Binary Multi-View Clustering

Zheng Zhang†, Li Liu†, Fumin Shen, Heng Tao Shen, Ling Shao*

Abstract—Clustering is a long-standing important research problem, however, remains challenging when handling large-scale image data from diverse sources. In this paper, we present a novel Binary Multi-View Clustering (BMVC) framework, which can dexterously manipulate multi-view image data and easily scale to large data. To achieve this goal, we formulate BMVC by two key components: compact collaborative discrete representation learning and binary clustering structure learning, in a joint learning framework. Specifically, BMVC collaboratively encodes the multi-view image descriptors into a compact common binary code space by considering their complementary information; the collaborative binary representations are meanwhile clustered by a binary matrix factorization model, such that the cluster structures are optimized in the Hamming space by pure, extremely fast bit-operations. For efficiency, the code balance constraints are imposed on both binary data representations and cluster centroids. Finally, the resulting optimization problem is solved by an alternating optimization scheme with guaranteed fast convergence. Extensive experiments on four large-scale multi-view image datasets demonstrate that the proposed method enjoys the significant reduction in both computation and memory footprint, while observing superior (in most cases) or very competitive performance, in comparison with state-of-the-art clustering methods.

Index Terms—Large-scale clustering, multi-view data, efficient, short binary code, discrete representation

1 Introduction

Data clustering has been a fundamental research topic in the computer vision and data mining communities [1]–[4]. The goal of clustering is to categorize data items with similar structures or patterns into the same group for reducing data complexity and facilitating interpretation. With the unprecedentedly explosive growth in the volume of visual data, how to effectively cluster large-scale image data becomes an interesting but challenging problem. Moreover, it is common in many real-world applications that data are collected from diverse sources or represented by heterogeneous features from different views [5]–[7]. However, existing single- and multi-view clustering methods fail to efficiently cluster large-scale multi-view data due to the high computational complexity and massive storage costs [5].

Much attention is paid to the k-means based clustering [1], [4] because of its simplicity and mathematical tractability [12]–[17]. However, the severely unaffordable computational time and storage requirement preclude k-means from real-world applications with large data and a big number of clusters. Accordingly, many optimized k-means methods [14]–[18] were proposed to reduce the iteration numbers or to circumvent unnecessary distance calculations. But these methods are incapable to completely overcome the limitations of the high complexity and cumbersome memory load.

Spectral clustering (SC) [2], [8] is another most successful single-view clustering method. Unfortunately, SC cannot be directly applied to large-scale datasets either, due to the expensive adjacency matrix and eigenspace constructions [9]. Generally, the complexities of these methods are $\mathcal{O}(n^2d+n^3)$ in time and $\mathcal{O}(n^2+nt)$ in space, where n, d and t are the number of data instances, the dimensionality, and neighbors for the adjacency matrix, respectively. Particularly, the adjacency matrix for 500,000 data points will consume about 2 TB of memory. The high computation burden is another main bottleneck that makes it unrealistic for large-scale data. Although there are some improved SC methods [9]–[11], they still cannot intrinsically tackle these deficiencies.

Recently, multi-view clustering by exploiting heterogeneous features of data has attracted considerable attention [5], [6], [20]. Compared to single-view clustering, multi-view clustering normally can access to more characteristics and structural information hidden in the data, and intuitively can exploit richer properties of data to improve the clustering performance. [1] Conventional methods either directly concatenate these features as a long representation or treat each view independently for multi-view clustering [5]. However, neither of the strategies is physically meaningful since they fail to exploit the compatible and complementary information of data from multiple distinct representations. Many multi-view clustering methods have been proposed [7], [13], [19], which can be roughly divided into three groups: the first category projects features from

This work is partially supported by the Nature Science Foundation of China (61502081, 61702117, 61632007), Science and Technology Program of Guangzhou (201804010355). (Corresponding author: L. Shao (ling.shao@ieee.org))

- *Z. Zhang is with is with School of Information Technology & Electrical Engineering, The University of Queensland, Brisbane, QLD 4072, Australia.*
  E-mail: darrenzz219@gmail.com
- *L. Liu and L. Shao are with the Inception Institute of Artificial Intelligence, Abu Dhabi, UAE.*
  E-mail: liuli1213@gmail.com, ling.shao@ieee.org
- *F. Shen and H. T. Shen are with Center for Future Media and School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.*
  E-mail: fumin.shen@gmail.com, shenhengtao@hotmail.com

† *indicates equal first author.*

1. Different to single-view clustering using singular data descriptor, in this paper, we first describe each data point (e.g., an image) by various features (e.g., different image descriptors, such as HOG, Color Histogram and GIST) and then feed these features from multiple descriptors into our clustering. It is noteworthy that the "Multiview" in our paper indicates multiple image descriptors of features rather than multiple modalities.

different views into a common low-dimensional feature subspace such as using CCA [6] to minimize the cross-correlation error, and then employs an existing clustering method (*e.g.,* k-means) to group data; the second category is the multi-view spectral clustering methods [11], [19]–[22], which construct multiple graphs to characterize the geometrical structure and employ an existing clustering method for data partition; the third category is based on matrix factorization [13], [23]–[25], which is equivalent to the relaxed k-means by decomposing the feature matrix into a centroid matrix and the cluster assignment. Instead of using spectral clustering, recent research on online multi-view clustering [26] provides a more feasible solution to solve large-scale multi-view clustering with incomplete views. Nonetheless, all these methods have not perfectly eliminated the primary concerns of efficient large-scale multi-view clustering, i.e. high computational costs and demanding memory overhead.

Fortunately, recent advancement of binary code learning [27]–[29] provides a preferable solution to overcome the above challenges due to its efficiency in computation and space. The key idea of binary code learning [27]–[31] is to encode the original feature descriptors by a set of short binary codes in a similarity-preserving low-dimensional Hamming space. It results in two prominent characteristics, i.e. exclusively fast hamming distance computation by build-in XOR operations and much less storage requirement compared to the original real-valued features. Binary coding has shown its powerful data-handling capability for various large-scale visual applications especially in visual retrieval [32]–[35] and object recognition [36]–[38]. Recently, learning hashing codes on multi-view data has attracted much research enthusiasm owing to the preferable retrieval performance in comparison with single-view data [39]–[42]. However, *binary code learning has been rarely studied for the large-scale multi-view clustering problem*.

In this paper, we introduce a novel framework for large-scale multi-view clustering, dubbed *binary multi-view clustering (BMVC)*, which can greatly reduce the computational complexity as well as the memory requirement. Different from the existing real-valued clustering methods, BMVC concurrently learns the collaborative binary representation of multi-view data, and at the same time optimizes binary clustering to ensure the consistent results among different views. For coding efficiency, the code balance constraints are explicitly enforced on the binary codes and cluster centroids. The contributions of this work mainly include:

1) The proposed BMVC simultaneously learns the collaborative binary codes for data from multiple views and binary cluster structures in a joint framework. To the best of our knowledge, BMVC is the very first effort of addressing the large-scale multi-view clustering problem by the binary coding technique.
2) An alternating optimization algorithm with rigorous convergence proof is developed to effectively tackle the resulting discrete programming problem. To solve the key subproblem of binary clustering centroids learning, we propose an adaptive discrete proximal linear method (ADPLM), which can directly handle the binary variables during the optimization.
3) By virtue of the proposed effective optimization algorithm, BMVC shows clearly superior multi-view clustering performance in comparison with state-of-the-art methods, including the real-valued ones. More importantly, BMVC requires significantly less computational time and memory overhead, which are critical for realistic large-scale applications with limited computing and memory resources.

Related Work Only few studies have been devoted to the large-scale binary data clustering. Gong et al. [43] developed a binary single-view clustering method composed of two separate stages, i.e. generating binary codes and binary k-means clustering. Its major drawbacks are that the binary codes are obtained by using a data-independent method, and such a two-step clustering destroys the relationship between binary codes and the segmentation of data. Shen et al. [12] integrated binary structured SVM and k-means into one optimization model to accelerate large-scale single-view clustering. Both methods are not applicable to large-scale multi-view clustering, and the beneficial multi-view traits of data are not well explored. Moreover, the inferior single-view clustering performance of these methods also limits the use of the designed systems. The conference version of our method has been reviewed in [44].

2 Binary Multi-View Clustering

2.1 The proposed BMVC Framework

In this section, we elaborate our scalable binary multi-view clustering (**BMVC**) approach, which establishes a general framework of large-scale multi-view discrete clustering. BMVC incorporates two key components: *collaborative discrete representation learning (**CDRL**)* and *binary clustering structure learning (**BCSL**)*, into a unified learning framework.

Suppose a multi-view dataset is composed of M representations (i.e. views) for n instances, which are denoted by a set of matrices $\mathcal{X} = [\boldsymbol{X}^1, \cdots, \boldsymbol{X}^M]$, where $\boldsymbol{X}^v \in \Re^{d_v \times n}$ is the data matrix of the v-th view, and d_v is the dimensionality of data features from the v-th view. We assume that points in each view are zero-centered, i.e. $\sum_s \boldsymbol{X}_s^v = \boldsymbol{0}$. As the preprocessing step, we first encode data by the simple nonlinear RBF mapping

$$\phi(\boldsymbol{x}_s^v) = [exp(-\|\boldsymbol{x}_s^v - \boldsymbol{a}_1^v\|^2/\sigma), \cdots, exp(-\|\boldsymbol{x}_s^v - \boldsymbol{a}_m^v\|^2/\sigma)]^T, \quad (1)$$

where σ is the kernel width, $\phi(\boldsymbol{x}_s^v) \in \Re^m$ indicates an m-dimensional nonlinear embedding for the s-th sample from the v-th view $\boldsymbol{x}_s^v \in \Re^{d_v}$, and $\{\boldsymbol{a}_i^v\}_{i=1}^m$ are the randomly selected m anchor samples from the v-th view.

CDRL Loss: CDRL aims to project features from different views into a common Hamming space, $\Re \rightarrow \{-1, 1\}^{l \times n}$. To fulfill this, we define the binary hash function for $\boldsymbol{x}_s^v$ as

$$\boldsymbol{h}_s^v(\phi(\boldsymbol{x}_s^v); \boldsymbol{U}^v) = sgn(\boldsymbol{U}^v \phi(\boldsymbol{x}_s^v)), \quad (2)$$

where $sgn(\cdot)$ is an element-wise sign operator, and $\boldsymbol{U}^v \in \Re^{l \times m}$ is the mapping matrix for the v-th view. To concurrently consider the multi-view correlations and complementary nature of multiple views, the objective of CDRL writes

$$\begin{aligned} \min_{\boldsymbol{U}^v, \boldsymbol{b}_s, \boldsymbol{\alpha}} & \sum_{v=1}^{M} (\boldsymbol{\alpha}^v)^r \Big(\sum_{s=1}^{n} \|\boldsymbol{b}_s - \boldsymbol{h}_s^v\|_F^2 + \beta \|\boldsymbol{U}^v\|_F^2 - \gamma \sum_{s=1}^{n} var(\boldsymbol{h}_s^v) \Big) \\ s.t. & \sum_v \boldsymbol{\alpha}^v = 1, \boldsymbol{\alpha}^v > 0, \boldsymbol{b}_s \in \{-1, 1\}^{l \times 1}. \end{aligned} \quad (3)$$

Here $\boldsymbol{b}_s$ is the collaborative binary code for the s-th instance, $\boldsymbol{\alpha}$=$[\boldsymbol{\alpha}^1, \cdots, \boldsymbol{\alpha}^M] \in \Re^M$ is a nonnegative normalized weighting vector to balance the significance of different views and $r > 1$ is a scalar controlling the weights; γ is a nonnegative constant. Specifically, the first term ensures to learn a unified binary code for different views; the last two terms contribute to the stable solution and balanced bits, respectively. Particularly, the last term maximizes the variance of the encoding function producing balanced bits, which is a typical requirement of binary code learning [33].

BCSL Loss: Besides the collaborative multi-view representation learning, we also consider keeping the consistent cluster structures of different views. Due to the equivalence of the conventional k-means clustering and matrix factorization [45], we construct the BCSL scheme by

$$\begin{aligned} \min_{\boldsymbol{C},\boldsymbol{g}_s} \ & \|\boldsymbol{b}_s - \boldsymbol{C}\boldsymbol{g}_s\|_F^2 \ \ s.t. \ \ \boldsymbol{C}^T\mathbf{1} = \mathbf{0}, \boldsymbol{C} \in \{-1,1\}^{l\times c}, \\ & \boldsymbol{g}_s \in \{0,1\}^c, \sum_i \boldsymbol{g}_{is} = 1, \end{aligned} \tag{4}$$

where $\boldsymbol{C}$ and $\boldsymbol{g}_s$ are the clustering centroids and indicator vector, respectively. To produce efficient code and maximize the information of each bit, we impose the balanced constraint (the first constraint) on the clustering centroids to make them adapt to binary clustering task.

Overall Objective Function: The collaborative discrete representation learning and binary clustering structure learning are both indispensable to scalable multi-view clustering. By relaxing the sign function in (2) with its signed magnitude, we formulate BMVC as

$$\begin{aligned} \min \mathcal{F}(\boldsymbol{U}^v, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{G}, \boldsymbol{\alpha}) = & \\ \sum_{v=1}^{M} (\boldsymbol{\alpha}^v)^r \Big(& \|\boldsymbol{B} - \boldsymbol{U}^v\phi(\boldsymbol{x}^v)\|_F^2 + \beta\|\boldsymbol{U}^v\|_F^2 \\ & - \frac{\gamma}{n} tr((\boldsymbol{U}^v\phi(\boldsymbol{x}^v))(\boldsymbol{U}^v\phi(\boldsymbol{x}^v))^T\Big) + \lambda\|\boldsymbol{B} - \boldsymbol{C}\boldsymbol{G}\|_F^2 \\ s.t. \ \ \boldsymbol{C}^T\mathbf{1} = \mathbf{0}, & \sum_v \boldsymbol{\alpha}^v = 1, \boldsymbol{\alpha}^v > 0, \boldsymbol{B} \in \{-1,1\}^{l\times n}, \\ & \boldsymbol{C} \in \{-1,1\}^{l\times c}, \ \boldsymbol{G} \in \{0,1\}^{c\times n}, \sum_i \boldsymbol{g}_{is} = 1, \end{aligned} \tag{5}$$

where $\boldsymbol{B} = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_n]$, $\boldsymbol{G} = [\boldsymbol{g}_1, \cdots, \boldsymbol{g}_n]$ and λ is the regularization parameter. To solve this difficult discrete optimization problem, we propose an alternating optimization algorithm with guaranteed convergence, as shown in the next section.

2.2 Optimization Algorithm

The optimization problem (5) involves the discrete constraints which result in an NP-hard problem. In this section, we develop an alternating optimization strategy, which separates the problem into several subproblems such that each subproblem is tractable. That is, we alternatingly update each variable when fixing others.

Updating $\boldsymbol{U}^v$: By fixing all variables but $\boldsymbol{U}^v$, problem (5) reduces to

$$\begin{aligned} \min \mathcal{F}(\boldsymbol{U}^v) = & \|\boldsymbol{B} - \boldsymbol{U}^v\phi(\boldsymbol{x}^v)\|_F^2 + \beta\|\boldsymbol{U}^v\|_F^2 \\ & - \frac{\gamma}{n} tr\left((\boldsymbol{U}^v\phi(\boldsymbol{x}^v))(\boldsymbol{U}^v\phi(\boldsymbol{x}^v))^T\right), \end{aligned} \tag{6}$$

which has a closed-form solution obtained by setting the derivation $\frac{\partial\mathcal{F}(\boldsymbol{U}^v)}{\partial\boldsymbol{U}^v} = 0$, *i.e.*,

$$\boldsymbol{U}^v = \boldsymbol{B}\phi^T(\boldsymbol{x}^v)\boldsymbol{W}. \tag{7}$$

$\boldsymbol{W}$=$\left((1-\frac{\gamma}{n})\phi(\boldsymbol{x}^v)\phi^T(\boldsymbol{x}^v)+\beta\boldsymbol{I}\right)^{-1}$ can be calculated beforehand.

Updating $\boldsymbol{B}$: Similarly, problem (5) writes w.r.t. $\boldsymbol{B}$:

$$\begin{aligned} \min_{\boldsymbol{B}} & \sum_{v=1}^{M} (\boldsymbol{\alpha}^v)^r \left(\|\boldsymbol{B} - \boldsymbol{U}^v\phi(\boldsymbol{x}^v)\|_F^2\right) + \lambda\|\boldsymbol{B} - \boldsymbol{C}\boldsymbol{G}\|_F^2 \\ & = tr\left[\boldsymbol{B}^T\left(\sum_{v=1}^{M}(\boldsymbol{\alpha}^v)^r\boldsymbol{I} + \lambda\boldsymbol{I}\right)\boldsymbol{B}\right] \\ & \quad - 2tr\left[\boldsymbol{B}^T\left(\sum_{v=1}^{M}(\boldsymbol{\alpha}^v)^r\boldsymbol{U}^v\phi(\boldsymbol{x}^v) + \lambda\boldsymbol{C}\boldsymbol{G}\right)\right] + con \\ s.t. \ \ & \boldsymbol{B} \in \{-1,1\}^{l\times n}, \end{aligned} \tag{8}$$

where '*con*' means the constant value w.r.t. $\boldsymbol{B}$. Since $tr(\boldsymbol{B}\boldsymbol{B}^T) = tr(\boldsymbol{B}^T\boldsymbol{B}) = nl$ is a constant, problem (8) can be rewritten as

$$\begin{aligned} \min_{\boldsymbol{B}} & -2tr\left[\boldsymbol{B}^T\left(\sum_{v=1}^{M}(\boldsymbol{\alpha}^v)^r\boldsymbol{U}^v\phi(\boldsymbol{x}^v) + \lambda\boldsymbol{C}\boldsymbol{G}\right)\right] \\ s.t. \ \ & \boldsymbol{B} \in \{-1,1\}^{l\times n}, \end{aligned} \tag{9}$$

which has a closed-form optimal solution:

$$\boldsymbol{B} = sgn\left(\sum_{v=1}^{M}(\boldsymbol{\alpha}^v)^r\boldsymbol{U}^v\phi(\boldsymbol{x}^v) + \lambda\boldsymbol{C}\boldsymbol{G}\right). \tag{10}$$

Updating $\boldsymbol{C}$ and $\boldsymbol{G}$: In this part, we optimize the binary clustering structure in the Hamming space. By removing the irrelevant terms, we obtain the following problem:

$$\begin{aligned} & \min_{\boldsymbol{C},\boldsymbol{G}} \|\boldsymbol{B} - \boldsymbol{C}\boldsymbol{G}\|_F^2 \\ & s.t. \ \boldsymbol{C}^T\mathbf{1} = \mathbf{0}, \ \boldsymbol{C} \in \{-1,1\}^{l\times c}, \ \boldsymbol{G} \in \{0,1\}^{c\times n}, \sum_i \boldsymbol{g}_{is} = 1. \end{aligned} \tag{11}$$

We reformulate (11) to

$$\begin{aligned} & \min_{\boldsymbol{C},\boldsymbol{G}} \|\boldsymbol{B} - \boldsymbol{C}\boldsymbol{G}\|_F^2 + \rho\|\boldsymbol{C}^T\mathbf{1}\|^2 \\ & s.t. \ \boldsymbol{C} \in \{-1,1\}^{l\times c}, \ \boldsymbol{G} \in \{0,1\}^{c\times n}, \sum_i \boldsymbol{g}_{is} = 1, \end{aligned} \tag{12}$$

which is equivalent to (11) with arbitrarily large ρ. Similar as the conventional k-means method, we iteratively optimize the cluster centroids and indicators as follows.

C-Step: Optimizing $\boldsymbol{C}$ is difficult because of the discrete constraint. Inspired by the recently proposed discrete proximal linearized minimization (DPLM) [32], we devise an effective algorithm by keeping the discrete constraints during optimization, which is critical to obtain high-quality binary solutions [32], [35]. With $\boldsymbol{G}$ fixed, we have

$$\begin{aligned} \min \mathcal{F}(\boldsymbol{C}) &= \|\boldsymbol{B} - \boldsymbol{C}\boldsymbol{G}\|_F^2 + \rho\|\boldsymbol{C}^T\mathbf{1}\|^2 \\ &= -2tr(\boldsymbol{B}^T\boldsymbol{C}\boldsymbol{G}) + \rho\|\boldsymbol{C}^T\mathbf{1}\|^2 + con \\ s.t. \ \ & \boldsymbol{C} \in \{-1,1\}^{l\times c}. \end{aligned} \tag{13}$$

According to the rule of DPLM, we update $\boldsymbol{C}$ in the p+1-th iteration by

$$\boldsymbol{C}^{p+1} = sgn(\boldsymbol{C}^p - \frac{1}{\mu}\nabla\mathcal{F}(\boldsymbol{C}^p)), \tag{14}$$

where $\nabla\mathcal{F}(\boldsymbol{C})$ is the gradient of $\mathcal{F}(\boldsymbol{C})$.

It should be noted that DPLM adopts a fixed step-size $1/\mu$, which is not the best choice with the $sgn(\cdot)$ function, since an improper value of step-size may confine the algorithm to converge to a bad local minimum. To this

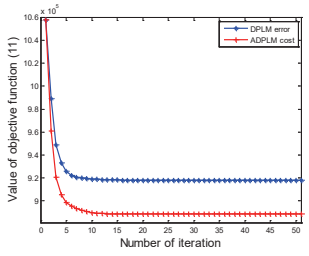


Fig. 1: Error comparison of DPLM and ADPLM w.r.t. the number of inner iteration for optimizing (11) on the Caltech101 dataset with 128 bits.

end, we propose an *adaptive discrete proximal linear method* (**ADPLM**), which adopts a self-tuning manner to adaptively choose the proper step-size for optimal convergence. Specifically, we initialize $\mu^0 = 0.5$, and update $\mu^{p+1} = 0.5\mu^p$ if $\mathcal{F}(\boldsymbol{C}^{p+1}) \leq \mathcal{F}(\boldsymbol{C}^p)$ for the p+1-th iteration, and $\mu^{p+1} = 1.2\mu^p$ otherwise. Moreover, to guarantee the convergence of ADPLM, we set $\mu^{p+1} \in (L, 2L)$ [32], where L is the Lipschitz constant. The comparison of these two algorithms is shown in Fig. 1. As can be seen, both DPLM and ADPLM can obtain their respective local optimum. However, DPLM with an improper step-size does not converge to the optimal point, as compared to ADPLM. In contrast, ADPLM converges to a better point than that DPLM does, since the objective value of ADPLM at the limit point is much lower. Therefore, we can conclude DPLM and ADPLM employ different learning strategies to approximate the global minimum asymptotically, but ADPLM employs a better approach with an adaptively determined learning step-size.

$\boldsymbol{G}$-Step: The optimal solution of the indicator matrix $\boldsymbol{G}$ at indices (i, j) can be easily obtained by

$$\boldsymbol{g}_{ij}^{p+1} = \begin{cases} 1, & j = \arg\min_s H(\boldsymbol{b}_i, \boldsymbol{c}_s^{p+1}), \\ 0, & otherwise, \end{cases} \tag{15}$$

where $H(\boldsymbol{b}_i, \boldsymbol{c}_s)$ is the hamming distance between the i-th binary code $\boldsymbol{b}_i$ and the s-th cluster centroid $\boldsymbol{c}_s$. It is noteworthy that the Hamming distance computation is clearly faster than the Euclidean distance.

Updating the weighting coefficient $\boldsymbol{\alpha}^v$: Let $g^v = \|\boldsymbol{B} - \boldsymbol{U}^v\phi(\boldsymbol{x}^v)\|_F^2 + \beta\|\boldsymbol{U}^v\|_F^2 - \frac{\gamma}{n} tr\left((\boldsymbol{U}^v\phi(\boldsymbol{x}^v))(\boldsymbol{U}^v\phi(\boldsymbol{x}^v))^T\right)$, then Eqn. (5) w.r.t α can be rewritten as:

$$\min_{\boldsymbol{\alpha}^v} \sum_{v=1}^{M} (\boldsymbol{\alpha}^v)^r g^v \ s.t. \ \sum_v \boldsymbol{\alpha}^v = 1, \boldsymbol{\alpha}^v > 0, \tag{16}$$

which can be solved by using the method of Lagrange multiplier. By introducing the Lagrange multiplier $\boldsymbol{\eta}$, the above problem becomes:

$$\min \mathcal{F}(\boldsymbol{\alpha}^v, \boldsymbol{\eta}) = \sum_{v=1}^{M} (\boldsymbol{\alpha}^v)^r g^v - \boldsymbol{\eta}(\sum_{v=1}^{M} \boldsymbol{\alpha}^v - 1). \tag{17}$$

Its partial derivatives with respect to $\boldsymbol{\alpha}^v$ and $\boldsymbol{\eta}$ are

$$\begin{cases} \frac{\partial \mathcal{F}}{\partial \boldsymbol{\alpha}^v} = & r(\boldsymbol{\alpha}^v)^{r-1} g^v - \boldsymbol{\eta}, \\ \frac{\partial \mathcal{F}}{\partial \boldsymbol{\eta}} = & \sum_{v=1}^{M} \boldsymbol{\alpha}^v - 1. \end{cases} \tag{18}$$

By setting $\nabla_{\boldsymbol{\alpha}^v, \boldsymbol{\eta}} \mathcal{F} = \boldsymbol{0}$, we have the optimal solution of $\boldsymbol{\alpha}^v$ as

$$\boldsymbol{\alpha}^v = \frac{(g^v)^{\frac{1}{1-r}}}{\sum_v (g^v)^{\frac{1}{1-r}}}. \tag{19}$$

We have so far presented the whole optimization procedures for the problem (5). An alternating optimization scheme is employed to iteratively update the mapping matrix $\boldsymbol{U}^v$, discrete representation $\boldsymbol{B}$, binary cluster centroids $\boldsymbol{C}$ and indicator $\boldsymbol{G}$. The optimization usually converges within 5 iterations in our experiments.

2.3 Convergence Analysis

The efficiency of BMVC requires the fast convergence of the elaborated alternating optimization procedures in Sec. 2.2.

Theorem 1. *The objective function $\mathcal{F}(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{G}, \boldsymbol{\alpha})$ in (5) is bounded. The proposed optimization algorithm monotonically decreases the value of $\mathcal{F}(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{G}, \boldsymbol{\alpha})$ in each optimization step.*

Proof. It is easy to see that problem (5) is bounded from below, i.e. $\mathcal{F}(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{G}, \boldsymbol{\alpha}) \geq 0$, due to the summation of norms with positive penalty parameters. It is obvious that $\{\boldsymbol{U}^p, \boldsymbol{B}^p, \boldsymbol{\alpha}^p\}$ generated via the process (7), (10) and (19), are the exact minimum points of the subproblems (6), (8) and (17), respectively. According to the theoretical analysis in [32], (13) has an analytical solution by using ADPLM. Based on the learning scheme of k-means, $\{\boldsymbol{C}, \boldsymbol{G}\}$ computed by using ADPLM and (15) are the optimal solution of subproblem (12). As a result, the value of the objective function $\mathcal{F}(\boldsymbol{U}, \boldsymbol{B}, \boldsymbol{C}, \boldsymbol{G}, \boldsymbol{\alpha})$ in (5) is decreasing in each iteration of the proposed optimization algorithm. □

Denote $\{\boldsymbol{U}^t, \boldsymbol{B}^t, \boldsymbol{C}^t, \boldsymbol{G}^t, \boldsymbol{\alpha}^t\}_{t=1}$ as $\boldsymbol{\Psi}^t$, and let $\{\boldsymbol{\Psi}^t\}_{t=1}$ be a sequence generated by the t-th main iteration of the proposed optimization, and then $\{\boldsymbol{\Psi}^t\}_{t=1}$ is a bounded below monotone decreasing sequence based on the above theorem. Therefore, according to the bounded monotone convergence theorem [46] that asserts the convergence of every bounded monotone sequence, the proposed optimization algorithm converges.

2.4 Complexity and Memory Load Analysis

The computational burden of BMVC is mainly composed of two parts, i.e. CDRL and BCSL. To save the time-consuming in (7), $\boldsymbol{W}$ is pre-computed outside of the main iterations in BMVC, and calculating mapping matrix $\boldsymbol{U}$ costs $\mathcal{O}(lmn)$. Computing the discrete representation $\boldsymbol{B}$ consumes $\mathcal{O}(Mlmn + lcn)$. So, the time complexity of CDRL is $\mathcal{O}((M+1)lmn + lcn)$. The most time-consuming part of optimizing BMVC is to solve BCSL, which needs $\mathcal{O}(nc)$ on l bitwise operations for p iterations. In total, the time complexity of BMVC learning is $\mathcal{O}(t((M+1)lmn + lcn + pnc))$, where t and p are empirically set to 5 and 10 in our experiments, respectively. In general, the computational complexity of the proposed optimization algorithm on BMVC learning is linear to the number of samples $\mathcal{O}(n)$.

For memory usage in BMVC, it is essential for storing the mapping matrix $\boldsymbol{U}^v$ for each view, and there are M real-valued matrices with the storage load $\mathcal{O}(lm)$. Importantly, the learned discrete representation and cluster centroids cost the *bit-wise* memory load $\mathcal{O}(l(n+c))$, which is much less than the memory load of k-means clustering requiring $\mathcal{O}(d(n+c))$ real-valued numerical storage costs.

TABLE 1: Clustering accuracies, NMI and purity comparisons of different methods on the Caltech101 dataset.

	Alg.	k-means	SC	LSC-K	AMGL	MVKM	MultiNMF	MLAN	OMVC	MVSC	LSH+bk-means	CKM	BMVC-A	BMVC-TS	**BMVC (ours)**
ACC	Gabor	0.1013	0.1167	0.1033	0.0989	0.1031	0.1183	0.1176	0.1128	0.1318	0.1021	0.1150	0.1414	0.1309	**0.1518**
	WM	0.1229	0.1198	0.1247	0.1192	0.1299	0.1389	0.1379	0.1379	0.1113	0.1128	0.1217	0.1317	0.1473	**0.1736**
	CENTRIST	0.1358	0.1050	0.1218	0.1421	0.1283	0.1528	0.1052	0.1473	0.1209	0.1072	0.1185	0.1392	0.1643	**0.1836**
	HOG	0.2488	0.2304	0.2466	0.2204	0.2468	0.2209	0.2177	0.2316	0.2201	0.1914	0.2006	0.2267	0.2415	**0.2534**
	GIST	0.2362	0.2308	0.2296	0.2657	0.2396	0.2361	0.1866	0.2358	0.2099	0.1935	0.1789	0.2345	0.2429	**0.2645**
	LBP	0.2072	0.1929	0.1943	0.2241	0.2274	0.2013	0.2159	0.2082	0.2190	0.1312	0.1974	0.2247	0.2295	**0.2353**
	All View	0.1331	0.1019	0.1832	0.2247	0.2669	0.2597	0.2171	0.2680	0.2463	0.1224	0.1859	0.2432	0.2501	**0.2930**
NMI	Gabor	0.2579	0.2840	0.2609	0.2527	0.2536	0.2725	0.1425	0.2616	0.2842	0.2603	0.2543	0.3125	0.3089	**0.3132**
	WM	0.2913	0.3002	0.2919	0.2921	0.2927	0.3151	0.1833	0.3074	0.2813	0.2614	0.2043	0.2904	0.3463	**0.3472**
	CENTRIST	0.3023	0.3251	0.3025	0.3030	0.3056	0.3208	0.1338	0.3196	0.3348	0.2680	0.2517	0.3144	0.3720	**0.3760**
	HOG	0.4700	0.4710	0.4664	0.4157	0.2869	0.4393	0.3126	0.4290	0.4516	0.3696	0.2260	0.4037	**0.4823**	0.4740
	GIST	0.4462	0.4481	0.4384	0.4038	0.3569	0.4456	0.2215	0.4245	0.4530	0.3690	0.2261	0.4102	0.4713	**0.4784**
	LBP	0.3996	0.4051	0.3837	0.3759	0.3540	0.4071	0.2217	0.3887	0.4036	0.2870	0.1610	0.4034	**0.4536**	0.4480
	All View	0.3056	0.3226	0.3074	0.3812	0.3703	0.4713	0.3119	0.4377	0.4690	0.2793	0.3453	0.4447	0.4817	**0.4900**
Purity	Gabor	0.2440	0.1380	0.2279	0.2441	0.2414	0.2532	0.1894	0.2379	0.1399	0.2359	0.2457	0.2972	0.3070	**0.3214**
	WM	0.2830	0.2974	0.2620	0.2816	0.2834	0.3099	0.2267	0.2806	0.3033	0.2452	0.2102	0.2827	0.3458	**0.3550**
	CENTRIST	0.2936	0.2309	0.2770	0.2956	0.2942	0.3081	0.1867	0.2917	0.2339	0.2507	0.2329	0.3088	0.3654	**0.3867**
	HOG	0.4623	0.4780	0.4046	0.4157	0.2988	0.4417	0.3782	0.4235	0.4123	0.3663	0.2911	0.4299	0.4637	**0.4824**
	GIST	0.4390	0.4513	0.4010	0.4204	0.3673	0.4401	0.2713	0.4257	0.4378	0.3641	0.2652	0.4199	0.4815	**0.4840**
	LBP	0.3808	0.4058	0.3789	0.3831	0.3467	0.4092	0.3002	0.3962	0.4228	0.2834	0.2385	0.4226	0.4507	**0.4531**
	All View	0.2909	0.2207	0.2962	0.3953	0.3808	0.4748	0.3663	0.4643	0.4567	0.2768	0.3401	0.4543	0.4824	**0.4907**

*Note: The bold **black** and bold **blue** numbers indicate the best single-view and multi-view clustering results,respectively.

TABLE 2: Time-consuming (in seconds) comparison of different methods on the Caltech101 dataset.

Alg.	k-means		SC		AMGL		MVKM		MultiNMF		OMVC		CKM		BMVC-TS		**BMVC (ours)**	
	Time	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup
Gabor	27	1×	200	0.14×	530	0.05×	278	0.10×	497	0.05×	25	1.08×	4	6.75 ×	8	3.38 ×	**1.2**	**22.50**×
WM	12	1×	96	0.13×	303	0.04×	290	0.04×	244	0.05×	16	0.75×	4	3.00 ×	10	1.20 ×	**1.3**	**9.23**×
CENTRIST	28	1×	229	0.12×	319	0.09×	385	0.07×	338	0.08×	23	1.22×	7	4.00 ×	5	5.60 ×	**1.4**	**20.00**×
HOG	77	1×	99	0.78×	673	0.11×	974	0.08×	1609	0.05×	80	0.96×	14	5.50 ×	9	8.56 ×	**1.3**	**59.23**×
GIST	29	1×	98	0.30×	591	0.05×	411	0.07×	554	0.05×	50	0.58×	7	4.14 ×	8	3.63 ×	**1.3**	**22.31**×
LBP	50	1×	97	0.52×	460	0.11×	405	0.12 ×	441	0.11×	62	0.81×	11	4.55 ×	7	7.14 ×	**1.4**	**35.71**×
All View	204	1×	442	0.46×	1772	0.12×	1894	0.11×	2054	0.10×	291	0.70×	22	9.27 ×	16	12.75 ×	**3.5**	**58.29**×

3 EXPERIMENTS

In this section, we evaluate the effectiveness of the proposed BMVC for large-scale multi-view image clustering task from the aspects of clustering performance, computational efficiency, and memory load. All the experiments are implemented using Matlab 2013a on a standard Window PC with an Intel 3.4-GHz CPU and 64-GB RAM.

3.1 Datasets and Evaluation Metrics

Five datasets are employed in the evaluation: Caltech101[2], NUS-WIDE-Obj[3], Cifar-10[4], Sun-397[5] and YouTube Faces[6]. Specifically, ***Caltech101*** consists of 9,144 images associated with 101 object and one background categories. ***NUS-WIDE-Obj*** is composed of 30,000 images from 31 classes, and ***Cifar-10*** is comprised of 60,000 tiny color images in 10 kinds of objects, with 6,000 images per class. ***Sun397*** contains 108,754 images in total for 397 categories, and the number of images varies from categories, but more than 100 images per category. A subset of ***YouTube Faces*** contains 152,549 faces including 66 different people, each of which has more than 1,500 face images. For each image, multi-view features are extracted to describe it. Specifically, following the experimental features used in [11], five different features are selected for Caltech101, i.e. 48-dim Gabor feature, 40-dim wavelet moments (WM), 254-dim CENTRIST feature, 1984-dim HOG feature, 512-dim GIST feature, and 928-dim LBP feature. Five publicly available features for NUS-WIDE-Obj are provided on its homepage website, i.e. 65-dim color Histogram (CH), 226-dim color moments (CM), 145-dim color correlation (CORR), 74-dim edge distribution (ED) and 129-dim wavelet texture (WT). For Cifar-10, Sun397 and YouTube Faces, we would like to extract three well-attested features, i.e. 768-dim color histogram (CH), 1024-dim Gist [47], 1152-dim histogram of oriented gradients (HOG) [48].

We report the experimental results using three most used evaluation metrics, i.e. clustering accuracy (ACC), normalized mutual information (NMI) and purity [49]. Moreover, running time and memory load are both compared. For the fair comparison, we use the codes by the corresponding authors with the default or optimal parameter settings of their original papers. To make a fair runtime comparison, the time costs of parameter tuning for all the methods are not included to reflect the pure efficiency of algorithms themselves. For the binary coding methods, the 128-bit coding length is used for all datasets, and the averaged clustering results are reported based on different random initializations.

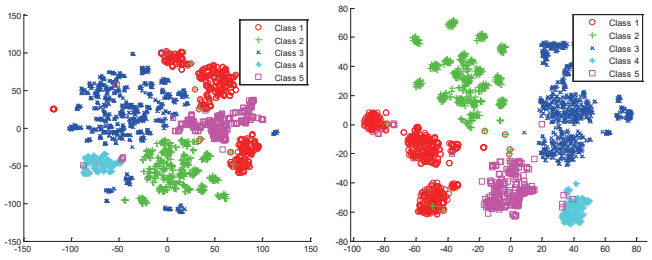
3.2 Toy Experiment

We first use the medium-size Caltech101 dataset to explicitly uncover the superiority of our BMVC by answering the following questions:

Q1. How does BMVC perform as compared to other state-of-the-art single- and multi-view clustering?
Q2. How does BMVC speed up multi-view clustering?
Q3. How do the nonlinear anchor embedding and the unified learning framework contribute to the overall effectiveness of BMVC?
Q4. Why BMVC outperforms the real-valued clustering?

Specifically, we compare BMVC to the state-of-the-art clustering methods consisting of single-view clustering, including k-means [1], SC [2], LSC-K [10] and two existing binary clustering methods (*i.e.*, LSH+bk-means [43] and CKM [12]), and multi-view clustering, including AMGL [21], MVKM [7], Multi-NMF [13], MLAN [22], OMVC [26] and MVSC [11]. Additionally, two variants of BMVC are

2. http://www.vision.caltech.edu/Image_Datasets/Caltech101/.
3. http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm.
4. https://www.cs.toronto.edu/~kriz/cifar.html.
5. http://vision.princeton.edu/projects/2010/SUN/.
6. https://www.cs.tau.ac.il/~wolf/ytfaces/.



(a) HOG (b) BMVC codes of HOG

Fig. 2: t-SNE visualization of (a) the original HOG feature and (b) 128-bit BMVC codes of the HOG feature for the randomly selected 5 categories (more than $2k$ samples) in Caltech101.

taken into comparison, *i.e.*, removing the nonlinear anchor embedding (BMVC-A), and BMVC with two separate steps of CDRL and binary k-means (BMVC-TS). For single-view clustering methods, we concatenate feature vectors of different views together for the 'All-view' clustering setting. The clustering results and running time comparisons on Caltech101 have been summarized in Table 1 and Table 2, respectively.

BMVC vs. Baselines (Q1): From Table 1, we can observe that our BMVC is superior to all the compared real-valued and binary clustering methods in most cases. Specifically, BMVC can achieve better or comparable results on every single view, which indicates that the learned discrete representation by BMVC is effective for clustering. For the multi-view clustering case ('All View' in the table), BMVC consistently outperforms all compared methods by large margins. This result demonstrates the effectiveness of BMVC on the multi-view representation learning and precise cluster structure learning. For the single-view methods such as k-means, SC and LSC-K, we observe that directly concatenating multi-view features may lead to redundant information in most situations, leading to the sub-optimal clustering results. Moreover, the data-dependent BMVC learns discrete representation from data, which is more effective than the data-independent binary code learning (as in LSH+bk-means). BMVC clearly outperforms the existing binary clustering methods such as LSH+bk-means and CKM due to the preferable unified framework of the optimal binary code learning and clustering structure construction. Finally, we can also notice that the single- and multi-view methods have similar performance on the single-view clustering task. In contrast, on the multi-view clustering task, the multi-view methods in most cases are greatly superior to the single-view ones. This is not surprising since multi-view methods can collectively consider the correlations between heterogeneous representations.

Running-time comparison (Q2): The time comparison shown in Table 2 demonstrates clearer advantages for binary clustering methods, which are much faster than the real-valued ones. The main reason is that they benefit from the highly efficient distance calculation using the Hamming metric rather than the Euclidean distance measurement. Particularly, our BMVC costs much less time than all other compared method, which also proves the superiority of the proposed efficient discrete optimization algorithm. Specifically, the binary clustering methods CKM and BMVC-TS are respectively 9.27 and 12.75 times faster than k-means for multi-view clustering, while the speed-up of our BMVC is more obvious by a margin of 58.29 times. Importantly, compared to the BMVC-TS model, our unified framework can greatly shorten the computation time of binary clustering.

BMVC model evaluation (Q3): By comparing the results between BMVC-A and BMVC, we can conclude that nonlinear anchor embedding is distinctly important to enhance the interpretation of the original features. Comparing with the two-step method BMVC-TS, the unified framework BMVC achieves better results due to the collaborative learning of discrete representation and binary cluster structures. Therefore, the nonlinear anchor embedding and the unified learning scheme are both indispensable to BMVC.

Why BMVC outperforms the real-valued clustering? (Q4): The results summarized in Table 1 clearly show that BMVC can attain competitive or higher clustering performance when comparing to the real-value clustering methods. The main reasons are: 1) The encouraging clustering performance of BMVC hinges on the proposed effective discrete optimization algorithm, where the learned high-quality collaborative representation well eliminates some redundant and noise information in the original real-valued features. Fig. 2 shows the t-SNE visualization of the HOG feature and the learned discrete representation by BMVC. As can be seen, the learned representations not only preserve the neighborhood structure of data in the code space but also mitigate the effect of possible disturbances from the original features. 2) BMVC incorporates the collaborative discrete representation learning and the precise data clustering into a joint learning framework, which is shown to be better than the approaches (*e.g.*, SC, LSC-K, AMGL, and MLAN) with disjoint learning steps.

3.3 Experiments on Large-scale Multi-view Datasets

To verify the strong capacity of BMVC on handling large-scale multi-view clustering, in this section, we compare with seven state-of-the-art scalable clustering methods on four datasets. The results are reported in Table 3. We do not compare with the multi-view methods on these large-scale datasets due to their demanding computation time. From these results, we have the following observations.

1) BMVC consistently obtains the highest results on the multi-view clustering task. In contrast, conventional single-view real-valued clustering methods do not perform well although they achieve promising single-view performance.

2) BMVC can also achieve encouraging single-view clustering results, in terms of ACC, NMI and purity on all datasets. This demonstrates that the discrete codes learned by BMVC are competitive in comparison with the real-valued representations, which also testifies the effectiveness of discrete representation for clustering.

3) We can also observe that BMVC using multiple features always outperforms that using single-view features. This indicates BMVC can capture the hidden correlations and complementary traits of multi-view representations.

In Fig. 3, we present the t-SNE visualization of the learned discrete representation by BMVC on Cifar-10 and YouTube-Faces, where the learned representation can effectively assemble similar samples into the same clusters.

TABLE 3: Clustering accuracies, NMI and purity comparisons of different methods on the four large-scale multi-view datasets.

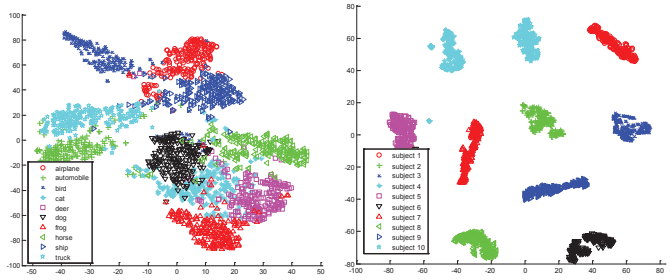
		Alg.	k-means	k-means++	k-Medoids	Nyström [9]	LSC-K	LSH+bk-means	CKM	BMVC-TS	**BMVC (ours)**
NUS-WIDE-OBJ	ACC	CH	0.1230	**0.1245**	0.1152	0.1047	0.1019	0.1035	0.1123	0.1232	0.1193
		CM	0.1179	**0.1188**	0.1114	0.1099	0.1163	0.1124	0.1036	0.1082	0.1029
		COCO	0.1074	0.1031	0.1069	0.0994	0.1047	0.1040	0.1117	0.1115	**0.1295**
		ED	0.1387	0.1383	0.1201	0.1098	0.1181	0.1124	0.1242	**0.1502**	0.1006
		WT	0.1143	0.1169	0.1176	0.1186	0.1182	0.1265	0.1257	0.1320	**0.1364**
		All View	0.1459	0.1375	0.1313	0.1220	0.1398	0.1214	0.1368	0.1123	**0.1680**
	NMI	CH	0.0893	0.0902	0.0864	0.0887	0.0819	0.0835	0.0756	0.0981	**0.0938**
		CM	0.1065	**0.1091**	0.0938	0.0874	0.1059	0.0911	0.0682	0.0840	0.1056
		COCO	0.0992	0.0963	0.0939	0.0812	0.0975	0.0929	0.0937	0.1093	**0.1096**
		ED	0.1256	**0.1274**	0.1168	0.0910	0.1201	0.1094	0.0873	0.1106	0.1014
		WT	0.1015	0.1024	0.0979	0.0996	0.1051	0.1038	0.1111	0.1236	**0.1301**
		All View	0.1415	0.1400	0.1224	0.1112	0.1399	0.1041	0.1154	0.1142	**0.1621**
	Purity	CH	0.1963	0.1977	0.2022	0.1996	0.1987	0.1912	0.1827	0.2166	**0.2170**
		CM	0.2109	0.2092	0.2022	0.2116	**0.2236**	0.2054	0.1723	0.2062	0.2128
		COCO	0.2137	0.2120	0.2097	0.1911	0.2110	0.2082	0.2046	0.2425	**0.2536**
		ED	0.2434	0.2524	0.2406	0.2312	**0.2575**	0.2451	0.1986	0.2406	0.2514
		WT	0.2394	0.2393	0.2340	0.2339	0.2395	0.2338	0.2355	0.2585	**0.2655**
		All View	0.2576	0.2615	0.2310	0.2313	0.2653	0.2301	0.2285	0.2250	**0.2872**
Cifar-10	ACC	CH	0.1771	0.1777	0.1787	0.1239	0.1003	0.1714	0.1681	0.1611	**0.1984**
		GIST	0.2847	0.2846	0.2242	0.2523	**0.2983**	0.2170	0.2480	0.2435	0.2853
		HOG	0.2351	0.2711	0.2348	0.2310	0.2514	0.2182	0.2125	**0.3057**	0.2752
		All View	0.2777	0.2761	0.2308	0.2375	0.3186	0.2294	0.2361	0.3273	**0.3472**
	NMI	CH	0.0502	0.0503	0.0435	0.1085	0.1014	0.0506	0.0368	0.0452	**0.1859**
		GIST	0.1693	0.1691	0.1299	0.1209	0.1696	0.1025	0.1123	0.1403	**0.1772**
		HOG	0.1547	0.1649	0.1307	0.1208	0.1846	0.1129	0.0979	**0.2142**	0.1895
		All View	0.1714	0.1745	0.1342	0.1260	0.2170	0.1012	0.1077	0.2163	**0.2399**
	Purity	CH	0.1880	0.1882	0.1834	0.1294	0.1004	0.1886	0.1791	0.1754	**0.2628**
		GIST	0.2926	0.3053	0.2374	0.2562	**0.3142**	0.2296	0.2480	0.2643	0.2801
		HOG	0.2666	0.2946	0.2493	0.2521	0.2995	0.2293	0.2228	**0.3140**	0.2694
		All View	0.3064	0.3053	0.2485	0.2610	0.3523	0.2313	0.2414	0.3334	**0.3752**
Sun-397	ACC	CH	0.0330	0.0278	0.0305	0.0182	0.0290	0.0270	0.0289	0.0296	**0.0348**
		GIST	0.0511	0.0564	0.0501	0.0528	0.0508	0.0313	0.0260	**0.0565**	0.0518
		HOG	0.0518	0.0515	0.0429	0.0429	0.0467	0.0267	0.0279	**0.0529**	0.0466
		All View	0.0544	0.0559	0.0307	0.0474	0.0490	0.0277	0.0272	0.0568	**0.0605**
	NMI	CH	0.1532	0.1782	0.1737	0.1034	0.1788	0.1782	0.1368	0.1656	**0.2010**
		GIST	0.2454	0.2528	0.2067	0.2424	0.2434	0.1903	0.1321	**0.2600**	0.2449
		HOG	0.2542	0.2426	0.1828	0.2272	0.2339	0.1641	0.1651	**0.2446**	0.2210
		All View	0.2556	0.2534	0.1756	0.2442	0.2788	0.1716	0.1692	0.2595	**0.3013**
	Purity	CH	0.0488	0.0533	0.0533	0.0301	0.0517	0.0518	0.0436	0.0507	**0.0585**
		GIST	0.0867	0.0939	0.0701	0.0815	0.0876	0.0564	0.0415	**0.0972**	0.0871
		HOG	0.0898	0.0862	0.0604	0.0768	0.0807	0.0477	0.0476	**0.0911**	0.0846
		All View	0.0881	0.0928	0.0525	0.0844	0.0817	0.0499	0.0484	0.0969	**0.0981**
YouTube-Faces	ACC	CH	0.3414	0.3363	0.3380	0.2485	0.4260	0.3587	0.3846	0.4625	**0.6429**
		GIST	0.4324	0.4305	0.3840	0.2922	0.4760	0.3301	0.3588	0.5523	**0.5986**
		HOG	0.6095	0.5992	0.5322	0.5915	**0.6607**	0.4184	0.4011	0.5909	0.5971
		All View	0.5553	0.4430	0.3392	0.5804	0.6265	0.3880	0.4112	0.6347	**0.6827**
	NMI	CH	0.5011	0.5031	0.4665	0.4081	0.5945	0.4814	0.5682	0.6339	**0.7750**
		GIST	0.5457	0.5376	0.5012	0.4460	0.6122	0.4470	0.4863	0.7316	**0.7498**
		HOG	0.7306	0.7364	0.6547	0.7175	**0.7634**	0.4698	0.5756	0.7326	0.7416
		All View	0.7236	0.6016	0.4786	0.7125	0.7622	0.5486	0.5073	0.7650	**0.8104**
	Purity	CH	0.4016	0.4099	0.3977	0.3019	0.4917	0.4096	0.4534	0.5765	**0.6908**
		GIST	0.4823	0.4823	0.4505	0.3373	0.5531	0.3796	0.3780	0.6490	**0.6505**
		HOG	**0.6896**	0.6696	0.6026	0.6641	0.6834	0.4566	0.4528	0.6621	0.6793
		All View	0.6431	0.5085	0.4043	0.6446	0.6939	0.4687	0.4445	0.7032	**0.7275**

*Note: BMVC can achieve comparable results for single-view clustering, but clearly outperforms the other methods for multi-view clustering.

TABLE 4: Time-consuming (in seconds) comparison of different methods on the four large-scale multi-view datasets.

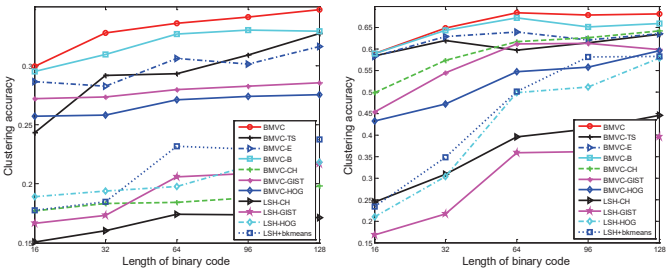
	Alg.	k-means		k-means++		Nyström		LSC-K		LSH+bk-means		CKM		BMVC-TS		**BMVC (ours)**	
		Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup
NUS-WIDE	CH	40	1×	36	1.11×	12	3.33×	233	0.17 ×	6	6.67 ×	14	2.86 ×	19	2.11 ×	**2.9**	**13.79**×
	CM	102	1×	71	1.44×	17	6.00×	231	0.44 ×	7	14.57 ×	22	4.64 ×	14	7.29 ×	**2.7**	**37.78**×
	COCO	105	1×	55	1.91×	14	7.50×	196	0.54 ×	8	13.13 ×	19	5.53 ×	16	6.56 ×	**2.8**	**37.50**×
	ED	79	1×	43	1.84×	16	4.94×	187	0.42 ×	7	11.29 ×	14	5.64 ×	15	5.27 ×	**2.9**	**27.24**×
	WT	115	1×	74	1.55×	18	6.39×	206	0.56 ×	10	11.50 ×	18	6.39 ×	15	7.67 ×	**2.9**	**39.66**×
	All View	305	1×	210	1.45×	30	10.17×	339	0.90 ×	**9**	**33.89** ×	27	11.30 ×	34	8.97 ×	10	30.50×
Cifar-10	CH	389	1×	229	1.70×	34	11.44 ×	895	0.43×	8	48.63 ×	30	12.97 ×	24	16.20 ×	**4.7**	**82.77**×
	GIST	434	1×	236	1.84×	25	17.36 ×	786	0.55×	7	62.00 ×	31	14.00 ×	18	24.11 ×	**4.5**	**96.44**×
	HOG	510	1×	292	1.75×	25	20.40 ×	922	0.55×	7	72.86 ×	33	15.45 ×	19	26.84 ×	**4.7**	**108.51**×
	All View	1299	1×	698	1.86×	51	25.47 ×	1609	0.81×	12	108.25 ×	65	19.99 ×	37	35.11 ×	**11**	**118.09**×
Sun-397	CH	2648	1×	2354	1.12×	1302	2.03 ×	3694	0.72×	43	61.58 ×	297	8.92 ×	84	31.52 ×	**28**	**94.57**×
	GIST	1963	1×	2642	0.74×	1402	1.40 ×	4412	0.45×	43	45.65 ×	247	7.95 ×	89	22.06 ×	**29**	**67.69**×
	HOG	3320	1×	4561	0.73×	1352	2.46 ×	4303	0.77×	46	72.17 ×	321	10.34 ×	83	40.00 ×	**29**	**114.48**×
	All View	6127	1×	6569	0.93×	1626	3.77 ×	7921	0.77×	56	109.41 ×	455	13.47 ×	92	66.60 ×	**37**	**165.60**×
YouTube	CH	1920	1×	1389	1.38×	883	2.17×	3670	0.52×	28	68.57 ×	142	13.52 ×	91	21.10 ×	**25**	**76.80**×
	GIST	1976	1×	1485	1.33×	917	2.15×	4293	0.46×	38	52.00 ×	143	13.82 ×	57	34.67 ×	**18**	**109.78**×
	HOG	2207	1×	1700	1.30×	899	2.45×	6542	0.34×	39	56.58 ×	242	9.12 ×	78	28.30 ×	**25**	**88.28**×
	All View	6191	1×	5648	1.10×	1006	6.15×	12061	0.51×	**40**	**154.78**×	391	15.83 ×	101	61.30 ×	47	131.72×

BMVC Components Analysis: To evaluate the indispensability of different components of BMVC, the effectiveness of different components is evaluated with various code lengths in Fig. 4. Specifically, in addition to BMVC-TS, BMVC-E regards each view features equivalently by setting the weighting coefficient $\boldsymbol{\alpha}$=$\mathbf{1}$, while BMVC-B removes the balanced factors on the collaborative discrete representation and clustering centroids. BMVC-'view' and LSH-'view' respectively denote the clustering results using BMVC and LSH+bk-means on each single-view features, and the remaining symbols refer to the multi-view clustering results. From Fig. 4, we can observe that lacking of any component will lead to worse performance than the full model. BMVC collectively considers the semantic



(a) Cifar-10 (b) YouTube-Faces

Fig. 3: t-SNE visualization of 128-bit BMVC codes of 10 categories in the (a) Cifar-10 and (b) YouTube-Faces, respectively.



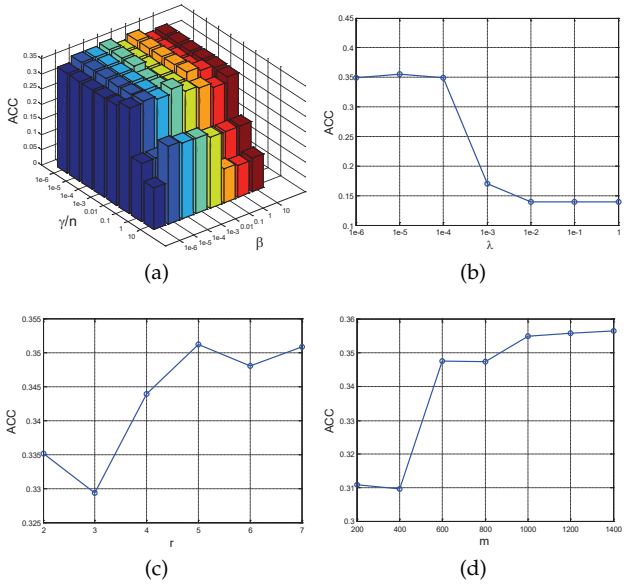
(a) Cifar-10 (b) YouTube-Faces

Fig. 4: Clustering accuracy comparison with different number of bits for binary representation learning methods.

correlations between different views resulting in the best performance in comparison with the single-view results even for the very short code length.

Parameter Sensitivity Analysis: In the proposed framework, there are five parameters to be tuned, i.e. β, γ, λ, r, and m. In the experiments, we first employ the grid search strategy to find the best choices for all parameters on the smaller dataset, i.e. Caltech101. When applying to another dataset, we only fine-tune these parameters on randomly selected $10k$ data points from this dataset, and then the optimal parameters are exploited for clustering the whole dataset. Fig. 5 (a)-(d) explicitly show the clustering accuracy variation curves w.r.t. different parameters on Cifar-10, respectively. We can see that the best clustering results are established when the values of β and γ/n are not too large, usually smaller than 1 but bigger than 10^{-3}. When the values of λ and r are respectively 10^{-5} and 5, the optimal cluster performance is obtained. When the anchor number m is bigger than 1000, the performance improvements become slower.

Performance v.s. Number of Clusters:: All the above experiments are evaluated by the exact number of clusters in data, i.e. c, but how does the performance change w.r.t. different numbers of clusters? To this end, we also perform experiments on the Cifar-10 dataset to evaluate the stabilities of different methods w.r.t. number of clusters. Fig. 6 illustrates the clustering performance changes with increasing number of clusters from 5 to 40 with an interval 5. Intuitively, the best clustering performance of BMVC is obtained when the number of cluster is 10. This suggests that 10 is the optimal number of clusters. Moreover,



(a) (b) (c) (d)

Fig. 5: Clustering accuracy v.s. different parameters (a) β and γ/n, (b) λ, (c) r, and (d) m on the Cifar-10 dataset, respectively.

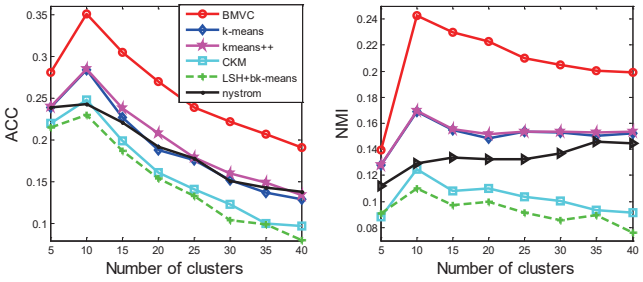


Fig. 6: Clustering performance v.s. number of clusters by using different methods on the Cifar-10 dataset.

the clustering performance of BMVC is superior to all compared methods. Therefore, our BMVC is capable of tackling the varying clusters problem.

3.4 Time and Memory Load Comparison

The running time comparisons of all methods on four multi-view datasets are summarized in Table 4, where the fastest method is established by BMVC in almost all cases. In particular, BMVC is 165× and 214× faster than k-means and LSC-K, respectively, on Sun-397 for multi-view clustering. BMVC can cluster more than $150k$ multi-view samples from YouTube-Faces in 47 seconds, while k-means costs more than 1.7 hours. We also argue that *the efficiency of BMVC not only benefits from the highly efficient computation in Hamming space, but also from the fast convergence of our optimization algorithm*. In the experiments, we found that our method could converge efficiently, and the corresponding performance approximated to be stable within 5-10 iterations.

It should be noted that our BMVC requires similar memory costs to other binary clustering methods, but can significantly reduce the memory overhead in comparison with the real-valued methods. Table 5 shows the memory load of BMVC and k-means. We can observe that BMVC

TABLE 5: Memory load comparison of 'All-view' k-means and BMVC on four large-scale multi-view datasets. 'Memory' and 'Reduction' respectively denote the memory load and times of memory reduction against k-means.

Datasets	k-means		**BMVC (ours)**	
	Memory	Reduction	Memory	Reduction
NUS-WIDE-OBJ	153MB	1×	**0.48MB**	**319**×
Cifar-10	1.41GB	1×	**0.96MB**	**1469**×
Sun-397	2.57GB	1×	**1.74MB**	**1477**×
YouTube-Faces	3.59GB	1×	**2.44MB**	**1471**×

significantly reduces the memory load for large-scale clustering. For example, for Sun-397 with over $108k$ multi-view features, BMVC only needs 1.74MB memory, which is 1477 times lower than the real-valued k-means. Therefore, our BMVC can manipulate large-scale multi-view clustering with both efficient computation and much less memory requirement.

4 Conclusion

In this paper, a principled binary multi-view clustering method, dubbed BMVC, was proposed for solving the challenging problem of multi-view clustering on large-scale image data. In BMVC, the collaborative discrete representations and binary cluster structures were jointly learned, which could effectively integrate the collaborative information from multiple views. Moreover, an effective alternating optimization algorithm with guaranteed convergence was proposed to ensure the high-quality binary solutions. Extensive experiments performed on the large-scale multi-view datasets showed the clear superiority of BMVC in comparison with state-of-the-art clustering methods in terms of clustering performance, meanwhile with significantly reduced computation and memory overhead. Similar to most clustering methods, we found the main limitation of our method may be sensitive to some parameter initializations, which remains to our future study.

References

[1] J. Hartigan, M. Wong, "Algorithm AS 136: A k-means clustering algorithm," J. Royal Stat. Soc., 28(1): 100-108, 1979.
[2] A. Y. Ng, M. I. Jordan, Y. Weiss, "On spectral clustering: analysis and an algorithm," in NIPS., 14(2): 849-856, 2011.
[3] P. Berkhin, "A survey of clustering data mining techniques," Grouping multidimensional data, pp. 25-71, 2006.
[4] A. K. Jain, "Data clustering: 50 years beyond k-means," Pattern Recogn. Lett., 31(8): 651-666, 2010.
[5] C. Xu, D. Tao, C. Xu, "A survey on multi-view learning," arXiv preprint arXiv:1304.5634, 2013.
[6] K. Chaudhuri, S. Kakade, et al. "Multi-view clustering via canonical correlation analysis," in ICML, pp. 129-136, 2009.
[7] X. Cai, F. Nie, H. Huang, " Multi-view k-Means clustering on big data," in IJCAI, pp. 2598-2604, 2013.
[8] X. He, S. Yan, et al. "Face recognition using Laplacianfaces," IEEE Trans. PAMI, 27(3):328 - 340, 2005.
[9] W. Chen, Y. Song, et al. "Parallel spectral clustering in distributed systems," IEEE Trans. PAMI, 33(3): 568-586, 2011.
[10] X. Chen, D. Cai, "Large scale spectral clustering with Landmark-based representation," in AAAI, pp. 313-318, 2011.
[11] Y. Li, F. Nie, et al. "Large-scale multi-view spectral clustering via bipartite graph," in AAAI, pp. 2750-2756, 2015.
[12] X. Chen, W. Liu, et al. "Compressed k-means for large-scale clutering," in AAAI, pp. 2527-2533, 2017.
[13] J. Liu, C. Wang, et al. "Multi-view clustering via joint nonnegative matrix factorization," in ICDM, pp. 252-260, 2013.
[14] D. Arthur, S. Vassilvitskii, "k-means++: The advantages of careful seeding," Symp. Discret. Algoritm., pp. 1027-1035, 2007.
[15] Y. Ding, Y. Zhao, et al. "Yinyang k-means: A drop-in replacement of the classic k-means with consistent speedup," in ICML, pp. 579-587, 2015.
[16] G. Hamerly, J. Drake, "Accelerating Lloyds algorithm for k-means clustering," Partitional Clustering Algorithms, pp. 41-78, 2015.
[17] J. Newling, F. Fleuret, "Fast k-means with accurate bounds," in ICML, EPFL-CONF-219846, 2016.
[18] O. Bachem, M. Lucic,et al. "Approximate k-means++ in sublinear time," in AAAI, pp. 1459-1467, 2016.
[19] A. Kumar, P. Rai P, H. Daume, "Co-regularized multi-view spectral clustering," in NIPS, pp. 1413-1421, 2011.
[20] T. Xia, D. Tao, et al. "Multiview spectral embedding," TCYB, 40(6): 1438-1446, 2010.
[21] F. Nie, J. Li, X. Li, "Parameter-free auto-weighted multiple graph learning: a framework for multiview clustering and semi-supervised classification," in IJCAI, pp. 1881-1887, 2016.
[22] F. Nie, G. Cai, X. Li, "Multi-view clustering and semi-supervised classification with adaptive neighbours," in AAAI, pp. 2408-2414, 2017.
[23] H. Wang, F. Nie, et al. "Fast nonnegative matrix tri-factorization for large-scale data coclustering," in IJCAI, pp. 1553-1558, 2011.
[24] Y. Yang, F. Shen, et al., "A unified framework for discrete spectral clustering," in IJCAI, pp. 2273-2279, 2016.
[25] Y. Yang, F. Shen, et al., "Discrete Nonnegative Spectral Clustering," in IEEE Trans. KDE, 29(9): 1834-1845, 2017.
[26] W. Shao, L. He, et al., "Online multi-view clustering with incomplete views," in IEEE Big Data, pp. 1012-1017, 2016.
[27] J. Wang, T. Zhang, et al. "A survey on learning to hash," IEEE Trans. PAMI, 40(4): 769-790, 2018.
[28] F. Shen, Y. Xu, et al. "Unsupervised Deep Hashing with Similarity-Adaptive and Discrete Optimization," IEEE Trans. PAMI, 2018.
[29] F. Shen, Y. Yang, et al. "Asymmetric Binary Coding for Image Search," IEEE Trans. Multimedia, 19(9): 2022–2032, 2017.
[30] L. Liu, M. Yu, L. Shao, "Latent Structure Preserving Hashing," IJCV 122(3): 439-457, 2017.
[31] L. Liu, M. Yu, L. Shao, "Learning Short Binary Codes for Large-scale Image Retrieval," IEEE Trans. IP, 26(3): 1289-1299, 2017.
[32] F. Shen, X. Zhou, et al. "A fast optimization method for general binary code learning," IEEE Trans. IP, 25(12): 5610-5621, 2016.
[33] Y. Gong, S. Lazebnik, et al. "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," IEEE Trans. PAMI, 35(12): 2916-2929, 2013.
[34] L. Liu, M. Yu and L. Shao, "Latent structure preserving hashing," in IJCV, 122(3): 439-457, 2017.
[35] F. Shen, C. Shen, et al. "Supervised discrete hashing," in CVPR, pp. 37-45, 2015.
[36] L. Liu, M. Yu, L. Shao, "Projection bank: From high-dimensional data to medium-length binary codes," in ICCV, pp. 2821-2829, 2015.
[37] Y. Gong, S. Kumar, et al. "Learning binary codes for high-dimensional data using bilinear projections," in CVPR, pp. 484-491, 2013.
[38] F. Shen, Y. Mu, et al. "Classification by retrieval: binarizing data and classifier," in SIGIR, 2017.
[39] L. Wu, Y. Wang, "Robust hashing for multi-view data: Jointly learning low-rank kernelized similarity consensus and hash functions," IVC, 57: 58-66, 2017.
[40] R. Yang, Y. Shi, X. Xu, "Discrete Multi-view Hashing for Effective Image Retrieval," in ACM ICMR, pp. 175-183, 2017.
[41] L. Xie, J. Shen, et al., "Dynamic Multi-View Hashing for Online Image Retrieval," in IJCAI, pp. 3133-3139,2017.
[42] L. Liu, M. Yu, L. Shao, "Multiview Alignment Hashing for Efficient Image Search," IEEE Trans. IP, 24(3):956-966, 2015.
[43] Y. Gong, M. Pawlowski, et al. "Web scale photo hash clustering on a single machine," in CVPR, pp. 19-27, 2015.
[44] Z. Zhang, L. Liu, et al. "Highly-Economized Multi-View Binary Compression for Scalable Image Clustering," in ECCV, 2018.
[45] C. Ding, X. He, H. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in ICDM, pp. 606-610, 2005.
[46] W. Rudin, "Principles of mathematical analysis," New York: McGraw-hill, 1964.
[47] A. Oliva, A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," IJCV, 42(3): 145-175, 2001.
[48] N. Dalal, B. Triggs, "Histograms of oriented gradients for human detection," in CVPR, pp. 886-893, 2005.
[49] C. D. Manning, P. Raghavan, H. Schutze, "Introduction to information retrieval," Cambridge: Cambridge university press, 2008.