Simultaneous Compression and Quantization:
A Joint Approach for Efficient Unsupervised Hashing

Tuan Hoang[a,*], Thanh-Toan Do[b], Huu Le[c], Dang-Khoa Le-Tan[a], Ngai-Man Cheung[a]

[a]Singapore University of Technology and Design, Singapore
[b]University of Liverpool, Liverpool, UK
[c]Queensland University of Technology, Brisband, Australia

Abstract

For unsupervised data-dependent hashing, the two most important requirements are to preserve similarity in the low-dimensional feature space and to minimize the binary quantization loss. A well-established hashing approach is Iterative Quantization (ITQ), which addresses these two requirements in separate steps. In this paper, we revisit the ITQ approach and propose novel formulations and algorithms to the problem. Specifically, we propose a novel approach, named **Simultaneous Compression and Quantization (SCQ)**, to jointly learn to compress (reduce dimensionality) and binarize input data in a single formulation under strict orthogonal constraint. With this approach, we introduce a loss function and its relaxed version, termed Orthonormal Encoder (OnE) and Orthogonal Encoder (OgE) respectively, which involve challenging binary and orthogonal constraints. We propose to attack the optimization using novel algorithms based on recent advance in cyclic coordinate descent approach. Comprehensive experiments on unsupervised image retrieval demonstrate that our proposed methods consistently outperform other state-of-the-art hashing methods. Notably, our proposed methods outperform recent deep neural networks and GAN based hashing in accuracy, while being very computationally-efficient.

1. Introduction

For decades, image hashing has been an active research field in vision community (Andoni and Indyk (2008); Gong and Lazebnik (2011); Weiss et al. (2009); Zhang et al. (2010)) due to its advantages in storage and computation speed for similarity search/retrieval under specific conditions (Gong and Lazebnik (2011)). Firstly, the binary code should be short so as to the whole hash table can fit in the memory. Secondly, the binary code should preserve the similarity, i.e., (dis)similar images have (dis)similar hashing codes in the Hamming distance space. Finally, the algorithm to learn parameters should be fast and for unseen samples, the hashing method should produce the hash codes efficiently. It is very challenging to simultaneously satisfy all three requirements, especially, under the binary constraint which leads to an NP-hard mixed-integer optimization problem. In this paper, we aim to tackle all these challenging conditions and constraints.

The proposed hashing methods in literature can be categorized into data-independence (Gionis et al. (1999); Kulis and Grauman (2009); Raginsky and Lazebnik (2009)) and data-dependence; in which, the latter recently receives more attention in both (semi-)supervised (Do et al. (2016b); Kulis and Darrell (2009); Lin et al. (2014); Liu et al. (2012); Norouzi et al. (2012); Shen et al. (2015); Chen et al. (2018); Cao et al. (2018); Jain et al. (2017); Liu et al. (2016); Lin et al. (2015);

Lai et al. (2015); Lin et al. (2016)) and unsupervised (Carreira-Perpiñán and Raziperchikolaei (2015); Do et al. (2016a, 2017, 2019); Gong and Lazebnik (2011); He et al. (2013); Heo et al. (2012); Shen et al. (2018); Hu et al. (2018); Huang and Lin (2018); y. Duan et al. (2018); Wang et al. (2018); Duan et al. (2017); En et al. (2017); Do et al. (2019)) manners. Supervised hashing have shown superior performance over unsupervised hashing. However, in practice, labeled datasets are limited and costly; hence, in this work, we focus only on the unsupervised setting. We refer readers to recent surveys (Grauman and Fergus (2013); Wang et al. (2015, 2014, 2017)) for more detailed reviews of data-independent/dependent hashing methods.

1.1. Related works

The most relevant work to our proposal is Iterative Quantization (ITQ) (Gong and Lazebnik (2011)), which is a very fast and competitive hashing method. The fundamental of ITQ is two folds. Firstly, to achieve low-dimensional features, it uses the well-known Principle Component Analysis (PCA) method. PCA maximizes the variance of projected data and keeps dimensions pairwise uncorrelated. Hence, the low-dimension data, projected using the top PCA component vectors, can preserve data similarity well. Secondly, minimizing the binary quantization loss using an orthogonal rotation matrix strictly maintains the data pairwise distance. As a result, ITQ learns binary codes that can highly preserve the local structure of the data. However, optimizing these two steps separately, especially when no binary constraint is enforced in the first step, i.e., PCA, leads to suboptimal solutions. In contrast, we propose to jointly optimize the projection variance and the quantization loss.

---

[*]Corresponding author:
Email address: `nguyenanhtuan_hoang@mymail.sutd.edu.sg` (Tuan Hoang)

Other works that are highly relevant to our proposed method are Binary Autoencoder (BA) (Carreira-Perpiñán and Raziperchikolaei (2015)), UH-BDNN (Do et al. (2016a)), DBD-MQ (Duan et al. (2017)), and Stacked convolutional AutoEncoders (SAE) (En et al. (2017)). In these methods, the authors proposed to combine the data dimension reduction and binary quantization into a single step by using encoder of autoencoder, while the decoder encourages (dis)similar inputs map to (dis)similar binary codes. However, the reconstruction criterion is not a direct way for preserving the similarity (Do et al. (2016a)). Additionally, although achieving very competitive performances, UH-BDNN and DBD-MQ are based on the deep neural network (DNN); hence, it is difficult to produce the binary code computationally-efficiently. Particularly, given an extracted CNN feature, these methods require a forward propagation through multiple fully-connected and activation layers to produce the binary code. While our proposed method only requires a single linear transformation, i.e., one BLAS operation (`gemv` or `gemm`), and a comparison operation.

Recently, many works (Liny et al. (2016); Duan et al. (2017); Song (2018)) leverage the powerful capability of Convolution Neuron Network (CNN) to jointly learn the image representations and binary codes. However, due to the non-smooth property of the binary constraint causing the ill-gradient in back-propagation, these methods resort to relaxation or approximation. As a result, even thought achieving high-discriminative image representations, these methods can only produce sub-optimal binary codes. In the paper, we show that by directly considering the binary constraint, our methods can obtain much better binary codes. Hence, higher retrieval performances can be achieved. This emphasizes the necessity of having an effective method to preserve the discrimination power of high-dimensional CNN features in very compact binary representations, i.e., effectively handling the challenging binary and orthogonal constraints.

Besides, several works has been proposed to handle the difficulty of training deep models with the binary constraint. Cao et al. (2017) proposed to handles the non-smooth problem of the sign function by continuation, i.e., starting the training with a smoothed approximation and gradually reducing the smoothness as the training proceeds, i.e., $\lim_{\beta \to \infty} \tanh(\beta x) = \text{sign}(x)$. Chen et al. (2018) transformed the original binary optimization into differentiable optimization problem over hash functions through Taylor series expansion. Cao et al. (2018) introduced a pairwise cross-entropy loss based on the Cauchy distribution, which penalizes significantly similar image pairs with Hamming distance larger than the given Hamming radius threshold, e.g., greater than 2. Nevertheless, these methods requires class labels for the training process (i.e., supervised hashing). This is not the focus of our methods which aim to learn optimal binary codes from given image representations in the unsupervised manner.

1.2. Contributions

In this work, to address the problem of learning to preserve data affinity in low-dimension binary codes, *(i)* we first propose a novel loss function to learn a single linear transformation un-der the *column orthonormal constraint*[1] in the unsupervised manner that *compresses* and *binarizes the input data jointly*. The approach is named as **Simultaneous Compression and Quantization (SCQ)**. Noted that the idea of jointly compressing and binarizing data has been explored in Carreira-Perpiñán and Raziperchikolaei (2015); Do et al. (2016a). However, due to the difficulty of the non-convex orthogonal constraint, these works try to relax the orthogonal constraint and resort to the reconstruction criterion as an indirect way to handle the similarity perserving concern. Our work is the first one to tackle the similarity concern by enforcing ***strict*** orthogonal constraints.

(ii) Under the strict orthogonal constraints, we conduct analysis and experiments to show that our formulation is able to retain a high amount of the variance, i.e., preserve data similarity, and achieve small quantization loss, which are important requirements in hashing for image retrieval (Gong and Lazebnik (2011); Carreira-Perpiñán and Raziperchikolaei (2015); Do et al. (2016a)). As a result, this leads to improved accuracy as demonstrated in our experiments.

(iii) We then propose to relax the *column **orthonormal** constraint* to *column **orthogonal** constraint* on the transformation matrix. The relaxation not only helps to gain extra retrieval performances but also significantly improves the training time.

(iv) Our proposed loss functions, with column orthonormal and orthogonal constraints, are confronted with two main challenges. The first is the binary constraint, which is the traditional and well-known difficulty of hashing problem (Andoni and Indyk (2008); Gong and Lazebnik (2011); Weiss et al. (2009)). The second challenge is the non-convex nature of the orthonormal/orthogonal constraint (Wen and Yin (2013)). To tackle the binary constraint, we propose to apply an alternating optimization with an auxiliary variable. Additionally, we resolve the orthonormal/orthogonal constraint by using the cyclic coordinate descent approach to learn one column of the projection matrix at a time while fixing the others. The proposed algorithms are named as *Orthonormal Encoder (OnE)* and *Orthogonal Encoder (OgE)*.

(v) Comprehensive experiments on common benchmark datasets show considerable improvements on retrieval performance of proposed methods over other state-of-the-art hashing methods. Additionally, the computational complexity and training / online-processing time are also discussed to show the computational efficiency of our methods.

1.3. Notations and Term definitions

We first introduce the notations. Given a zero-centered dataset $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{n \times d}$ which consists of n images and each image is represented by a d-dimension feature descriptor, our proposed hashing methods aim to learn a column orthonormal/orthogonal matrix $\mathbf{V} \in \mathbb{R}^{d \times L}(L \ll d)$ which simultaneously compresses input data $\mathbf{X}$ to L-dimensional space, while retains a high amount of variance, and quantizes to binary codes $\mathbf{B} \in \{-1, +1\}^{n \times L}$.

It is important to note that, in this work, we abuse the terms: ***column orthonormal/orthogonal matrix***. Specifically,

---

[1] Please refer to section 1.3 for our term definitions.

the term **column orthonormal matrix** is used to indicate the matrix $\mathbf{V}$ that $\mathbf{V}^\top\mathbf{V} = \mathbf{I}_{L\times L}$, where $\mathbf{I}_{L\times L}$ is the $L \times L$ identity matrix. While the term **column orthogonal matrix** indicates matrix $\mathbf{V}$ that $\mathbf{V}^\top\mathbf{V} = \mathbf{D}_{L\times L}$, where $\mathbf{D}_{L\times L}$ is an arbitrary $L \times L$ diagonal matrix. Noted that the word "column" in these terms means that columns of the matrix are pairwise independent.

We define $\Lambda = [\lambda_1, \cdots, \lambda_d]$ as the eigenvalues of the covariance matrix $\mathbf{X}^T\mathbf{X}$ sorted in descending order. Finally, let $\mathbf{b}_k, \mathbf{v}_k$ be the k-th ($1 \le k \le L$) columns of $\mathbf{B}, \mathbf{V}$ respectively.

The remainder of the paper is organized as follow. Firstly, Section 2 presents in details our proposed hashing method, i.e., **Orthonormal Encoder (OnE)** and provide the analysis to show that our method can retain a high amount of variance and achieve small quantization loss. Section 3 presents a relax version of OnE, i.e., **Orthogonal Encoder (OgE)**. Section 4 presents experiment results to validate the effectiveness of our proposed methods. We conclude the paper in Section 5.

2. Simultaneous Compression & Quantization: Orthonormal Encoder

2.1. Problem Formulation

In order to jointly learn data dimension reduction and binary quantization using a single linear transformation $\mathbf{V}$, we propose to solve the following constrained optimization:

$$\arg\min_{\mathbf{B},\mathbf{V}} Q(\mathbf{B}, \mathbf{V}) = \frac{1}{n}\|\mathbf{B} - \mathbf{X}\mathbf{V}\|_F^2 \tag{1}$$
$$\text{s.t. } \mathbf{V}^\top\mathbf{V} = \mathbf{I}_{L\times L}; \mathbf{B} \in \{-1, +1\}^{n\times L},$$

where $\|\cdot\|_F$ denotes the Frobenius norm. Additionally, the orthonormal constrained on the column of $\mathbf{V}$ is necessary to make sure no redundant information is captured in binary codes (Wang et al. (2012)) (i.e., the projected low-dimensional features are strictly pairwise uncorrelated) and the projection vectors do not scale up/down projected data.

It is noteworthy to highlight the differences between our loss function Eq. (1) and the binary quantization loss function of ITQ (Gong and Lazebnik (2011)). Firstly, different from ITQ, which works on the compressed low-dimensional feature space after using PCA, i.e., $\mathbf{X} \in \mathbf{R}^{n\times L}$; our approach, instead, works directly on the original high-dimensional feature space $\mathbf{X} \in \mathbf{R}^{n\times d}(d \gg L)$. This leads to the second main difference that the non-square column orthonormal matrix $\mathbf{V} \in \mathbb{R}^{d\times L}$ simultaneously *(i) compresses data* to low-dimension and *(ii) quantizes to binary codes*. However, it is important to note that solving for a non-square projection matrix $\mathbf{V}$ is challenging. To handle this difficulty, ITQ propose to solve the data compression and binary quantization problems in two separated optimizations. Specifically, it applys PCA to compress data to L dimension, and then uses the Orthogonal Procrustes approach (Schönemann (1966)) to learn a $L\times L$ square rotation matrix to optimize binary quantization loss. However, there is a limitation in ITQ approach as no consideration for the binary constraint in the data compression step, i.e., PCA. Consequently, the solution is suboptimal. In this paper, by adopting recent advance in cyclic coordinate descent approach (Shen

et al. (2015); Do et al. (2016a); Gurbuzbalaban et al. (2017); Yuan and Ghanem (2017)), we propose a novel and efficient algorithm to resolve the ITQ limitation by simultaneously attacking both problems in a single optimization problem under the strict orthogonal constraint. Hence, our optimization can lead to a better optimal solution.

2.2. Optimization

In this section, we discuss the key details of the algorithm (Algorithm 1) for solving the optimization problem Eq. (1). In order to handle the binary constraint in Eq. (1), we propose to use alternating optimization over $\mathbf{V}$ and $\mathbf{B}$.

2.2.1. Fix $\mathbf{V}$ and update $\mathbf{B}$

When $\mathbf{V}$ is fixed, the problem becomes exactly the same as when fixing rotation matrix in ITQ. To make the paper self-contained, we repeat the explaination of Gong and Lazebnik (2011). By expanding the objective function in Eq. (1), we have

$$Q(\mathbf{B}, \mathbf{V}) = \frac{1}{n}\left(\|\mathbf{B}\|_F^2 + \|\mathbf{U}\|_F^2 - 2\text{tr}(\mathbf{B}\mathbf{U})\right)$$
$$= \frac{1}{n}\left(nL + \|\mathbf{U}\|_F^2 - 2\text{tr}(\mathbf{B}\mathbf{U})\right), \tag{2}$$

where $\mathbf{U} = \mathbf{X}\mathbf{V}$. Because $\mathbf{V}$ is fixed, so $\mathbf{U}$ is fixed, minimizing (2) is equivalent to maximizing

$$\text{tr}(\mathbf{B}\mathbf{U}) = \sum_{i=1}^{n}\sum_{j=1}^{L} B_{ij}U_{ij} \tag{3}$$

Algorithm 1 Orthonormal Encoder

Input:
  $\mathbf{X} = \{\mathbf{x}\}_{i=1}^{n} \in \mathbb{R}^{n\times d}$: training data;
  L: code length;
  max_iter: maximum iteration number;
  $\{\epsilon, \epsilon_b, \epsilon_u\}$: convergence error-tolerances;
Output
  Column Orthonormal matrix $\mathbf{V}$.

1: Randomly initialize $\mathbf{V}$ such that $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$.
2: **for** $t = 1 \rightarrow$ *max_iter* **do**
3:   **procedure** FIX $\mathbf{V}$, UPDATE $\mathbf{B}$.
4:     Compute $\mathbf{B}$ (Eq. (4)).
5:   **procedure** FIX $\mathbf{B}$, UPDATE $\mathbf{V}$.
6:     Find ν_1 using binary search (BS) (Eq. (8)).
7:     Compute $\mathbf{v}_1$ (Eq. (7)).
8:     **for** $k = 2 \rightarrow L$ **do**
9:       **procedure** SOLVE $\mathbf{v}_k$
10:         Initialize $\Phi_k = [0, \cdots, 0]$.
11:         **while** true **do**
12:           Fix Φ_k, solve for ν_k using BS.
13:           Fix ν_k, compute $\Phi_k = A_k^{-1}c_k$.
14:           Compute $\mathbf{v}_k$ (Eq. (12)).
15:           **if** $(\mathbf{v}_k^\top\mathbf{v}_k - 1) < \epsilon_u$ **then**
16:             **return** $\mathbf{v}_k$
17:     **if** $t > 1$ and $(Q_{t-1} - Q_t)/Q_t < \epsilon$ **then break**
18: **return** $\mathbf{V}$

where B_{ij} and U_{ij} denotes elements of $\mathbf{B}$ and $\mathbf{U}$ respectively. To maximize this expression with respect to $\mathbf{B}$, we need to have $B_{ij} = 1$ whenever $U_{ij} \geq 0$ and $B_{ij} = -1$ otherwise. Hence, the optimal value of $\mathbf{B}$ can be simply achieved by

$$\mathbf{B} = sign(\mathbf{XV}). \tag{4}$$

2.2.2. *Fix $\mathbf{B}$ and update $\mathbf{V}$*

When fixing $\mathbf{B}$, the optimization is no longer a mix-integer problem. However, the problem is still non-convex and difficult to solve due to the orthonormal constraint (Wen and Yin (2013)). It is important to note that $\mathbf{V}$ is not a square matrix. It means that the objective function is not the classic Orthogonal Procrustes problem (Schönemann (1966)). Hence, we cannot achieve the closed-form solution for $\mathbf{V}$ as proposed in Gong and Lazebnik (2011). To the best of our knowledge, there is no easy way for achieving the closed-form solution of non-square $\mathbf{V}$. Hence, in order to overcome this challenge, inspired by PCA and recent methods in cyclic coordinate descent (Shen et al. (2015); Do et al. (2016a); Gurbuzbalaban et al. (2017); Yuan and Ghanem (2017)), we iteratively learn one vector, i.e., one column of $\mathbf{V}$, at a time. We now consider two cases for $k = 1$ and $2 \leq k \leq L$.

• **1-st vector**

$$\arg \min_{\mathbf{v}_1} Q_1 = \frac{1}{n}\|\mathbf{b}_1 - \mathbf{X}\mathbf{v}_1\|^2 \quad \text{s.t. } \mathbf{v}_1^\top \mathbf{v}_1 = 1, \tag{5}$$

where $\|\cdot\|$ is the $l2$-norm.

Let $v_1 \in \mathbb{R}$ be the Lagrange multiplier, we formulate the *Lagrangian $\mathcal{L}_1$*:

$$\mathcal{L}_1(\mathbf{v}_1, v_1) = \frac{1}{n}\|\mathbf{b}_1 - \mathbf{X}\mathbf{v}_1\|^2 + v_1(\mathbf{v}_1^\top \mathbf{v}_1 - 1). \tag{6}$$

By minimizing $\mathcal{L}_1$ over $\mathbf{v}_1$, we can achieve:

$$\mathbf{v}_1 = (\mathbf{X}^\top \mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top \mathbf{b}_1, \tag{7}$$

given v_1 that maximizes the *dual function $\mathcal{G}_1(v_1)$* [2] of $\mathcal{L}_1(\mathbf{v}_1, v_1)$ (Boyd and Vandenberghe (2004)). Equivalently, v_1 should satisfy the following conditions:

$$\begin{cases} v_1 > -\lambda_d/n \\ \frac{\partial \mathcal{G}_1}{\partial v_1} = [(\mathbf{X}^\top \mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top \mathbf{b}_1]^\top \\ \qquad [(\mathbf{X}^\top \mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top \mathbf{b}_1] - 1 = 0 \end{cases} \tag{8}$$

where λ_d is the smallest eigenvalue of $\mathbf{X}^\top \mathbf{X}$. The detail derivation is provided in Appendix section A.

In Eq. (8), the first condition is to ensure that $(\mathbf{X}^\top \mathbf{X} + nv_1\mathbf{I})$ is non-singular and the second condition is achieved by setting the derivative of $\mathcal{G}_1(v_1)$ with regard to v_1 equal to 0.

The second equation in Eq. (8) can be recognized as a *d*-order polynomial equation of v_1 which has no explicit closed-form solution for v_1 when $d > 4$. Fortunately, since $\mathcal{G}(v_1)$ is a

---

concave function of v_1, $\partial \mathcal{G}_1/\partial v_1$ is monotonically decreasing. Hence, we can simply solve for v_1 using binary search with a small error-tolerance ϵ_b. Note that:

$$\begin{cases} \lim_{v_1 \to (-\lambda_d/n)^+} \frac{\partial \mathcal{G}_1}{\partial v_1} = +\infty \\ \lim_{v_1 \to +\infty} \frac{\partial \mathcal{G}_1}{\partial v_1} = -1 \end{cases}, \tag{9}$$

thus $\partial \mathcal{G}_1/\partial v_1 = 0$ always has a solution.

• ***k*-th vector** ($2 \leq k \leq L$)

For the second vector onward, besides the unit-norm constraint, we also need to ensure that the current vector is independent with its $(k-1)$ previous vectors.

$$\arg \min_{\mathbf{v}_k} Q_k = \frac{1}{n}\|\mathbf{b}_k - \mathbf{X}\mathbf{v}_k\|^2$$
$$\text{s.t. } \mathbf{v}_k^\top \mathbf{v}_k = 1; \mathbf{v}_k^\top \mathbf{v}_i = 0, \forall i \in [1, k-1]. \tag{10}$$

Let $v_k \in \mathbb{R}$ and $\Phi_k = [\phi_{k1}, ..., \phi_{k(k-1)}]^\top \in \mathbb{R}^{(k-1)}$ be the Lagrange multipliers, we also formulate the *Lagrangian $\mathcal{L}_k$*:

$$\mathcal{L}_k(\mathbf{v}_k, v_k, \Phi_k) = \frac{1}{n}\|\mathbf{b}_k - \mathbf{X}\mathbf{v}_k\|^2$$
$$+ v_k(\mathbf{v}_k^\top \mathbf{v}_k - 1) + \sum_{i=1}^{k-1} \phi_{ki}\mathbf{v}_k^\top \mathbf{v}_i. \tag{11}$$

Minimizing $\mathcal{L}_k$ over $\mathbf{v}_k$, similar to Eq. (7), we can achieve:

$$\mathbf{v}_k = (\mathbf{X}^\top \mathbf{X} + nv_k\mathbf{I})^{-1}\left(\mathbf{X}^\top \mathbf{b}_k - \frac{n}{2}\sum_{i=1}^{k-1} \phi_{ki}\mathbf{v}_i\right), \tag{12}$$

given $\{v_k, \Phi_k\}$ that satisfy the following conditions which make the corresponding *dual function $\mathcal{G}_k(v_k, \Phi_k)$* maximum:

$$\begin{cases} v_k > -\lambda_d/n \\ \mathbf{v}_k^\top \mathbf{v}_k = 1 \\ \mathbf{A}_k\Phi_k = \mathbf{c}_k \end{cases} \tag{13}$$

where

$$\begin{cases} \mathbf{A}_k = \frac{n}{2}\begin{bmatrix} \mathbf{v}_1^\top \mathbf{Z}_k\mathbf{v}_1 & \cdots & \mathbf{v}_1^\top \mathbf{Z}_k\mathbf{v}_{(k-1)} \\ \vdots & \ddots & \vdots \\ \mathbf{v}_{(k-1)}^\top \mathbf{Z}_k\mathbf{v}_1 & \cdots & \mathbf{v}_{(k-1)}^\top \mathbf{Z}_k\mathbf{v}_{(k-1)} \end{bmatrix} \\ \mathbf{c}_k = \begin{bmatrix} \mathbf{v}_1^\top \mathbf{Z}_k\mathbf{X}^\top \mathbf{b}_k & \cdots & \mathbf{v}_{(k-1)}^\top \mathbf{Z}_k\mathbf{X}^\top \mathbf{b}_k \end{bmatrix}^\top \end{cases} \tag{14}$$

in which $\mathbf{Z}_k = (\mathbf{X}^\top \mathbf{X} + nv_k\mathbf{I})^{-1}$. The detail derivation is provided in Appendix section B.

There is also no straight-forward solution for $\{v_k, \Phi_k\}$. In order to resolve this difficulty, we propose to use alternative optimization to solve for v_k and Φ_k. In particular, (i) given a fixed Φ_k (initialized as $[0, \cdots, 0]^\top$), we find v_k using binary search as discussed above. Additionally, similar to v_1, there is always a solution for v_k. Then, (ii) with fixed v_k, we can get the closed-form solution for Φ_k as $\Phi_k = \mathbf{A}_k^{-1}\mathbf{c}_k$. Note that since the dual function $\mathcal{G}_k$ is a concave function of $\{v_k, \Phi_k\}$, alternative

---

[2] The dual function $\mathcal{G}_1(v_1)$ can be simply constructed by substituting $\mathbf{v}_1$ from Eq. (7) into Eq. (6).
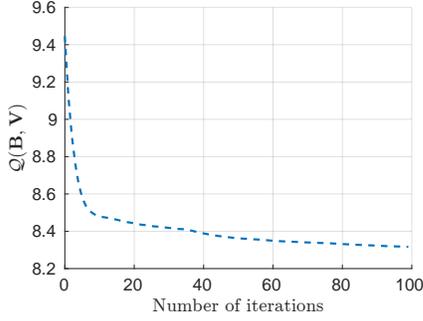
Figure 1: Quantization error for learning the projection matrix $\mathbf{V}$ with $L = 32$ on the CIFAR-10 dataset (section 4.1).

Figure 2: An illustration of the relationship between the *minimizing quantization loss* and *maximizing retained variance* problems.

optimizing between ν_k and Φ_k still guarantees the solution to approach the global optimal one.

Additionally, we note that solving for $\{\nu_k, \Phi_k\}$ requires a matrix inversion $\mathbf{Z}_k^{-1}$ (for each ν_k), which is very computationally expensive. However, by utilizing the Singular Value Decomposition (SVD), we can efficiently compute the inversion as follows:

$$(\mathbf{X}^\top \mathbf{X} + n\nu_k \mathbf{I})^{-1} = \hat{\mathbf{U}} \hat{\mathbf{\Sigma}} \hat{\mathbf{U}}^\top, \qquad (15)$$

where $\hat{\mathbf{U}} \in \mathbb{R}^{d \times d}$ is the matrix of eigenvectors corresponding to $\hat{\Lambda}$ in columns ($\hat{\Lambda} = [\lambda_d, \cdots, \lambda_1]$ is the eigenvalues of $\mathbf{X}^\top \mathbf{X}$ sorted in *ascending* order) and

$$\hat{\mathbf{\Sigma}} = \text{diag}\left(\left[\frac{1}{\lambda_d + n\nu_k}, \cdots, \frac{1}{\lambda_1 + n\nu_k}\right]\right) \qquad (16)$$

with "diag($\cdot$)" is the operation to convert vectors to square diagonal matrices. Note that, given $\mathbf{X}$, $\hat{\mathbf{U}}$ and $\hat{\Lambda}$ are fixed and can be computed in advance.

Figure 1 shows an error convergence curve of the optimization problem Eq. (1). We stop the optimization when the relative reduction of the quantization loss is less than ϵ, i.e., $(Q_{t-1} - Q_t)/Q_t < \epsilon$.

2.3. Retained variance and quantization loss

In the hashing problem for image retrieval, both retained variance and quantization loss are important. In this section, we provide analysis to show that, when solving Eq. (1), it is possible to retain a high amount of the variance and achieve small quantization loss. As will be discussed in more details, this can be accomplished by applying an appropriate scale S on the input dataset. Noticeably, by applying any positive scale $s > 0$ [3] on the dataset, the local structure of data is strictly preserved, i.e., the ranking nearest neighbor set of every data point is always the same. Therefore, in the hashing problem for retrieval task, it is equivalent to work on a scaled version of the dataset, i.e., $\mathbf{X}_s = s\mathbf{X}$. We can re-write the loss function of Eq. (1) as following:

$$Q(s, \mathbf{V}) = \|\mathbf{1} - s|\mathbf{XV}|\|_F^2 \quad \text{s.t. } \mathbf{V}^\top \mathbf{V} = \mathbf{I}_{L \times L}; s > 0, \qquad (17)$$

---

[3] For simplicity, we only discuss positive value $s > 0$. Negative value $s < 0$ should have similar effects.
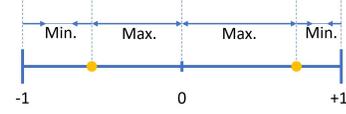
where $|\cdot|$ is the element-wise operation to find the absolute values and $\mathbf{1}$ is the all-1 ($n \times L$) matrix. In what follows, we discuss how s can affect the retained variance and quantization loss.

2.3.1. Maximizing retained variance

We recognize that by scaling to the dataset $\mathbf{X}$ by an appropriate scale s, such that all projected data points are inside the hyper-cube of 1, i.e., $\max(s|\mathbf{XV}|) \leq 1$, the *maximizing retained variance problem* (PCA) can achieve similar results to the *minimizing quantization loss problem*, i.e., $\arg\max_{\mathbf{V}} \|s\mathbf{XV}\|_F^2 \approx \arg\min_{\mathbf{V}} \|\mathbf{1} - s|\mathbf{XV}|\|_F^2$. Intuitively, we can interpret the former problem, i.e., PCA, as to find the projection that maximizes the distances of projected data points from the coordinate origin. While the latter problem, i.e., minimizing binary quantization loss, tries to find the projection matrix that minimizes the distances of projected data points from -1 or $+1$ correspondingly. A simple 1-D illustration to explain the relationship between two problems is given in Figure 2.

Since each vector of $\mathbf{V}$ is constrained to have the unit norm, the condition $\max(s|\mathbf{XV}|) \leq 1$ actually can be satisfied by scaling the dataset by $s_{\text{max\_var}}$ to have all data points in the original space inside the hyper-ball with unit radius, in which $1/s_{\text{max\_var}}$ is equal to the largest $l2$-distance between data points and the coordinate origin.

2.3.2. Minimizing quantization loss

Regarding the quantization loss $Q(s, \mathbf{V})$ (Eq. 17), which is a convex function of $s|\mathbf{XV}|$, by setting $\partial Q(s, \mathbf{V})/\partial s|\mathbf{XV}| = \mathbf{0}$, we have the optimal solution for $Q(s, \mathbf{V})$ as following:

$$\frac{\partial Q(s, \mathbf{V})}{\partial s|\mathbf{XV}|} = 2(s|\mathbf{XV}| - \mathbf{1}) = \mathbf{0} \Leftrightarrow \begin{cases} \text{mean}(|\mathbf{XV}|) = 1/s \\ \text{var}(|\mathbf{XV}|) = 0 \end{cases} \qquad (18)$$

where $\mathbf{0}$ is the all-0 ($n \times L$) matrix.

Considering $s \geq s_{\text{max\_var}}$, there are two important findings. Firstly, there is obviously no scaling value s that can concurrently achieve $s \max(|\mathbf{XV}|) \leq 1$ and $s \text{mean}(|\mathbf{XV}|) = 1$, except the case $s|\mathbf{XV}| = \mathbf{1}$ which is unreal in practice. Secondly, from Eq. (18), we can recognize that as s gets larger, i.e., $1/s$ gets smaller, minimizing the loss Q will produce $\mathbf{V}$ that focuses on lower-variance directions so as to achieve smaller mean($|\mathbf{XV}|$) as well as smaller var($|\mathbf{XV}|$). It means that $s|\mathbf{XV}|$ gets closer to the global minimum of $Q(s, \mathbf{V})$. Consequently, the quantization loss becomes smaller. In Figure 3, we show a toy example to illustrate that as s increases, minimizing quantization loss diverts the projection vector from top-PCA component (Figure 3a) to smaller variance directions (Figure 3b → 3c), while the quantization loss (per bit) gets smaller (Figure 3a → 3c). In summary, as var($|\mathbf{XV}|$) gets smaller, the quantization loss is
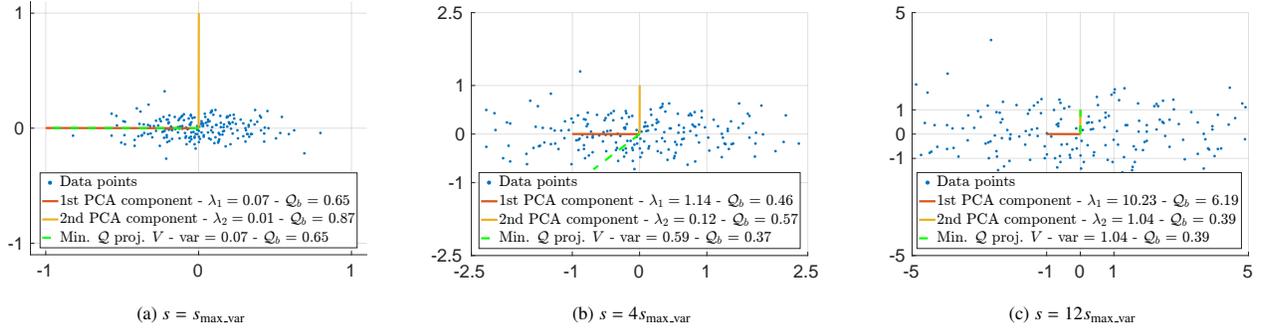
Figure 3: A toy example for $n = 200, d = 2$ and $L = 1$ to illustrate how the quantization loss and the minimizing quantization loss vector (green dash line) vary when s increases. The values in legends present the variances and the quantization losses per bit, Q_b, of the data which is projected in corresponding vectors (rounding to two decimal places).

smaller and vice versa. However, note that keeping increasing s when $\mathbf{V}$ already focuses on least-variance directions will make the quantization loss larger.

Note that the scale s is a hyper-parameter in our system. In the experiment section (Section 4.2), we will additionally conduct experiments to quantitatively analyse the effect of the scale hyper-parameter s and determine proper values using validation dataset.

3. Simultaneous Compression & Quantization: Orthogonal Encoder

3.1. Problem Re-formulation: Orthonormal to Orthogonal

In Orthonormal Encoder (OnE), we work with the *column orthonormal* constraint on $\mathbf{V}$. However, we recognize that relaxing this constraint to *column orthogonal* constraint, i.e., relaxing the unit norm constraint on each column of $\mathbf{V}$, by converting it into a penalty term, provides three important advantages. We now achieve the new loss function as following:

$$\arg \min_{\mathbf{B},\mathbf{V}} Q(\mathbf{B},\mathbf{V}) = \frac{1}{n}\|\mathbf{B} - \mathbf{XV}\|_F^2 + \mu \sum_{i=1}^{L} \mathbf{v}_i^\top \mathbf{v}_i \tag{19}$$
$$\text{s.t. } \mathbf{B} \in \{-1,+1\}^{n\times L}; \mathbf{v}_i^\top \mathbf{v}_j = 0, \forall i \neq j,$$

where μ is a fixed positive hyper-parameter to penalize large norms of $\mathbf{v}_i$. It is important to note that, in Eq. (19), we still enforce the strict pairwise independent constraint of projection vectors to ensure no redundant information is captured.

Firstly, with an appropriately large μ, the optimization prefers to choose large variance components of $\mathbf{X}$ since this helps to achieve the projection vectors that have smaller norms. In other words, without penalizing large norms of $\mathbf{v}_i$, the optimization has no incentive to focus on high variance components of $\mathbf{X}$ since it can produce projection vectors with arbitrary large norms that can scale any components appropriately to achieve minimum binary quantization loss. Secondly, this provides more flexibility of having different scale values for different directions. Consequently, relaxing the unit-norm constraint of each column of $\mathbf{V}$ helps to mitigate the difficulty of choosing the scale value s. However, it is important to note

that a too large μ, on the other hand, may distract the optimization from minimizing the binary quantization term. Finally, from OnE Optimization (Section 2.2), we observed that the unit norm constraint on each column of $\mathbf{V}$ makes the OnE optimization difficult to be solved efficiently since there is no closed-form solution for $\{\mathbf{v}\}_{k=1}^L$. By relaxing this unit norm constraint, we now can achieve the closed-form solutions for $\{\mathbf{v}\}_{k=1}^L$; hence, it is very computationally beneficial. We will discuss more about the computational aspect in section 3.3.

3.2. Optimization

Similar to the Algorithm 1 for solving Orthonormal Encoder, we apply alternative optimize $\mathbf{V}$ and $\mathbf{B}$ with the $\mathbf{B}$ step is exactly the same as Eq. (4). For $\mathbf{V}$ step, we also utilize the cyclic coordinate descent approach to iteratively solve $\mathbf{V}$, i.e., column by column. The loss functions are rewritten and their corresponding closed-form solutions for $\{\mathbf{v}\}_{k=1}^L$ can be efficiently achieved as following:

- **1-st vector**

$$\arg \min_{\mathbf{v}_1} Q_1 = \frac{1}{n}\|\mathbf{b}_1 - \mathbf{Xv}_1\|^2 + \mu \mathbf{v}_1^\top \mathbf{v}_1. \tag{20}$$

We can see that Eq. (20) is the regularized least squares problem, whose closed-form solution is given as:

$$\mathbf{v}_1 = (\mathbf{X}^\top \mathbf{X} + n\mu \mathbf{I})^{-1} \mathbf{X}^\top \mathbf{b}_1. \tag{21}$$

- **k-th vector** ($2 \leq k \leq L$)

$$\arg \min_{\mathbf{v}_k} Q_k = \frac{1}{n}\|\mathbf{b}_k - \mathbf{Xv}_k\|^2 + \mu \mathbf{v}_k^\top \mathbf{v}_k \tag{22}$$
$$\text{s.t. } \mathbf{v}_k^\top \mathbf{v}_i = 0, \forall i \in [1, k-1].$$

Given the Lagrange multiplier $\Phi_k = [\phi_{k1}, \cdots, \phi_{k(k-1)}]^\top \in \mathbb{R}^{(k-1)}$, similar to Eq. (7) and Eq. (11), we can obtain $\mathbf{v}_k$ as following:

$$\mathbf{v}_k = (\mathbf{X}^\top \mathbf{X} + n\mu \mathbf{I})^{-1} \left(\mathbf{X}^\top \mathbf{b}_k - \frac{n}{2} \sum_{i=1}^{k-1} \phi_{ki} \mathbf{v}_i \right), \tag{23}$$

Algorithm 2 Orthogonal Encoder

Input:

$\mathbf{X} = \{\mathbf{x}\}_{i=1}^{n} \in \mathbb{R}^{n \times d}$: training data;

L: code length;

max_iter: maximum iteration number;

ϵ: convergence error-tolerance;

Output

Column Orthogonal matrix $\mathbf{V}$.

1: Randomly initialize $\mathbf{V}$ such that $\mathbf{V}^\top \mathbf{V} = \mathbf{I}$.
2: **for** $t = 1 \rightarrow$ *max_iter* **do**
3:     Fix $\mathbf{V}$, update $\mathbf{B}$: Compute $\mathbf{B}$ (Eq. (4)).
4:     Fix $\mathbf{B}$, update $\mathbf{V}$: Compute $\{\mathbf{v}_i\}_{i=1}^{L}$ (Eq. (21), (23)).
5:     **if** $t > 1$ and $(Q_{t-1} - Q_t)/Q_t < \epsilon$ **then break**
6: **return** $\mathbf{V}$

---

where $\Phi_k = \mathbf{A}_k^{-1}\mathbf{c}_k$, in which

$$\begin{cases} \mathbf{A}_k = \frac{n}{2} \begin{bmatrix} \mathbf{v}_1^\top \mathbf{Z} \mathbf{v}_1 & \cdots & \mathbf{v}_1^\top \mathbf{Z} \mathbf{v}_{(k-1)} \\ \vdots & \ddots & \vdots \\ \mathbf{v}_{(k-1)}^\top \mathbf{Z} \mathbf{v}_1 & \cdot & \mathbf{v}_{(k-1)}^\top \mathbf{Z} \mathbf{v}_{(k-1)} \end{bmatrix} \\ \mathbf{c}_k = \begin{bmatrix} \mathbf{v}_1^\top \mathbf{Z} \mathbf{X}^\top \mathbf{b}_k & \cdots & \mathbf{v}_{(k-1)}^\top \mathbf{Z} \mathbf{X}^\top \mathbf{b}_k \end{bmatrix}^\top \end{cases} \quad (24)$$

and $\mathbf{Z} = (\mathbf{X}^\top \mathbf{X} + n\mu \mathbf{I})^{-1}$.

Note that, given a fixed μ, $\mathbf{Z}$ is a constant matrix, the $(k-1) \times (k-1)$ matrix $\mathbf{A}_k$ contains $(k-2) \times (k-2)$ matrix $\mathbf{A}_{(k-1)}$ in the top-left corner. It means that only the $(k-1)$-th row and column of matrix $\mathbf{A}_k$ are needed to be computed. Thus, Φ_k can be solved even more effectively.

Finally, similar to OnE (Fig. 1), we also empirically observe the convergence of the optimization problem Eq. 19. We summarize the Orthogonal Encoder method in Algorithm 2.

3.3. Complexity analysis

The complexity of the two algorithms, OnE and OgE, are shown in Table 1. In our empirical experiments, t is usually around 50, t_1 is at most 10 iterations, and $d^2 \gg n$ (for CNN fully-connected features (Section 4.1)). Firstly, we can observe that OgE is very efficient as its complexity is only linearly depended on the number of training samples n, feature dimension d, and code length L. In addition, OgE is also faster than OnE. Furthermore, as our methods aim to learn the projection matrices that preserve high-variance components, it is unnecessary to work on very high dimensional features. As there are many low-variance/noisy components, which will be discarded eventually. More importantly, we observe no retrieval performance drop when applying PCA to compress features to a much lower dimension, e.g., 512-D, in comparison with using the original 4096-D features. While this helps to achieve significant speedup in training time for both algorithms, especially for the OnE, as its time complexity is depended on d^3 for large d. In addition, we conduct experiments to measure the actual running time of the algorithms and compare with other methods in section 4.4.

Table 1: Computational complexity of algorithm OnE and OgE. where n is the number of training samples, d is the feature dimension, t is the number of iteration to alternative update $\mathbf{B}$ and $\mathbf{V}$, and t_1 is the number of iterations for solving $\mathbf{v}_k$ in Algorithm 1.

	Computational complexity
OnE	$O(tt_1 dL(\max(n, d^2)))$
OgE	$O(tdLn)$

4. Experiments

4.1. Datasets, Evaluation protocols, and Implementation notes

The **CIFAR-10** dataset (Krizhevsky and Hinton (2009)) contains $60,000$ fully-annotated color images of 32×32 from 10 object classes ($6,000$ images for each class). The provided test set ($1,000$ images for each class) is used as the query set. The remaining 50,000 images are used as the training set and the database.

The **LabelMe-12-50k** dataset (Uetz and Behnke (2009)) consists of $50,000$ fully annotated color images of 256×256 of 12 object classes, which is a subset of LabelMe dataset (Russell et al. (2008)). In this dataset, for any image having multiple label values in the range of $[0.0, 1.0]$, the object class of the largest label value is chosen as the image label. We also use the provided test set as the query set and the remaining images as the training set and the database.

The **SUN397** dataset (Xiao et al. (2016)) contains approximately $108,000$ fully annotated color images from 397 scene categories. We select a subset of 42 categories which contain more than 500 images per category to construct a dataset of approximately $35,000$ images in total. We then randomly sample 100 images per class to form the query set. The remaining images are used as the training set and the database.

For these above image datasets, each image is represented by a 4096-D feature vector extracted from the fully-connected layer 7 of pre-trained VGG (Simonyan and Zisserman (2014)).

Evaluation protocols. As datasets are fully annotated, we use semantic labels to define the ground truths of image queries. We apply three standard evaluation metrics, which are widely used in literature (Carreira-Perpiñán and Raziperchikolaei (2015); Erin Liong et al. (2015); Gong and Lazebnik (2011)), to measure the retrieval performance of all methods: 1) mean Average Precision (*mAP*(%)); 2) precision at Hamming radius of 2 (*prec@r2* (%)) which measures precision on retrieved images having Hamming distance to query ≤ 2 (we report zero precision for queries that return no image); 3) precision at top 1000 return images (*prec@1k* (%)) which measures the precision on the top 1000 retrieved images.

Implementation notes. As discussed in section 3.3, for computational efficiency, we apply PCA to reduce the feature dimension to 512-D for our proposed methods. The hyperparameter μ of OgE algorithm is empirically set as 0.02 for all experiments. Finally, for both OnE and OgE, we set all error-tolerance values, $\epsilon, \epsilon_b, \epsilon_n$, as 10^{-4} and the maximum number of iteration is set as 100. The implementation of our methods is available at `https://github.com/hnanhtuan/SCQ.git`.
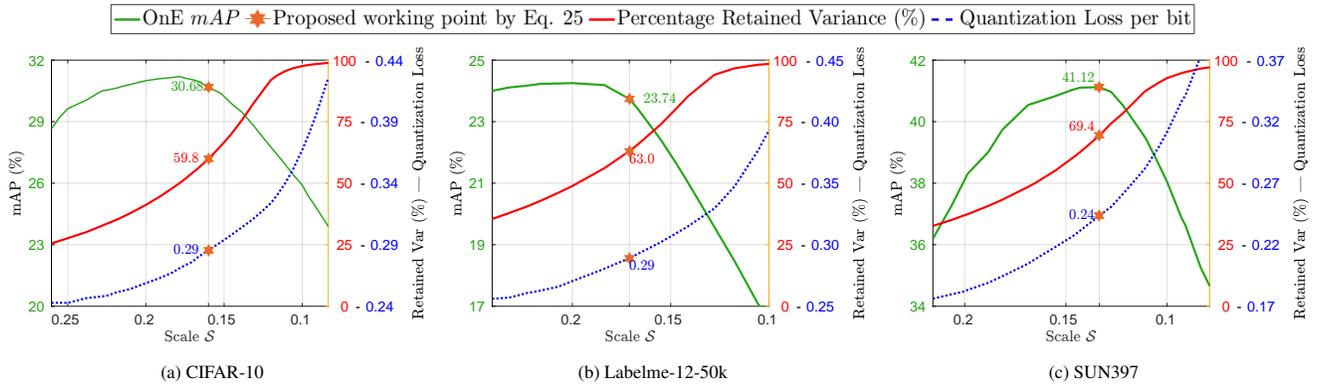
Figure 4: Analyzing the effects of the scale value s on *(i)* the quantization loss per bit (blue dash line with blue right Y-axis), *(ii)* the percentage of total retained variance by the minimizing quantization loss projection matrix (using Algorithm 1) in comparison with the total retained variance of top-L PCA components (red line with red right Y-axis), and *(iii)* the retrieval performance in mAP (green line with green left Y-axis). Note that x-axis is in descending order.

Table 2: Performance comparison with the state-of-the-art unsupervised hashing methods. The **Bold** and <u>Underline</u> values indicate the **best** and <u>second best</u> performances respectively.

	Dataset	**CIFAR-10**				**LabelMe-12-50k**				**SUN397**			
	L	8	16	24	32	8	16	24	32	8	16	24	32
mAP	SpH	17.09	18.77	20.19	20.96	11.68	13.24	14.39	14.97	9.13	13.53	16.63	19.07
	KMH	22.22	24.17	24.71	24.99	16.09	16.18	16.99	17.24	21.91	26.42	28.99	31.87
	BA	23.24	24.02	24.77	25.92	17.48	17.10	17.91	18.07	20.73	31.18	35.36	36.40
	ITQ	24.75	26.47	26.86	27.19	17.56	17.73	18.52	19.09	20.16	30.95	35.92	37.84
	SCQ - OnE	**27.08**	**29.64**	<u>30.57</u>	<u>30.82</u>	<u>19.76</u>	<u>21.96</u>	**23.61**	<u>24.25</u>	<u>23.37</u>	<u>34.09</u>	<u>38.13</u>	<u>40.54</u>
	SCQ - OgE	<u>26.98</u>	<u>29.33</u>	**30.65**	**31.15**	**20.63**	**23.07**	<u>23.54</u>	**24.68**	**23.44**	**34.73**	**39.47**	**41.82**
prec@r2	SpH	18.04	30.58	37.28	21.40	11.72	19.38	25.14	13.66	6.88	23.68	37.21	27.39
	KMH	21.97	36.64	42.33	27.46	15.20	26.17	32.09	18.62	9.50	<u>36.14</u>	**51.27**	39.29
	BA	23.67	38.05	42.95	23.49	16.22	25.75	31.35	13.14	**10.50**	**37.75**	<u>50.38</u>	41.11
	ITQ	<u>24.38</u>	**38.41**	<u>42.96</u>	28.63	15.86	25.46	31.43	17.66	<u>9.78</u>	35.15	49.85	46.34
	SCQ - OnE	**24.48**	36.49	41.53	<u>43.90</u>	**16.69**	27.30	<u>34.63</u>	<u>33.04</u>	8.68	30.12	43.54	<u>50.41</u>
	SCQ - OgE	24.35	<u>38.30</u>	**43.01**	**44.01**	<u>16.57</u>	**27.80**	**34.77**	**34.64**	8.76	29.31	45.03	**51.88**
prec@1k	SpH	22.93	26.99	29.50	31.98	14.07	16.78	18.52	19.27	10.79	15.36	18.21	20.07
	KMH	32.30	33.65	35.52	37.77	21.07	20.97	21.41	21.98	18.94	24.93	25.74	28.26
	BA	31.73	34.16	35.67	37.01	21.14	21.71	22.64	22.83	19.22	28.68	31.31	31.80
	ITQ	32.40	36.35	37.25	37.96	21.01	22.00	22.98	23.63	18.86	28.62	31.56	32.74
	SCQ - OnE	<u>33.38</u>	<u>37.82</u>	<u>39.13</u>	<u>40.40</u>	<u>22.91</u>	<u>25.39</u>	<u>26.55</u>	<u>27.16</u>	<u>19.26</u>	29.95	<u>32.72</u>	<u>34.08</u>
	SCQ - OgE	**33.41**	**38.33**	**39.54**	**40.70**	**23.94**	**25.94**	**26.99**	**27.46**	**20.10**	29.95	**33.43**	**35.00**

For all compared methods, e.g., Spherical Hashing (SpH) (Heo et al. (2012)), K-means Hashing (KMH)[4] (He et al. (2013)), Binary Autoencoder (BA) (Carreira-Perpiñán and Raziperchikolaei (2015)), and Iterative Quantization (ITQ) (Gong and Lazebnik (2011)); we use the implementation with suggested parameters provided by the authors. Besides, to improve the statistical stability of the results, we report the average values of 5 executions.

4.2. Effects of parameters

As discussed in section 2.3, when s decreases, the projection matrix **V** can be learned to retain a very high amount of

variance, as much as PCA can. However, it causes undesirable large binary quantization loss and vice versa. In this section, we additionally provide quantitative analysis of the effects of the scale parameter on these two factors (i.e., the amount of retained variance and the quantization loss) and, moreover, on the retrieval performance.

In this experiment, for all datasets, e.g., CIFAR-10, LabelMe-12-50k, and SUN397, we random select 20 images for each class in the training set (as discussed in section 4.1) for validation set. The remaining images are used for training. To obtain each data point, we solve the problem Eq. (1) at various scale values s and use OnE algorithm (Algorithm 1 - Section 2.2) to tackle the optimization.

Figure 4 presents (i) the quantization loss per bit, (ii) the percentage of total retained variance by the minimizing quantization loss projection matrix in comparison with the total retained variance of top-L PCA components as s varies, and (iii)

---

[4]Due to very long training time at high-dimension of KMH (He et al. (2013)), we apply PCA to reduce dimension from 4096-D to 512-D. Additionally, we execute experiments for KMH with $b = \{2, 4, 8\}$ and report the best results.

Table 3: Summary of the percentage of retained variance (%), quantization loss per bit, and retrieval performance (*mAP*) on validation sets for ITQ and our SCQ-OnE methods (at the proposed scale of Eq. (25)).

	Method	CIFAR-10	LabelMe	SUN397
% Retained	ITQ	100%	100%	100%
variance	SCQ-OnE	59.6%	63.0%	69.4%
Quantization	ITQ	0.75	0.71	0.65
error	SCQ-OnE	0.29	0.29	0.24
mAP	ITQ	27.01	18.24	37.79
	SCQ-OnE	**30.68**	**23.74**	**41.12**

Table 4: Performance comparison in *mAP* and *prec@r2* with Deep Hashing (DH) (Erin Liong et al. (2015)) and Unsupervised Hashing with Binary Deep Neural Network (UH-BDNN) (Do et al. (2016a)) on CIFAR-10 dataset for $L = 16$ and 32. The **Bold** values indicate the **best** performances.

	Methods	*mAP*		*prec@r2*	
		16	32	16	32
CIFAR-10	DH	16.17	16.62	23.33	15.77
	UH-BDNN	17.83	18.52	**24.97**	18.85
	SCQ - OnE	17.97	18.63	24.57	23.72
	SCQ - OgE	**18.00**	**18.78**	24.15	**25.69**

the retrieval performance (*mAP*) of the validation sets. Firstly, we can observe that there is no scale *s* that can simultaneously maximizes the retained variance and minimizes the optimal quantization loss. On the one hand, as the scale value *s* decreases, minimizing the loss function Eq. (17) produces a projection matrix that focuses on high-variance directions, i.e., retains more variance in comparison with PCA (red line). On the other hand, at smaller *s*, the quantization loss is much larger (blue dash line). The empirical results are consistent with our discussion in section 2.3.

Secondly, regarding the retrieval performance, unsurprisingly, the performance drops as the scale *s* gets too small, i.e., a high amount of variance is retained but the quantization loss is too large, or *s* gets too large, i.e., the quantization loss is small but only low variance components are retained. Hence, it is necessary to balance these two factors. As data variance varies from dataset to dataset, the scale value should be determined from the dataset. In particular, we leverage the eigenvalues Λ, which are the variances of PCA components, to determine this hyper-parameter. From experimental results in Figure 4, we propose to formulate the scale parameter as:

$$s = \sqrt{\frac{L}{\sum_{i=1}^{L} \lambda_i}}, \tag{25}$$

One advantage of this setting is that it can generally achieve the best performances across multiple datasets, feature types, and hash lengths, without resort to conducting multiple trainings and cross-validations. The proposed working points of the scale are shown in Figure 4. We apply this scale parameter to the datasets for both OnE and OgE algorithms in all later experiments.

Note that the numerator of the fraction in Eq. 25, i.e., *L* is the hash code length, which is also the total variance of binary codes **B**. In addition, the denominator is the total variance of top *L*-th PCA components, i.e., the maximum amount of variance that can be retained in an *L*-dimension feature space. Hence, we can interpret the scale as a factor that make the amounts of variance, i.e., energy, of the input **X** and output (i.e. binary codes **B**) are comparable. This property is important as when the variance of input is much larger than the variance of output, obviously there is some information loss. On the other hand, when the variance of output is larger than it of input, the output contains undesirable additional information.

Additionally, in Table 3, we summarize the percentage of retained variance (%), quantization loss per bit, and retrieval

performance (*mAP*) on validation sets for ITQ and our SCQ-OnE methods. Even though, the projection matrix, learned by our Algorithm 1, can retain less variance in comparison to the optimal PCA projection matrix (i.e., the ITQ first step), this helps to achieve a much smaller quantization error. Hence, balancing the variance loss and quantization error is desirable and can result in higher retrieval performance.

4.3. Comparison with state-of-the-art

In this section, we evaluate our proposed hashing methods, SCQ - OnE and OgE, and compare to the state-of-the-art unsupervised hashing methods including SpH, KMH, BA, and ITQ. The experimental results in *mAP*, *prec@r2* and *prec@1k* are reported in Table 2. Our proposed methods clearly achieve significant improvement over all datasets at the majority of evaluation metrics. The improvement gaps are clearer at higher code lengths, i.e., $L = 32$. Additionally, OgE generally achieves slightly higher performance than OnE. Moreover, it is noticeable that, for *prec@r2*, all compared methods suffer performance downgrade at long hash code, e.g., $L = 32$. However, our proposed methods still achieve good *prec@r2* at $L = 32$. This shows that binary codes producing by our methods highly preserve data similarity.

Comparison with Deep Hashing (DH) (Erin Liong et al. (2015)) and Unsupervised Hashing with Binary Deep Neural Network (UH-BDNN) (Do et al. (2016a)). Recently, there are several methods (Erin Liong et al. (2015); Do et al. (2016a)) applying DNN to learn binary hash codes. These method can achieve very competitive performances. Hence, in order to have a complete evaluation, following the experiment settings of Erin Liong et al. (2015); Do et al. (2016a), we conduct experiments on the CIFAR-10 dataset. In this experiment, 100 images are randomly sampled for each class as a query set; the remaining images are for training and database. Each image is presented by a GIST 512-D descriptor (Oliva and Torralba (2001)). In addition, to avoid bias results due to test samples, we repeat the experiment 5 times with 5 different random training/query sets. The comparative results in term of *mAP* and *prec@r2* are presented in Table 4. Our proposed methods are very competitive with DH and UH-BDNN, specifically achieving higher *mAP* and *prec@r2* at $L = 32$ than DH and UH-BDNN.

Comparison with Binary Generative Adversarial Networks for Image Retrieval (BGAN) (Song (2018)). Recently, BGAN applies a continuous approximation of *sign* function to learn the binary codes which can help to generate images

Table 5: Performance comparison in *mAP* with BGAN (Song (2018)) on CIFAR-10 and NUS-WIDE datasets.

	Methods	CIFAR-10				NUS-WIDE			
		12	24	32	48	12	24	32	48
mAP	BGAN	40.1	51.2	53.1	55.8	67.5	69.0	71.4	72.8
	SCQ - OnE	53.59	**55.77**	57.62	58.14	69.82	70.53	**72.78**	**73.25**
	SCQ - OgE	**53.83**	55.65	**57.74**	**58.44**	**70.17**	**71.31**	72.49	72.95

plausibly similar to the original images. The method has been proven to achieve outstanding performances in unsupervised image hashing task. It is important to note that BGAN is different from our method and compared methods in the aspect that BGAN jointly learns image feature representations and binary codes, in which the binary codes are achieved by using an approximate smooth function of *sign*. While ours and compared methods learn the optimal binary codes given image representations. Hence, to further validate the effectiveness of our methods and to compare with BGAN, we apply our method on the FC7 features extracted from the feature extraction component in the pre-trained BGAN model[5] on CIFAR-10 and NUS-WIDE (Chua et al.) datasets. In this experiment, we aim to show that by applying our hashing methods on the pretrained features from feature extraction component of BGAN, our methods can produce better hash codes than the hash codes which are obtained from the jointly learning approach of BGAN.

Similar to the experiment setting in BGAN (Song (2018)), for both CIFAR-10 and NUS-WIDE, we randomly select 100 images per class as the test query set; the remaining images are used as database for retrieval. We then randomly sample from the database set 1,000 images per class as the training set. The Table 5 shows that by using the more discriminative features[6] from the pre-trained feature extraction component of BGAN, our methods can outperform BGAN, i.e., our methods can produce better binary codes in comparison to the *sign* approximate function in BGAN, and achieve the state-of-the-art performances in the unsupervised image hashing task. Hence, the experiment results emphasize the important of an effective method to preserve the discrimination power of high-dimensional CNN features in very compact binary representations, i.e., effectively handling the challenging binary and orthogonal constraints.

4.4. Training time and Processing time

In this experiment, we empirically evaluate the training time and online processing time of our methods. The experiments are carried out on a workstation with a 4-core i7-6700 CPU @ 3.40GHz. The experiments are conducted on the combination of CIFAR-10, Labelme-12-50k, and SUN397 datasets. For OnE and OgE, the training time include time for applying zero-mean, scaling, reducing dimension to $D = 512$. We use 50 iterations for all experiments. The Fig. 5 shows that our
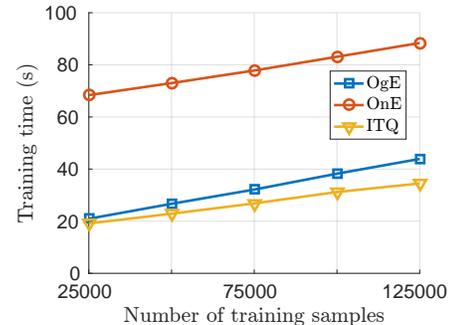
Figure 5: The training time for learning 32-bit hash code embedding.

proposed methods, OnE and OgE, are very efficient. OgE is just slightly slower than ITQ. Even though OnE is slower than OgE and ITQ, it takes just over a minute for 100.000 training samples which is still very fast and practical, in comparison with several dozen minutes for KMH, BA, and UH-BDNN[7].

Compared with training cost, the time to produce new hash codes is more important since it is done in real time. Similar to Semi-Supervised Hashing (SSH) (Wang et al. (2012)) and ITQ (Gong and Lazebnik (2011)), by using only a single linear transformation, our proposed methods require only one BLAS operation (`gemv` or `gemm`) and a comparison operation; hence, it takes negligible time to produce binary codes for new data points.

5. Conclusion

In this paper, we successfully addressed the problem of jointly learning to preserve data pairwise (dis)similarity in low-dimension space and to minimize the binary quantization loss with the strict diagonal constraint. Additionally, we show that as more variance is retained, the quantization loss is undesirably larger; and vice versa. Hence, by appropriately balancing these two factors using a scale, our methods can produce better binary codes. Extensive experiments on various datasets show that our proposed methods, Simultaneous Compression and Quantization (SCQ): Orthonormal Encoder (OnE) and Orthogonal Encoder (OgE), outperform other state-of-the-art hashing methods by clear margins under various standard evaluation metrics and benchmark datasets. Furthermore, OnE and OgE are very computationally efficient in both training and testing steps.

---

[5]The model is obtained after training BGAN method on CIFAR-10 and NUS-WIDE datasets accordingly. The same model is also used to obtain BGAN binary codes.

[6]In comparison with the image features which are obtained from the pre-trained off-the-shelf VGG network (Simonyan and Zisserman (2014)).

[7]For training 50000 CIFAR-10 samples using author's release code and dataset (Do et al. (2016a)).

References

Andoni, A., Indyk, P., 2008. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51.

Boyd, S., Vandenberghe, L., 2004. Convex Optimization. Cambridge University Press, New York, NY, USA.

Cao, Y., Long, M., Liu, B., Wang, J., 2018. Deep cauchy hashing for hamming space retrieval, in: CVPR.

Cao, Z., Long, M., Wang, J., Yu, P.S., 2017. Hashnet: Deep learning to hash by continuation, in: ICCV, pp. 5609–5618.

Carreira-Perpiñán, M.Á., Raziperchikolaei, R., 2015. Hashing with binary autoencoders, in: CVPR.

Chen, Z., Yuan, X., Lu, J., Tian, Q., Zhou, J., 2018. Deep hashing via discrepancy minimization, in: CVPR.

Chen, Z., Yuan, X., Lu, J., Tian, Q., Zhou, J., 2018. Deep hashing via discrepancy minimization, in: CVPR, pp. 6838–6847.

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

Do, T., Hoang, T.N.A., Le, K., Doan, D., Cheung, N., 2019. Compact hash code learning with binary deep neural network. IEEE Transactions on Multimedia .

Do, T.T., Doan, A.D., Cheung, N.M., 2016a. Learning to hash with binary deep neural network, in: ECCV.

Do, T.T., Doan, A.D., Nguyen, D.T., Cheung, N.M., 2016b. Binary hashing with semidefinite relaxation and augmented lagrangian, in: ECCV.

Do, T.T., Le, K., Hoang, T., Le, H., Nguyen, T.V., Cheung, N.M., 2019. Simultaneous feature aggregating and hashing for compact binary code learning. IEEE TIP .

Do, T.T., Tan, D.K.L., Pham, T., Cheung, N.M., 2017. Simultaneous feature aggregating and hashing for large-scale image search, in: CVPR.

y. Duan, L., Wu, Y., Huang, Y., Wang, Z., Yuan, J., Gao, W., 2018. Minimizing reconstruction bias hashing via joint projection learning and quantization. IEEE TIP 27, 3127–3141.

Duan, Y., Lu, J., Wang, Z., Feng, J., Zhou, J., 2017. Learning Deep Binary Descriptor with Multi-quantization, in: CVPR, pp. 4857–4866.

En, S., Crémilleux, B., Jurie, F., 2017. Unsupervised deep hashing with stacked convolutional autoencoders, in: ICIP, pp. 3420–3424.

Erin Liong, V., Lu, J., Wang, G., Moulin, P., Zhou, J., 2015. Deep hashing for compact binary codes learning, in: CVPR.

Gionis, A., Indyk, P., Motwani, R., 1999. Similarity search in high dimensions via hashing, in: VLDB.

Gong, Y., Lazebnik, S., 2011. Iterative quantization: A procrustean approach to learning binary codes, in: CVPR.

Grauman, K., Fergus, R., 2013. Learning binary hash codes for large-scale image search, in: Studies in Computational Intelligence.

Gurbuzbalaban, M., Ozdaglar, A., Parrilo, P.A., Vanli, N., 2017. When cyclic coordinate descent outperforms randomized coordinate descent, in: NIPS.

He, K., Wen, F., Sun, J., 2013. K-means hashing: An affinity-preserving quantization method for learning binary compact codes, in: CVPR.

Heo, J.P., Lee, Y., He, J., Chang, S.F., Yoon, S.E., 2012. Spherical hashing, in: CVPR.

Hu, M., Yang, Y., Shen, F., Xie, N., Shen, H.T., 2018. Hashing with angular reconstructive embeddings. IEEE TIP 27, 545–555.

Huang, Y., Lin, Z., 2018. Binary multidimensional scaling for hashing. IEEE TIP 27, 406–418.

Jain, H., Zepeda, J., Pérez, P., Gribonval, R., 2017. SuBiC: A Supervised, Structured Binary Code for Image Search, in: ICCV, pp. 833–842.

Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images, in: Technical report, University of Toronto.

Kulis, B., Darrell, T., 2009. Learning to hash with binary reconstructive embeddings, in: NIPS.

Kulis, B., Grauman, K., 2009. Kernelized locality-sensitive hashing for scalable image search, in: ICCV.

Lai, H., Pan, Y., Ye Liu, Yan, S., 2015. Simultaneous feature learning and hash coding with deep neural networks, in: CVPR, pp. 3270–3278.

Lin, G., Shen, C., Shi, Q., van den Hengel, A., Suter, D., 2014. Fast supervised hashing with decision trees for high-dimensional data, in: CVPR.

Lin, K., Lu, J., Chen, C.S., Zhou, J., 2016. Learning compact binary descriptors with unsupervised deep neural networks, in: CVPR.

Lin, K., Yang, H., Hsiao, J., Chen, C., 2015. Deep learning of binary hash codes for fast image retrieval, in: CVPR Workshop, pp. 27–35.

Liny, K., Luz, J., Cheny, C.S., Zhou, J., 2016. Learning compact binary descriptors with unsupervised deep neural networks, in: CVPR.

Liu, H., Wang, R., Shan, S., Chen, X., 2016. Deep Supervised Hashing for Fast Image Retrieval, in: CVPR, pp. 2064–2072.

Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F., 2012. Supervised hashing with kernels, in: CVPR.

Norouzi, M., Fleet, D.J., Salakhutdinov, R., 2012. Hamming distance metric learning, in: NIPS.

Oliva, A., Torralba, A., 2001. Modeling the shape of the scene: A holistic representation of the spatial envelope. IJCV , 145–175.

Raginsky, M., Lazebnik, S., 2009. Locality-sensitive binary codes from shift-invariant kernels, in: NIPS.

Russell, B.C., Torralba, A., Murphy, K.P., Freeman, W.T., 2008. Labelme: A database and web-based tool for image annotation. IJCV , 157–173.

Schönemann, P.H., 1966. A generalized solution of the orthogonal procrustes problem. Psychometrika .

Shen, F., Shen, C., Liu, W., Shen, H.T., 2015. Supervised discrete hashing, in: CVPR.

Shen, F., Xu, Y., Liu, L., Yang, Y., Huang, Z., Shen, H.T., 2018. Unsupervised deep hashing with similarity-adaptive and discrete optimization. IEEE TPAMI , 1–1.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. CoRR .

Song, J., 2018. Binary generative adversarial networks for image retrieval, in: AAAI.

Uetz, R., Behnke, S., 2009. Large-scale object recognition with cuda-accelerated hierarchical neural networks, in: IEEE International Conference on Intelligent Computing and Intelligent Systems (ICIS).

Wang, J., Kumar, S., Chang, S.F., 2012. Semi-supervised hashing for large-scale search. TPAMI .

Wang, J., Liu, W., Kumar, S., Chang, S., 2015. Learning to hash for indexing big data - a survey, in: Proceedings of the IEEE.

Wang, J., Tao Shen, H., Song, J., Ji, J., 2014. Hashing for similarity search: A survey .

Wang, J., Zhang, T., j. song, Sebe, N., Shen, H.T., 2017. A survey on learning to hash. TPAMI .

Wang, M., Zhou, W., Tian, Q., Li, H., 2018. A general framework for linear distance preserving hashing. IEEE TIP 27, 907–922.

Weiss, Y., Torralba, A., Fergus, R., 2009. Spectral hashing, in: NIPS.

Wen, Z., Yin, W., 2013. A feasible method for optimization with orthogonality constraints. Math. Program. .

Xiao, J., Ehinger, K.A., Hays, J., Torralba, A., Oliva, A., 2016. Sun database: Exploring a large collection of scene categories. IJCV .

Yuan, G., Ghanem, B., 2017. An exact penalty method for binary optimization based on mpec formulation, in: AAAI.

Zhang, D., Wang, J., Cai, D., Lu, J., 2010. Self-taught hashing for fast similarity search, in: ACM SIGIR.

Appendices

A. Derivation for Eq. (8)

Firstly, the dual function $\mathcal{G}_1(v_1)$ can be simply constructed by substituting $\mathbf{V}_1$ from Eq. (7) into Eq. (6):

$$\mathcal{G}(v_1) = \frac{1}{n} \left\| \mathbf{b}_1 - \mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 \right\|^2 + v_1 \left(\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 - 1 \right) \tag{26}$$

Firstly, we note that:

$$\frac{\partial(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}}{\partial v_1}$$
$$= -(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\frac{\partial(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})}{\partial v_1}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1} \tag{27}$$
$$= -n(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}$$

Hence,

$$\frac{\partial\mathcal{G}(v_1)}{\partial v_1}$$
$$= -\frac{2}{n}\left(\mathbf{b}_1 - \mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1\right)^\top \mathbf{X}\frac{\partial(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}}{\partial v_1}\mathbf{X}^\top\mathbf{b}_1$$
$$+ \left(\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 - 1\right)$$
$$+ 2v_1\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}\frac{\partial(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}}{\partial v_1}\mathbf{X}^\top\mathbf{b}_1$$
$$= 2\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1$$
$$- 2\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}\mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1$$
$$+ \left(\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 - 1\right)$$
$$- 2nv_1\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1$$
$$= 2\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1$$
$$- 2\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1$$
$$+ 2nv_1\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1$$
$$+ \left(\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 - 1\right)$$
$$- 2nv_1\mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1$$
$$= \mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 - 1 \tag{28}$$

$$\Leftrightarrow \frac{\partial\mathcal{G}_1}{\partial v_1} = \mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + nv_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 - 1 \tag{29}$$

Note that: $(\mathbf{X}^\top\mathbf{X} + n\nu_1\mathbf{I})^{-1} = (\mathbf{X}^\top\mathbf{X} + n\nu_1\mathbf{I})^{-\top}$.

A simpler way to achieve $\frac{\partial \mathcal{G}(\nu_1)}{\partial \nu_1}$ is to take the deravative of $\mathcal{L}$ w.r.t ν_1 first, then replace $\mathbf{v}_1$ by Eq. (7) later.

$$\frac{\partial \mathcal{L}}{\partial \nu_1} = \mathbf{v}_1^\top\mathbf{v}_1 - 1 \tag{30}$$

$$\Rightarrow \frac{\partial \mathcal{G}}{\partial \nu_1} = \mathbf{b}_1^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X} + n\nu_1\mathbf{I})^{-\top}(\mathbf{X}^\top\mathbf{X} + n\nu_1\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{b}_1 - 1 \tag{31}$$

By setting $\frac{\partial \mathcal{G}_1}{\partial \nu_1} = 0$ (Eq. (31)), we can obtain the second condition in Eq. (8).

B. Derivation for Eq. (13)

Following the similar derivation in Appendix section A, we can obtain the second condition of Eq. (13). We now provide the detail derivation for the third condition. Considering the i-th value (ϕ_{ki}) of the Lagrange multiplier Φ_k

$$\frac{\partial \mathcal{L}_k}{\partial \phi_{ki}} = \mathbf{v}_k^\top\mathbf{v}_i \tag{32}$$

$$\Rightarrow \frac{\partial \mathcal{G}_k}{\partial \phi_{ki}} = \left(\mathbf{X}^\top\mathbf{b}_k - \frac{n}{2}\sum_{j=1}^{k-1}\phi_{kj}\mathbf{v}_j\right)^\top (\mathbf{X}^\top\mathbf{X} + n\nu_k\mathbf{I})^{-\top}\mathbf{v}_i$$
$$= -\frac{n}{2}\phi_{k1}\mathbf{v}_1^\top\mathbf{Z}_k\mathbf{v}_i - \cdots - \frac{n}{2}\phi_{k(k-1)}\mathbf{v}_{(k-1)}^\top\mathbf{Z}_k\mathbf{v}_1 + \mathbf{b}_k^\top\mathbf{X}\mathbf{Z}_k\mathbf{v}_i \tag{33}$$

where $\mathbf{Z}_k = (\mathbf{X}^\top\mathbf{X} + n\nu_k\mathbf{I})^{-1}$.

By setting the derivative of $\mathcal{G}_k$ w.r.t $\Phi_k = [\phi_{k1}, ..., \phi_{k(k-1)}]^\top$ equal to $[0, \cdots, 0]$ and some simple manipulations, we can obtain the third condition of Eq. (13) as follows:

$$\mathbf{A}_k\Phi_k = \mathbf{c}_k, \tag{34}$$

where

$$\begin{cases} \mathbf{A}_k = \frac{n}{2}\begin{bmatrix} \mathbf{v}_1^\top\mathbf{Z}_k\mathbf{v}_1 & \cdots & \mathbf{v}_1^\top\mathbf{Z}_k\mathbf{v}_{(k-1)} \\ \vdots & \ddots & \vdots \\ \mathbf{v}_{(k-1)}^\top\mathbf{Z}_k\mathbf{v}_1 & \cdots & \mathbf{v}_{(k-1)}^\top\mathbf{Z}_k\mathbf{v}_{(k-1)} \end{bmatrix} \\ \mathbf{c}_k = \begin{bmatrix} \mathbf{v}_1^\top\mathbf{Z}_k\mathbf{X}^\top\mathbf{b}_k & \cdots & \mathbf{v}_{(k-1)}^\top\mathbf{Z}_k\mathbf{X}^\top\mathbf{b}_k \end{bmatrix}^\top \end{cases} \tag{35}$$