we tested datasets having a small number of classes, $C = 10$, with many methods, and then we tested datasets having over 100 classes with methods chosen according to the results of *Experiment I*.

A. Datasets

We tested the proposed method on four kinds of large-scale image datasets: CIFAR-10 and CIFAR-100 [43], SUN-397 [44], MNIST [45], and ImageNet-2012 [46]. The feature vectors of all datasets were normalized.

Recently, although an evaluation method for supervised hashing has been discussed [47], we use a standard method for the evaluation.

CIFAR-10 and CIFAR-100 include labeled subsets of 60,000 images. In *Experiment I*, we used 512-dimensional GIST features [48] extracted from the images. $N = 59,000$ training samples and 1,000 test samples were used for evaluation. The number of classes was $C = 10$ in CIFAR-10, and included "airplane," "automobile," "bird," and so on. In *Experiment II*, a CIFAR-100 of/with a large number of classes $C = 100$ was used.

SUN-397 is a large-scale image dataset for scene recognition with 397 categories, and consists of 108,754 labeled images. In *Experiment I*, we extracted 10 categories with $C = 10$ and $N = 5,000$ training samples. A total of 500 training samples per class and 1,000 test samples were used. We used 512-dimensional GIST features extracted from the images. For $C = 10$, we named the dataset "SUN-10." In *Experiment II*, we used $C = 397$ classes and $N = 79,400$ training samples.

MNIST includes an image dataset of handwritten digits. The feature vectors we used were given by $28\times 28 = 784$ [pix] of data that were normalized. The number of classes was $C = 10$, *i.e.*, '0' ~ '9' digits. We used $N = 30,000$ training samples and 1,000 test samples for evaluation.

ImageNet is a large-scale image dataset with over 20,000 categories and it includes 14 million images. We extracted 1024-dimensional GIST features from images resized to 64×64 pixels. We sampled 50,000 training samples and 1,000 test samples from 1,000 categories of the original dataset.

B. Comparative Methods and Settings

The proposed method was compared with six state-of-the-art supervised hashing methods: CCA-ITQ, SDH, FSDH [22], COSDISH [19], FastHash [20], and KSDH [18]. Unsupervised or semi-supervised methods were not assessed. All methods were implemented in MATLAB R2016 and tested on an Intel i7-6950X@3.0 GHz CPU with DDR4 SDRAM@128 GB.

CCA-ITQ: ITQ are state-of-the-art binary hashing methods. They can be converted into supervised binary hashing methods by preprocessing feature vectors $\mathbf{X}$ using label information. Canonical correlation analysis (CCA) transformation was performed and feature vectors were normalized and set to zero mean. They generated the projection matrix $\mathbf{P}$, and binary codes were assigned by (1).

COSDISH is a recently proposed supervised hashing method. COSDISH generates the projection matrix $\mathbf{P}$, as does ITQ. The feature vectors are transformed such that they have zero mean and are normalized through variance in preprocessing. We used open-source MATLAB code, published by the authors [19], for the computation.

FastHash is a nonlinear binary hashing method. The binary codes are generated by boosted decision-tree. We used open-source MATLAB code, published by the authors [20], for the implementation.

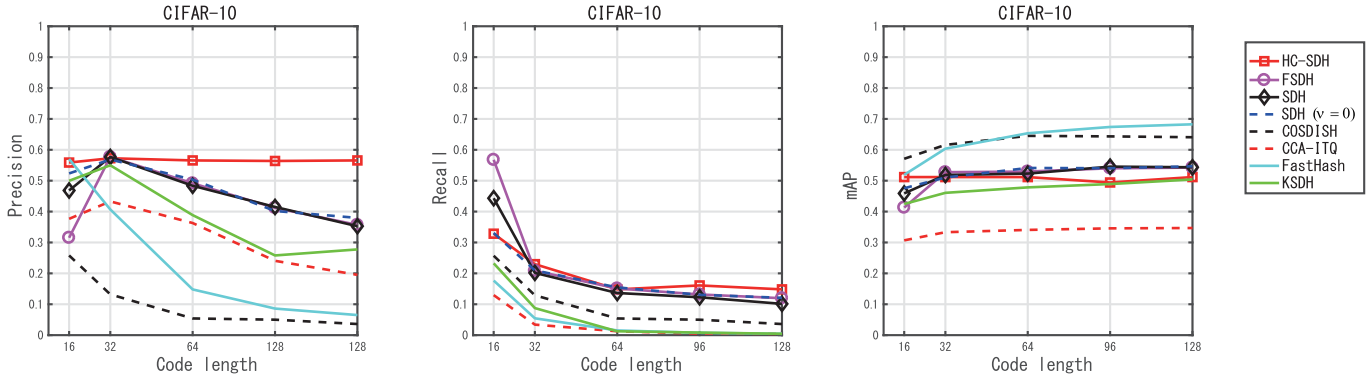

Fig. 5. Comparative results of precision, recall of Hamming distance within radius 2, and mAP for CIFAR-10 with code lengths from 16 to 128.

TABLE I

PRECISION OF HAMMING DISTANCE WITH RADIUS 2, MAP, TRAINING TIME, AND TEST TIME FOR CIFAR-10 WITH A 64-BIT CODE LENGTH

	anchor	Precision	mAP	Training time [s]	Test time [s]
SDH	1000	0.445	0.450	50.8	2.7e-05
	3000	0.352	0.543	309.4	4.5e-05
	5000	0.521	0.571	519.0	8.0e-05
SDH($\nu=0$)	3000	0.352	0.546	156.0	4.5e-05
FSDH	1000	0.446	0.429	9.9	2.7e-05
	3000	0.356	0.542	47.7	4.5e-05
	5000	0.521	0.572	96.7	8.0e-05
HC-SDH	1000	0.500	0.460	1.2	2.7e-05
	3000	0.565	0.511	6.1	4.5e-05
	5000	**0.592**	0.553	15.7	8.0e-05
KSDH	3000	0.277	0.504	1301.9	4.5e-05
CCA-ITQ	-	0.195	0.346	8.0	5.3e-06
COSDISH	-	0.036	0.640	345.1	6.2e-06
FastHash	-	0.065	**0.682**	2589.3	8.0e-04

KSDH is a *kernel-based discrete hashing* method, which is an improvement of *kernel-based supervised hashing* (KSH) [17]. KSDH generates the projection matrix $\mathbf{P}$. The feature vectors are transformed by kernel function and anchor points similar to SDH in preprocessing. Because public code of KSDH does not exist, we used the MATLAB code of KSH, published by the authors of KSH [17], and modified it to implement KSDH.

SDH and **FSDH** are well-known supervised discrete hashing methods. We used $\lambda=1$ and $\nu=10^{-5}$ with the maximum number of iterative cycles set to 5, anchor points $M=1,000$, $M=3,000$ and $M=5,000$, and kernel parameter $\sigma=0.4$ for all datasets. SDH generated the projection matrix $\mathbf{P}$, and binary codes were assigned by re-projection (1). Furthermore, to show the validity of the HC-SDH approximation, we evaluated the case where $\nu=0$ ($\text{SDH}_{\nu=0}$). We used the open-source MATLAB code, published by the authors of the SDH [7], for the computation. FSDH uses almost the same model as SDH. FSDH differs from SDH in that FSDH optimizes the binary code $\mathbf{B}$ by (11).

HC-SDH: The proposed method used the same parameters as SDH: anchor points $M=1,000$, $M=3,000$ and $M=5,000$, and kernel parameter $\sigma=0.4$ for all datasets. HC-SDH generated the projection matrix $\mathbf{P}$ and assigned binary codes through re-projection (1), as in SDH. Our code will be made available to the public,[1] and is shown in Fig. 2.

[1]https://github.com/goukoutaki/HCSDH

C. Experiment 1: Small Classes and Short Code Length

We tested two kinds of experimental settings. In the first experiment, a small number of classes $C=10$ and short code lengths $L=16, 32, 64, 96$, and 128 were used.

1) Precision, Recall, and mAP: Precision and recall were computed by calculating the Hamming distance between the training samples and the test samples with a Hamming radius of 2. Furthermore, the mAP was computed by ranking the Hamming distance. Anchor points $M=1,000$, $3,000$, and $5,000$ were used for SDH, FSDH, KSDH, and HC-SDH.

CIFAR-10: Table I provides the results of precision, mAP, training time, and test time for the CIFAR-10 dataset with a 64-bit code length. The results in the table indicate that FastHash achieved the highest mAP. The proposed HC-SDH yielded the best precision. SDH and SDH($\nu=0$) provided almost the same performance. The difference in training time is derived from the number of iterative cycles of SDH and SDH($\nu=0$) converged faster than SDH. The performance of FSDH was similar to that of SDH and the former required less time for training than the latter method. The mAP of HC-SDH is comparable to that of SDH. However, the precision of HC-SDH was high and the training time was the shortest.

All the methods SDH, FSDH, HC-SDH, and KSDH generate the binary code by a linear operation and sign function of (1) and the same feature used for kernel

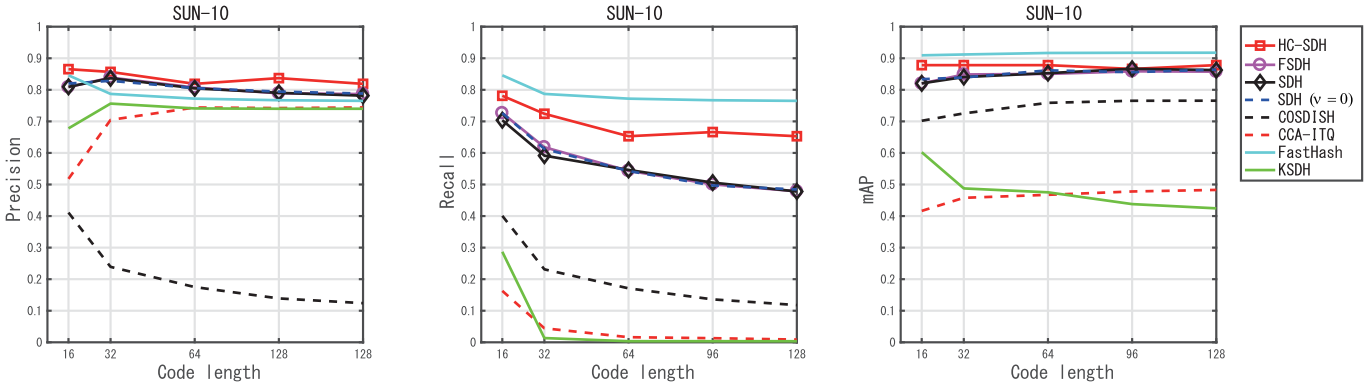

Fig. 6. Comparative results of precision, recall of Hamming distance within radius 2, and mAP for SUN-10 with code lengths from 16 to 128.

TABLE II
PRECISION AND MAP FOR MNIST AND SUN-10 WITH A 64-BIT CODE LENGTH

	MNIST		SUN-10	
	Precision	mAP	Precision	mAP
SDH	0.896	0.944	0.805	0.891
FSDH	0.899	0.950	**0.809**	0.892
HC-SDH	**0.937**	0.969	0.787	0.900
KSDH	0.224	0.734	0.740	0.470
CCA-ITQ	0.401	0.768	0.750	0.466
COSDISH	0.668	0.862	0.146	0.755
FastHash	0.828	**0.978**	0.774	**0.918**

TABLE III
COMPARISON OF PERFORMANCE WITH AND WITHOUT THE BIAS TERM

	CIFAR-10		SUN-10		MNIST	
	MAP	Pre.	MAP	Pre.	MAP	Pre.
SDH($\nu=10^{-5}$)	0.47	0.34	0.47	0.78	0.47	0.83
SDH($\nu=0$)	0.47	0.36	0.47	0.79	0.47	0.82

TABLE IV
$\|\mathbf{WY}\|^2$ AND $\nu\|\mathbf{P}^\top\mathbf{X}\|^2$ FOR ALL DATASETS FOR $\nu = 10^{-5}$

	$\|\mathbf{WY}\|^2$	$\nu\|\mathbf{P}^\top\mathbf{X}\|^2$	[%]
CIFAR-10	31.53	0.0132	0.041
SUN-10	9.16	0.0045	0.049
MNIST	22.96	0.0124	0.054
CIFAR-100	70.02	0.0147	0.020
SUN-397	58.55	0.0218	0.037
ImageNet	34.44	0.0323	0.092

transformation in Sec. II-B. Therefore, the time these methods requires for testing depends on the number of anchor points. The longest computational testing time was obtained for the kernel transformation. Although a high mAP was obtained by FastHash, it required more computational time than SDH because of the decision-tree-based binary code generation.

Figure 5 shows the results of precision, recall and mAP for the CIFAR-10 dataset with code lengths $L = 16, 32, 64, 96$, and 128. The number of anchor points $M = 3000$, was used for SDH, FSDH, HC-SDH, and KSDH. Although FastHash showed a satisfactory mAP, the precision was reduced as a function of the code length. As the code length increases, SDH and FSDH reduce the precision to the same extent as FastHash. However, the proposed HC-SDH maintains high precision and recall for any code length. This is a significant advantage of the proposed method. In general, an increase in the code length tends to decrease the precision with such a narrow threshold resulting from a Hamming radius of 2. SDH, FSDH, and SDH($\nu=0$) almost achieve the same results. The performance of HC-SDH improves when the number of anchor points increases.

SUN-10: Figure 6 shows the results of the SUN-10 dataset. The same number of anchor points was used as for CIFAR-10. In this dataset, the recall rates of the HC-SDH remained high in spite of long code lengths. When the SDH and HC-SDH had the same number of anchor points, HC-SDH was clearly superior. The mAP of FastHash was comparable to that of HC-SDH$_{5000}$, for which the number of anchor points was $M = 5,000$.

MNIST: Figure 7 shows the results of the MNIST dataset. HC-SDH yielded the best results for all datasets with the same trends. It retained high precision and recall even with long code lengths. The mAP is almost saturated for all methods.

2) ROC Curves by Hamming Ranking: Figure 8 shows the precision-recall ROC curves based on Hamming ranking. For CIFAR-10, FashHash and COSDISH show the best result. For the CIFAR-10 and MNIST datasets, SDH, FSDH, and HC-SDH show almost the same results. Increasing the number of anchor points of HC-SDH improves the performance of HC-SDH$_{5000}$ such that it becomes comparable to FastHash for SUN-10.

3) Validation of Our Approximation of SDH: We validated our approximation of SDH with $\nu = 0$ in (4) by comparing its performance with that of SDH with $\nu = 0$ under the same conditions. Table III presents the comparative results of SDH with $\nu = 10^{-5}$ and SDH with $\nu = 0$. For all datasets, the results of SDH and SDH with $\nu = 0$, were almost identical. Table IV provides $\|\mathbf{WY}\|^2$ and $\nu\|\mathbf{P}^\top\mathbf{X}\|^2$ of SDH for all datasets after optimization. We can confirm

TABLE V
TRAINING TIMES FOR CIFAR-10 [S] WITH CODE LENGTHS 16, 32, 64, 96, AND 128

	anchor	$L = 16$	$L = 32$	$L = 64$	$L = 96$	$L = 128$
SDH	1000	10.3	19.5	57.6	133.7	238.2
SDH($\nu = 0$)	1000	9.9	10.6	27.8	105.2	71.9
SDH	3000	40.6	48.4	82.3	148.1	225.7
FSDH	1000	8.0	8.1	8.6	9.5	10.2
FSDH	3000	37.1	38.0	40.3	40.5	42.7
HC-SDH	1000	1.1	1.1	1.1	1.1	1.1
HC-SDH	3000	6.0	5.7	5.6	5.5	5.6
HC-SDH	5000	13.4	14.9	13.6	14.4	13.9
KSDH	3000	1111.8	1178.9	1214.3	1276.8	1427.8
CCA-ITQ	-	0.8	1.7	3.9	6.0	8.0
COSDISH	-	5.2	18.3	75.4	177.9	335.7
FastHash	-	328.9	637.8	1283.5	1920.5	2567.8

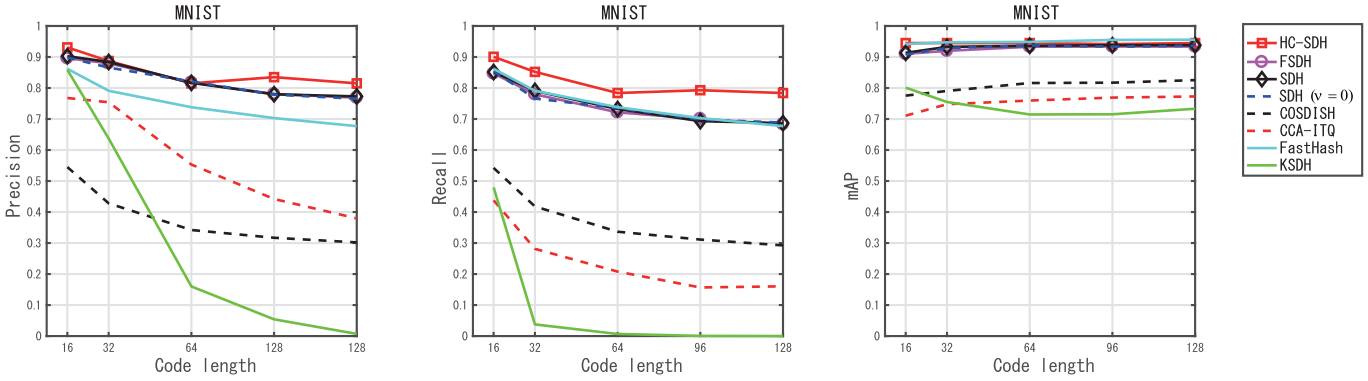

Fig. 7. Comparative results of precision, recall of Hamming distance within radius 2, and mAP for MNIST with code lengths from 16 to 128.

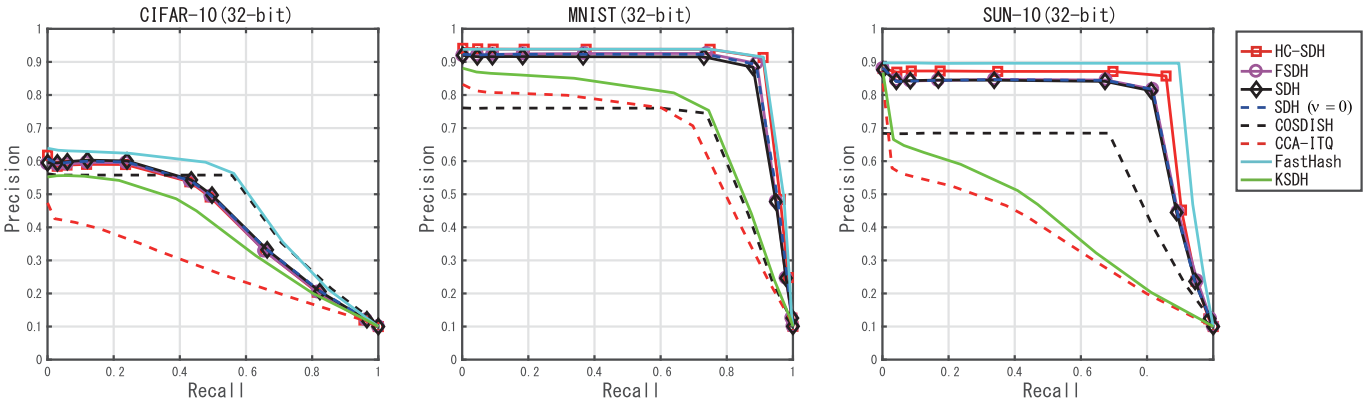

Fig. 8. Comparative results of Precision and Recall ROC curves for all datasets and all methods with 32-bit code lengths.

$\|\mathbf{WY}\|^2 \gg \nu\|\mathbf{P}^\top\mathbf{X}\|^2$. For CIFAR-10, $\nu\|\mathbf{P}^\top\mathbf{X}\|^2$ is 0.041% of $\|\mathbf{WY}\|^2$. This means that our approximation was appropriate for supervised hashing.

4) Training Time: Table V lists the training time of each method for CIFAR-10 with code length $L = 16, 32, 64, 96$, and 128 for 59,000 training samples. As the number of anchors increased, the training time increased for SDH, FSDH, and HC-SDH. The training time for SDH, KSDH, COSDISH, and FastHash increased significantly with the code length. On the other hand, the training times required by FSDH and HC-SDH are almost the same for all code lengths. The reason for the increase in the time for SDH is that the number of iterative cycles of the DCC method depended on the code length because it optimizes the binary code bitwise.

TABLE VI
LOSS COMPARISON OF SDH AND HC-SDH FOR SUN-10

L	SDH		HC-SDH	
	$\mathbf{W}$-loss	$\mathbf{P}$-loss	$\mathbf{W}$-loss	$\mathbf{P}$-loss
16	0.0124	160.7050	0.0088	151.6680
32	0.0054	224.2900	0.0044	214.4910
64	0.0028	316.3380	0.0022	303.3370
128	0.0068	448.1170	0.0011	428.9830

TABLE VII
BIT-SCALABILITY OF HC-SDH AND SDH FOR CIFAR-10 AND 10,000 TRAINING SAMPLES

	L	32	64	128	256	512	1024
HC-SDH	training time [s]	0.64	0.70	0.85	0.98	1.16	1.48
	Precision	0.50	0.47	0.47	0.47	0.47	0.47
	mAP	0.44	0.44	0.44	0.44	0.44	0.44
SDH	training time [s]	6.38	14.92	47.42	284.00	1189.49	5230.43
	Precision	0.49	0.40	0.20	0.11	0.03	0.01
	mAP	0.44	0.47	0.48	0.48	0.46	0.44

TABLE VIII
COMPARISON OF MAP, TOP-5 ACCURACY, TRAINING TIME, AND TEST TIME FOR SUN-397(C=397)

	code length	anchor	mAP	top-5 accuracy	Training time[s]	Test time [s]
SDH	32	10000	0.013	0.125	331	1.5e-4
	128	10000	0.029	0.271	615	1.5e-4
	512	10000	0.109	0.327	13231	1.5e-4
	512	30000	0.216	0.414	16167	4.7e-4
FSDH	512	10000	0.098	0.312	441	1.5e-4
	512	30000	0.189	0.385	3783	4.7e-4
HC-SDH	512	10000	0.264	0.456	77	1.5e-4
	512	30000	**0.527**	**0.564**	549	4.7e-4
FastHash	512	-	0.479	0.548	9241	3.3e-3

5) Loss Comparison: We define **W**-loss and **P**-loss of the SDH method in (4) as follows:

$$\begin{aligned} \mathbf{W}\text{-loss} &= \|\mathbf{Y} - \mathbf{W}^\top \mathbf{B}\|^2, \\ \mathbf{P}\text{-loss} &= \|\mathbf{B} - \mathbf{P}^\top \mathbf{X}\|^2. \end{aligned} \tag{24}$$

Tables VI show the two types of loss of SDH and HC-SDH after optimization of the SUN-10 datasets. As described in Sec. IV-A, HC-SDH can minimize **W**-loss exactly. Therefore, for all datasets, HC-SDH results in a lower value of **W**-loss than SDH. Furthermore, as described in Sec. IV-C, HC-SDH can also reduce **P**-loss. HC-SDH obtained a lower value of **P**-loss than SDH.

D. Experiment 2: Large Classes and Code Lengths

To validate our method for a more actual case, we evaluated the dataset with large classes and code lengths.

1) Bit-Scalability of Larger Code Lengths for CIFAR-10 and CIFAR-100: First, we show the bit-scalability of the training time for CIFAR-10. Table VII contains the comparative results in terms of computational time and performance with a wide range of code lengths $L = 32 \sim 1024$ for the CIFAR-10 dataset. In this experiment, $N = 10,000$ training samples, 1,000 test samples, and 1,000 anchors were used. The computational time of HC-SDH was almost identical in terms of the code length because the main computation in HC-SDH involved matrix multiplication and inversion $\left(\mathbf{X}\mathbf{X}^\top\right)^{-1}$ of (6). In practice, the inverse matrix was not computed directly, and a Cholesky decomposition was performed. In contrast, the computational time for SDH exponentially increased and the precision decreased significantly. This means that the DCC method converged to local minima in the case of large code lengths as discussed in Sec. III-B.

TABLE IX
THE MAP AND TOP-5 ACCURACY FOR CIFAR-100(128 BITS) AND IMAGENET(1024 BITS)

	CIFAR-100		ImageNet	
	mAP	acc.	mAP	acc.
SDH	0.198	0.288	0.008	0.023
FSDH	0.184	0.278	0.010	0.024
HC-SDH	**0.231**	**0.324**	**0.014**	**0.030**
FastHash	0.183	0.253	0.006	0.010

2) Larger Number of Classes and Larger Code Lengths: Second, we tested using three datasets with a larger number of classes as follows.

SUN-397: $C = 397$ classes with $N = 79,400$ training samples.

CIFAR-100: $C = 100$ classes with $N = 50,000$ training samples.

ImageNet: $C = 1000$ classes and we sampled $N = 50,000$ training samples from the original 1.2 million training samples.

Table VIII lists the results of mAP, top-5 accuracy, training time, and test time for the larger classes of the SUN dataset. The results of SDH show that, by increasing the number of anchor points, it becomes possible to improve the mAP. In general, a large number of bits is useful for a large number of classes. HC-SDH achieves the best mAP, accuracy, and training time compared with SDH and FSDH. When $M = 30,000$ is used, mAP $= 0.527$ can be obtained by HC-SDH. When $M = 10,000$, the training time of the proposed HC-SDH is five times faster than that of FSDH and 170 times faster than SDH.

The test times of these SDHs are similar. FastHash provides a good mAP; however, the training time and test time are longer than for HC-SDH. The time SDHs require for testing

TABLE X
PERFORMANCE FOR THE CIFAR-100 DATASET WITH OVER 128 BITS

	L	32	64	128	256	512	1024
SDH	traning time [s]	224.2	382.4	696.3	1883.3	6738.1	26351.5
	mAP	0.15	0.16	0.20	0.22	0.22	0.22
FSDH	traning time [s]	254.9	259.2	268.6	276.8	299.6	334.3
	mAP	0.14	0.18	0.18	0.21	0.21	0.19
HC-SDH	traning time [s]	-	-	40.2	42.4	42.5	45.3
	mAP	-	-	0.23	0.23	0.23	0.23

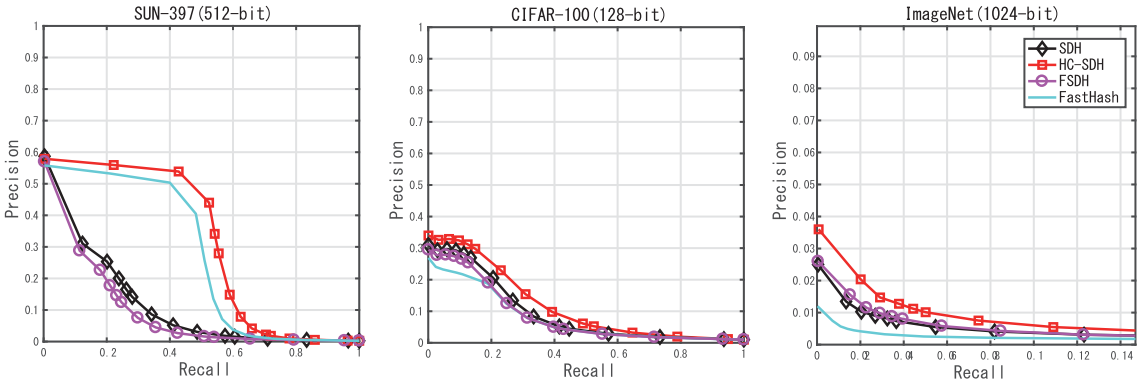

Fig. 9. Comparative results of precision and recall ROC curves for SUN-397, CIFAR-100 and ImageNet.

depends on the number of anchor points and most of the computation is derived from the kernel transformation. The test time of HC-SDH with $M = 30,000$ is seven times faster than FastHash and the mAP and top-5 accuracy of HC-SDH is superior to the results of FastHash.

Table IX lists the results of mAP, top-5 accuracy for CIFAR-100 and ImageNet datasets with 10000 anchors. As well as SUN-397, HC-SDH outperforms the other methods under the same conditions.

Figure 9 shows the precision and recall ROC curve of the SUN-397, CIFAR-100 and ImageNet datasets with 512, 128 and 1024 bits, respectively. FSDH, HC-SDH, and FastHash were evaluated and $M = 30,000$ anchor points were used for FSDH and HC-SDH. HC-SDH provided the best performance. Table XIII compares the loss between FSDH and HC-SDH for this dataset with 512 bits and $M = 30,000$. When the code length is large, FSDH cannot optimize the **W**-loss and **P**-loss and performs poorly. Although HC-SDH does not minimize the **P**-loss explicitly, as discussed in Sec. IV-C, HC-SDH reduces the **P**-loss indirectly. Actually, Table XIII shows that the **P**-loss of HC-SDH is less than that of SDH.

Moreover, Table X shows the results for CIFAR-100 with 32, 64, 128, 256, 512 and 1024-bit code lengths and $M = 10,000$ anchors. At 32 and 64 bits, we cannot use HC-SDH due to assumption A2. When the code length is less than the number of classes C, the mAP increases according to the code length. As well as the results of CIFAR-10, the proposed HC-SDH shows almost the same performance for all code lengths. The conventional SDH can improve the performance by increasing the code length, but, it cannot outperform the proposed HC-SDH.

TABLE XI
RESULTS FOR THE MULTI-LABELED NUS-WIDE DATASET

	mAP	traning time [s]
SDH	0.193	200.9
FSDH	0.197	20.2
HC-SDH	0.198	4.0

E. Computational Complexity Analysis

In the conventional SDH, **B-Step** requires approximately $\mathcal{O}(T_{\mathrm{DCC}} L^2\ NC)$, where T_{DCC} is the number of iterations of the DCC algorithm and it is set to $T_{\mathrm{DCC}} = 10$ in this study. The computational bottleneck is **P-Step**, *i.e.*, $\mathbf{P} = \left(\mathbf{X}\mathbf{X}^\top\right)^{-1}\mathbf{X}\mathbf{B}^\top$ especially the inverse of $\mathbf{X}\mathbf{X}^\top$ requiring $\mathcal{O}\left(M^3\right)$ for Cholesky factorization, [2] where M is the dimension of the feature vector $\mathbf{x}$ and it is also the number of anchor points. After adding some matrix multiplications, finally it approximately becomes $\mathcal{O}\left(M^2\ N + M^3 + MNL\right)$. The SDH iterates the steps until convergence or a maximum of $T = 5$ iterations. Although the computational cost of an iteration in the FSDH and HC-SDH is the same, the proposed method is $T = 5$ times faster than the FSDH because our method requires a one-time calculation.

[2]The computational bottleneck is at $\left(\mathbf{X}\mathbf{X}^\top\right)^{-1}$, and in our method, it is solved by the *mldivide* function of MATLAB. Since $\mathbf{X}\mathbf{X}^\top$ is a symmetric matrix, Cholesky factorization and forward and backward substitution are used for solving it. The computational complexity for each operation to solve $\mathbf{A}\mathbf{x} = \mathbf{b}$, where $\mathbf{A} := \mathbf{X}\mathbf{X}^\top$ of size $N \times N$, is as follows: $1/3\ N^3 + 2/3\ N$ in Cholesky factorization $\mathbf{A} = \mathbf{L}\mathbf{L}^\top$; N^2 in forward substitution $\mathbf{L}\mathbf{y} = \mathbf{b}$; N^2 in backward substitution $\mathbf{L}^\top\mathbf{x} = \mathbf{y}$. Additionally, other computational costs for matrix multiplication are added.

TABLE XII
INTRA-CLASS DIVERSITY OF BINARY CODES FOR EACH CLASS

	class	1	2	3	4	5	6	7	8	9	10	average
CIFAR-10	SDH	0.23	0.21	0.32	0.31	0.29	0.28	0.21	0.25	0.21	0.23	0.25
	FSDH	0.25	0.18	0.32	0.34	0.31	0.28	0.22	0.21	0.23	0.20	0.25
	HC-SDH	0.26	0.22	0.34	0.36	0.32	0.31	0.24	0.25	0.23	0.23	0.28
MNIST	SDH	0.03	0.02	0.08	0.08	0.06	0.08	0.04	0.07	0.08	0.09	0.06
	FSDH	0.03	0.02	0.07	0.07	0.06	0.08	0.04	0.06	0.09	0.07	0.06
	HC-SDH	0.03	0.02	0.07	0.07	0.05	0.07	0.03	0.06	0.08	0.07	0.05

F. Discussion and Limitation

From the two experiments, we conclude that:

- Increasing the number of anchor points of the SDHs can improve the mAP and accuracy (the maximum number of anchor points is limited to the number of training samples). This increase causes an increase in the training and test time. Thus, we need to choose the number of anchor points according to the test time.
- The training time of HC-SDH is more efficient than that of SDH and FSDH.
- Larger code lengths can improve the performance in a dataset with a larger number of classes. On the other hand, HC-SDH maintains its performance.

We assume explicitly some assumptions **A1**∼**A4** in Sec. IV and they also become the limitations of HC-SDH. **A1** is sufficiently practical and A4 has been confirmed experimentally through the evaluations for the six datasets in Table VI. **A2** means that the number of classes is limited to $C \leq L$. Usually, $L = 64 \sim 2048$-bits are used in binary hashing in practice. Although the maximum number of classes is limited to approximately 2000, it is practical for many applications within this limit. **A3** means that the proposed HC-SDH is limited to single-labeling problems. Despite this limitation, there are many applications of binary hashing for single-labeling problems such as surface categorization from a hyperspectral image in remote sensing [49], text categorization [50], analysis for DNA sequences [51], applications of intelligent transportation systems (ITS) such as traffic signs [52] and vehicle classification [53] and more.

1) Extension to Multi-Labeling Hashing of HC-SDH: With the assumption **A3**, although the proposed HC-SDH can only handle single-labeling problems, there exist some techniques to convert a single-labeling problem to multi-labeling problem [54]. Note that the final output of both the conventional SDH including multi-labeling models and the proposed HC-SDH is the projection matrix $\mathbf{P}$. A simple method to extend the HC-SDH to a multi-labeling model is to duplicate a multi-labeled training sample to multiple single-labeled training samples and then learn by HC-SDH.

Table XI shows the results of SDH, FSDH, and HC-SDH for the multi-labeled dataset NUS-WIDE [55]. In the evaluation, the dataset includes 81 classes, and 100,000 training samples and 10,000 test samples were used. $M = 1,000$ anchors were used in all methods. The proposed HC-SDH has a comparable performance to SDH and FSDH with less computational time despite the simple implementation. This indicates that the proposed HC-SDH can handle the multi-labeling problem.

TABLE XIII
LOSS COMPARISON OF FSDH AND HC-SDH FOR SUN-397 WITH 512 BITS

	$\mathbf{W}$-loss	$\mathbf{P}$-loss
FSDH	7.52	5464.56
HC-SDH	0.00275	4086.76

2) Re-Projection for Binary Codes and Intra-Class Diversity: As well as other types of the SDH methods, each binary code for test and training samples is converted by $\mathbf{b} = \mathrm{sign}\left(\mathbf{P}^\top \mathbf{x}\right)$ using the projection matrix $\mathbf{P}$ obtained in Algorithm 1. Although our optimization for HC-SDH tends to assign the same binary code to training samples in the same class, each sample has a slight different binary code due to re-projection. It can represent a unique characteristic of the samples in the same class.

To observe the intra-class diversity, we checked it by computing the diversity of some dataset. First, we defined the intra-class diversity of binary codes as

$$\sigma_c^2 := \frac{1}{N_c} \sum_{i=1}^{N_c} \|\mathbf{b}_{c,i} - \widehat{\mathbf{b}}_c\|^2, \tag{25}$$

where $\mathbf{b}_{c,i}$ is the binary code for the training sample $\mathbf{x}_i$ in class c. $\widehat{\mathbf{b}}_c$ is the mean vector of binary codes in class c. N_c is the number of samples for the class c. Table XII shows the diversity of CIFAR-10 and MNIST. The results show almost the same values, which means that HC-SDH can represent the diversity as well as SDH and FSDH.

3) Larger Number of Classes: For larger datasets with over 10,000 classes, hierarchical classification can be used. In hierarchical classification, multiple classifiers are used in classifying sub-trees [56], *e.g.*, a current version of ImageNet has 21,841 categories and those are divided into 27-subtrees. For a large-scale dataset, we can select our HC-SDH for classifying the subtrees.

VI. CONCLUSIONS

In this paper, we simplified the SDH (*supervised discrete hashing*) method to an HC-SDH (*hadamard coded-supervised discrete hashing*) method by approximating the bias term, and provided exact solutions for the proposed SDH method. We revealed that the exact solution can be provided by the Hadamard matrix. The proposed SDH approximation was validated by comparative experiments with the SDH method. Unlike conventional SDH, the HC-SDH does not require

alternating optimization which would cause it to converge to local minima. The HC-SDH is easy to implement. The experimental results showed that our method outperformed several state-of-the-art supervised hashing methods. In particular, for large code lengths, HC-SDH can maintain performance without losing precision. For the large datasets SUN-397 and ImageNet, HC-SDH provided the best mAP and top-accuracy compared to the conventional SDH methods with the same code length and FastHash. The training time of HC-SDH is 170 times faster than conventional SDH and the testing time is seven times faster than FastHash.

Our idea, which avoids using alternating optimization by using a Hadamard matrix, is easily applicable to other supervised hashing models such as the cross-modal model [57]. This model uses multiple features of not only images but also text or video data. Here, in [57], the problem of alternating optimization and its limitation was discussed. Our Hadamard coding assigns one category to one binary code. Therefore, it is an example of the vector quantization method in binary space. For the assignment, the idea of collective matrix factorization [58] may be useful. In future work, we plan to extend Hadamard coding to a kind of vector quantization approach to accelerate the search time.

APPENDIX

A. Proof of Lemma 1

The optimization problem in (16) is known as the *resource allocation problem* [59]–[61]. Here we present a simple proof for the solution.

The constraint $\sum_{i=1}^{N} x_i = L$ can be regarded as a surface equation in an N-dimensional space $(x_1, x_2, \ldots, x_N)$. On the other hand, the gradient vector of the object function $\sum_{i=1}^{N} f(x_i)$ is defined as

$$\mathbf{g} := [f'(x_1), f'(x_2), \ldots, f'(x_N)]^\top \tag{26}$$

where $f'(\cdot)$ is the differentiated version of $f(\cdot)$ and the i-th element (the gradient in the i-th direction) is given by $\frac{\partial}{\partial x_i}\sum_j f(x_j) = \sum_j \frac{\partial x_j}{\partial x_i}\frac{\partial}{\partial x_j} f(x_j)$, and $\frac{\partial x_j}{\partial x_i}$ becomes 1 if $i = j$ or 0 if $i \neq j$.

Then, the gradient along the surface is obtained as the projection of $\mathbf{g}$ onto the surface, and computed as the inner-product of $\mathbf{g}$ and a set of vectors $\{\mathbf{n}^\perp\}$ perpendicular to the normal vector of the surface:

$$\mathbf{n} := \tfrac{1}{\sqrt{N}}[1, 1, \ldots, 1]^\top \in \mathbb{R}^N, \tag{27}$$

and the projected gradient $\mathbf{g}^\top \mathbf{n}^\perp$ becomes 0 at the global extremum point on the surface. This also indicates $\mathbf{g}$ and $\mathbf{n}$ are parallel and their inner-product becomes

$$\left(\frac{\mathbf{g}}{\|\mathbf{g}\|_2}\right)^\top \mathbf{n} = 1 \quad \Rightarrow \quad \mathbf{g}^\top \mathbf{n} = \|\mathbf{g}\|_2. \tag{28}$$

Substituting (26) and (27) into (28), we get

$$\tfrac{1}{\sqrt{N}} \sum_i f'(x_i) = \sqrt{\sum_i f'(x_i)^2} \tag{29}$$

Additionally, when we express $f'(x_i)$ as $\sqrt{f'(x_i)^2}$ and $\frac{1}{\sqrt{N}}$ as $\frac{1}{N}\sqrt{N}$, we get

$$\tfrac{1}{N} \sum_i \sqrt{f'(x_i)^2} = \sqrt{\tfrac{1}{N} \sum_i f'(x_i)^2}. \tag{30}$$

The shape of this equality actually corresponds to Jensen's inequality: $\sum_i p_i h(y_i) \geq h(\sum_i p_i y_i)$ where $\sum_i p_i = 1$, and the equality holds if and only if $\{y_i\}$, *i.e.*, $\{f'(x_i)^2\}$ are all equal:

$$f'(x_1)^2 = f'(x_2)^2 = \ldots = f'(x_N)^2 \tag{31}$$

Additionally, when $f(\cdot)$ is a convex function, $f'(\cdot)$ becomes an injective function because $f''(\cdot) \geq 0$ is a monotonically increasing function. Also, if the sign of $f'(\cdot)$ does not change within the valid range of x_i (the case considered in this paper), $f'(\cdot)^2$ becomes injective. Hence,

$$x_1 = x_2 = \ldots = x_N. \tag{32}$$

Finally, substituting (32) into the condition $\sum_{i=1}^{N} x_i = L$, we get

$$\forall_i \; x_i = \frac{L}{N}. \tag{33}$$

B. Details of (17) to (18)

We consider the expansion of (17) as

$$\begin{aligned}&\|\mathbf{I} - \mathbf{W}^\top \mathbf{B}'\|^2 + \lambda \|\mathbf{W}\|^2 \\ &\quad = \|\mathbf{I}\|^2 - 2\mathrm{Tr}(\mathbf{W}^\top \mathbf{B}') + \mathrm{Tr}(\mathbf{W}^\top \mathbf{B}' \mathbf{B}'^\top \mathbf{W}) + \lambda \mathrm{Tr}(\mathbf{W}\mathbf{W}^\top).\end{aligned} \tag{34}$$

Here, we use $\|\mathbf{I}\|^2 = C$, $\mathbf{W} = \left(\mathbf{B}'\mathbf{B}'^\top + \lambda \mathbf{I}\right)^{-1} \mathbf{B}'$, and $\mathbf{B}'^\top \mathbf{B}' = \mathbf{U}^\top \mathbf{D} \mathbf{U}$ by eigendecomposition with an orthogonal matrix $\mathbf{U}^{-1} = \mathbf{U}^\top$ and $\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top \mathbf{U} = \mathbf{I}$. Here, note the following relationship to the inverse matrix:

$$\begin{aligned}\left(\mathbf{B}'\mathbf{B}'^\top + \lambda \mathbf{I}\right)^{-1} &= \left(\mathbf{U}^\top \mathbf{D} \mathbf{U} + \lambda \mathbf{U}^\top \mathbf{I} \mathbf{U}\right)^{-1} \\ &= \mathbf{U}^\top (\mathbf{D} + \lambda \mathbf{I})^{-1} \mathbf{U}.\end{aligned} \tag{35}$$

Using the property of exchanging order of trace $\mathrm{Tr}(\mathbf{AB}) = \mathrm{Tr}(\mathbf{BA})$ and $\mathrm{Tr}(\mathbf{D}) = \sum_{i=1}^{C} \sigma_i$ and $\mathrm{Tr}(\mathbf{D} + \lambda \mathbf{I})^{-1} = \sum_{i=1}^{C} \frac{1}{\sigma_i + \lambda}$, we get

$$\begin{aligned}(34) &= C - 2\mathrm{Tr}(\mathbf{W}^\top \mathbf{B}') + \mathrm{Tr}(\mathbf{W}^\top \mathbf{B}' \mathbf{B}'^\top \mathbf{W}) + \lambda \mathrm{Tr}(\mathbf{W}\mathbf{W}^\top) \\ &= \sum_{i=1}^{C} \left(1 - 2\frac{\sigma_i}{\sigma_i + \lambda} + \frac{\sigma_i^2}{(\sigma_i + \lambda)^2} + \frac{\lambda \sigma_i}{(\sigma_i + \lambda)^2}\right) \\ &= \sum_{i=1}^{C} \frac{\lambda}{\sigma_i + \lambda}.\end{aligned} \tag{36}$$

ACKNOWLEDGEMENT

The authors would like to thank Dr. Yusuke Matsui for providing valuable advice related to the PQ-type hashing algorithm.

REFERENCES

[1] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *Proc. Int. Conf. Very Large Data Bases (VLDB)*, 1999, pp. 518–529.

[2] 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.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[3] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2009, pp. 1042–1050.

[4] 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. (ICML)*, 2013, pp. 142–150.

[5] V. A. Nguyen, J. Lu, and M. N. Do, "Supervised discriminative hashing for compact binary codes," in *Proc. ACM Multimedia Conf. (ACMMM)*, 2014, pp. 989–992.

[6] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen, "Learning binary codes for maximum inner product search," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4148–4156.

[7] F. Shen, C. Shen, W. Liu, and H. T. Shen, "Supervised discrete hashing," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 37–45. [Online]. Available: https://github.com/bd622/DiscretHashing

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2010, pp. 778–792.

[9] S. Gog and R. Venturini, "Fast and compact Hamming distance index," in *Proc. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2016, pp. 285–294.

[10] Y. Cao *et al.*, "Binary hashing for approximate nearest neighbor search on big data: A survey," *IEEE Access*, vol. 6, pp. 2039–2054, 2018.

[11] J. Gui, T. Liu, D. Tao, Z. Sun, and T. Tan, "Representative vector machines: A unified framework for classical classifiers," *IEEE Trans. Cybern.*, vol. 46, no. 8, pp. 1877–1888, Aug. 2016.

[12] W. Liu, J. Wang, and S. F. Chang, "Hashing with graphs," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2011, pp. 1–22.

[13] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2014, pp. 3419–3427.

[14] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical Hashing: Binary Code Embedding with Hyperspheres," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 11, pp. 2304–2316, Nov. 2015.

[15] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2009, pp. 1753–1760.

[16] H. Hotelling, "Relations between two sets of variates," *Biometrika*, vol. 28, nos. 3–4, pp. 312–377, 1936.

[17] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2012, pp. 2074–2081. [Online]. Available: http://www.ee.columbia.edu/~wliu/

[18] X. Shi, F. Xing, J. Cai, Z. Zhang, Y. Xie, and L. Yang, "Kernel-based supervised discrete hashing for image retrieval," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2016, pp. 419–433.

[19] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2016, pp. 1230–1236. [Online]. Available: http://cs.nju.edu.cn/lwj/

[20] 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. Comput. Vis. Pattern Recognit. (CVPR)*, Apr. 2014, pp. 1971–1978. [Online]. Available: https://bitbucket.org/chhshen/fasthash/

[21] A. Schrijver, *Theory of Linear and Integer Programming*. Hoboken, NJ, USA: Wiley, 1986.

[22] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Recognit. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2017.

[23] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Supervised discrete hashing with relaxation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 3, pp. 608–617, Mar. 2018.

[24] Y. Guo, G. Ding, and J. Han, "Robust quantization for general similarity search," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 949–963, Feb. 2018.

[25] X. Liu, Z. Li, C. Deng, and D. Tao, "Distributed adaptive binary quantization for fast nearest neighbor search," *IEEE Trans. Image Process.*, vol. 26, no. 11, pp. 5324–5336, Nov. 2017.

[26] X. Liu, Y. Mu, D. Zhang, B. Lang, and X. Li, "Large-scale unsupervised hashing with shared structure learning," *IEEE Trans. Cybern.*, vol. 45, no. 9, pp. 1811–1822, Sep. 2015.

[27] X. Liu, B. Du, C. Deng, M. Liu, and B. Lang, "Structure sensitive hashing with adaptive product quantization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2252–2264, Oct. 2016.

[28] Y. Guo, G. Ding, L. Liu, J. Han, and L. Shao, "Learning to hash with optimized anchor embedding for scalable retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 3, pp. 1344–1354, Mar. 2017.

[29] G. Ding, J. Zhou, Y. Guo, Z. Lin, S. Zhao, and J. Han, "Large-scale image retrieval with sparse embedded hashing," *Neurocomputing*, vol. 257, pp. 24–36, Sep. 2017.

[30] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 2475–2483.

[31] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 1183–1192.

[32] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.

[33] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 744–755, Apr. 2014.

[34] X. Wang, T. Zhang, G.-J. Qi, J. Tang, and J. Wang, "Supervised quantization for similarity search," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2018–2026.

[35] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 4, pp. 769–790, Apr. 2018.

[36] H. Jain, J. Zepeda, P. Pérez, and R. Gribonval, "SuBiC: A supervised, structured binary code for image search," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 833–842.

[37] I. S. Dhillon and S. Sra, "Generalized nonnegative matrix approximations with Bregman divergences," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2005, pp. 283–290.

[38] M. Slawski, M. Hein, and P. Lutsik, "Matrix factorization with binary components," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2013, pp. 3210–3218.

[39] G. Koutaki, "Binary continuous image decomposition for multi-view display," *ACM Trans. Graph*, vol. 35, no. 4, pp. 69:1–69:12, 2016.

[40] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, "A fast optimization method for general binary code learning," *IEEE Trans. Image Process.*, vol. 25, no. 12, pp. 5610–5621, 2016.

[41] J. J. Sylvester, "Thoughts on inverse orthogonal matrices, simultaneous sign successions, and tessellated pavements in two or more colours, with applications to Newton's rule, ornamental tile-work, and the theory of numbers," *Philos. Mag.*, vol. 34, no. 232, pp. 461–475, 1867.

[42] J. Hadamard, "Résolution ́dune question relative aux déterminants," *Bull. Sci. Math.*, vol. 17, no. 2, pp. 240–246, 1893.

[43] A. Krizhevsky, "Learning multiple layers of features from tiny images," Dept. Comput. Sci., Univ. Toronto, Toronto, ON, Canada, Tech. Rep., 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/cifar.html

[44] 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. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2010, pp. 3485–3492. [Online]. Available: http://groups.csail.mit.edu/vision/SUN/

[45] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[46] 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. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2009, pp. 248–255. [Online]. Available: http://www.image-net.org

[47] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou, "How should we evaluate supervised hashing?" in *Proc. IEEE Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2017, pp. 1732–1736.

[48] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *Int. J. Comput. Vis.*, vol. 42, no. 3, pp. 145–175, 2001.

[49] B. Pan, Z. Shi, X. Xu, and Y. Yang, "Hashing based hierarchical feature representation for hyperspectral imagery classification," *Remote Sens.*, vol. 9, no. 11, p. 1094, 2017.

[50] S. Chaidaroon and Y. Fang, "Variational deep semantic hashing for text documents," in *Proc. Int. ACM SIGIR Conf. Res. Devel. Inf. Retr. (SIGIR)*, 2017, pp. 75–84.
[51] C. Caragea, A. Silvescu, and P. Mitra, "Protein sequence classification using feature hashing," in *Proc. IEEE Int. Conf. Bioinf. Biomed. (BIBM)*, Nov. 2011, pp. 538–543.
[52] A. de la Escalera, J. M. Armingol, and M. Mata, "Traffic sign recognition and analysis for intelligent vehicles," *Image Vis. Comput.*, vol. 21, no. 3, pp. 247–258, 2003.
[53] J. Krause, J. Deng, M. Stark, and F.-F. Li, "Collecting a large-scale dataset of fine-grained cars," in *Proc. 2nd Workshop Fine-Grained Vis. Categorization (CVPR workshop)*, 2013, pp. 1–2.
[54] G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data," in *Data Mining and Knowledge Discovery Handbook*. New York, NY, USA: Springer-Verlag, 2010, pp. 667–685.
[55] 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. ACM Conf. Image Video Retr. (CIVR)*, 2009, p. 48.
[56] A. Cevahir and K. Murakami, "Large-scale multi-class and hierarchical product categorization for an E-commerce giant," in *Proc. Int. Conf. Comput. Linguistics (COLING)*, 2016, pp. 525–535.
[57] X. Xu, F. Shen, Y. Yang, H. T. Shen, and X. Li, "Learning discriminative binary codes for large-scale cross-modal retrieval," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2494–2507, May 2017.
[58] G. Ding, Y. Guo, J. Zhou, and Y. Gao, "Large-scale cross-modality search via collective matrix factorization hashing," *IEEE Trans. Image Process.*, vol. 25, no. 11, pp. 5427–5440, Jun. 2016.
[59] R. E. Bellman and S. E. Dreyfus, *Applied Dynamic Programming*. Princeton, NJ, USA: Princeton Univ. Press, 1962.
[60] S. E. Dreyfus and A. M. Law, *The Art and Theory of Dynamic Programming*. San Francisco, CA, USA: Academic, 1977.
[61] T. Ibaraki and N. Katoh, *Resource Allocation Problems: Algorithmic Approaches*. Cambridge, MA, USA: MIT Press, 1988.

Gou Koutaki received the B.E., M.E., and D.Eng. degrees from Kumamoto University, Japan, in 2002, 2004, and 2007, respectively. He joined the Production Engineering Research Laboratory, Hitachi, Ltd., in 2007. He is currently an Associate Professor with Kumamoto University. His research interests are image processing and machine vision.

Keiichiro Shirai received the B.E., M.E., and D.Eng. degrees from Keio University, Yokohama, Japan, in 2001, 2003, and 2006, respectively. He is currently an Associate Professor with Shinshu University, Japan. His current research interests include signal processing, image processing, and computer vision.

Mitsuru Ambai received the B.E., M.S., and Ph.D. degrees in information and computer science from Keio University in 2002, 2004, and 2007, respectively. He is currently a Senior Engineer with the Research and Development Group, Denso IT Laboratory, Inc., Tokyo, Japan. His research interests include image processing and computer vision.