Highly-Economized Multi-View Binary Compression for Scalable Image Clustering

Zheng Zhang[1,2,3*], Li Liu[3*], Jie Qin[4*], Fan Zhu[3], Fumin Shen[5], Yong Xu[1†], Ling Shao[3], and Heng Tao Shen[5]

[1] Harbin Institute of Technology (Shenzhen), China
[2] The University of Queensland, Australia
[3] Inception Institute of Artificial Intelligence, UAE
[4] Computer Vision Laboratory, ETH Zurich, Switzerland
[5] University of Electronic Science and Technology of China, China

Abstract. How to economically cluster large-scale multi-view images is a long-standing problem in computer vision. To tackle this challenge, we introduce a novel approach named **Highly-economized Scalable Image Clustering (HSIC)** that radically surpasses conventional image clustering methods via binary compression. We intuitively unify the binary representation learning and efficient binary cluster structure learning into a joint framework. In particular, common binary representations are learned by exploiting both sharable and individual information across multiple views to capture their underlying correlations. Meanwhile, cluster assignment with robust binary centroids is also performed via effective discrete optimization under ℓ_{21}-norm constraint. By this means, heavy continuous-valued Euclidean distance computations can be successfully reduced by efficient binary XOR operations during the clustering procedure. To our best knowledge, HSIC is the first binary clustering work specifically designed for scalable multi-view image clustering. Extensive experimental results on four large-scale image datasets show that HSIC consistently outperforms the state-of-the-art approaches, whilst significantly reducing *computational time* and *memory footprint*.

Keywords: Large-scale image clustering · binary code learning · binary clustering · multi-view features

1 Introduction

Image clustering is a commonly used unsupervised analytical technique for practical computer vision applications [17]. The aim of image clustering is to discover the natural and interpretable structure of image representations, so as to group images that are similar to each other into the same cluster. Based on the number of sources where images are collected or number of features how images are described, existing clustering methods can be divided into single-view image clustering (SVIC) [1, 16, 32, 36] and multi-view[6] image clustering

[*] indicates equal contributions; † indicates the corresponding author.
[6] Despite 'multi-view' can refer to multiple features, domains or modalities, in this paper, we solely focus on the clustering problem for images with multiple features (*e.g.*, LBP, HOG and GIST).

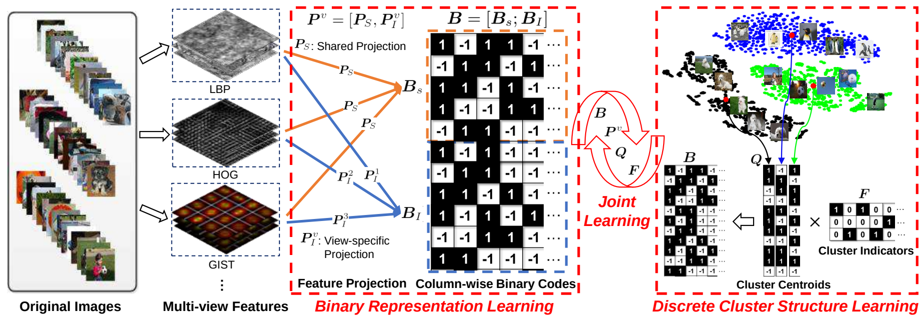


Fig. 1: The pipeline of HSIC. Common binary representation learning and discrete cluster structure learning are jointly and efficiently solved by alternating optimization.

(MVIC) [3, 4, 22, 47, 48]. Recently, MVIC [3, 48, 51] has been evoking more and more attention due to the flexibility of extracting multiple heterogeneous features from a single image. Compared to SVIC, MVIC has access to more characteristics and structural information of the data, and the features from diverse views can potentially complement each other and produce more effective clustering performance.

Existing MVIC methods can be roughly divided into three groups: multi-view spectral clustering [19, 30, 31], multi-view matrix factorization [4, 22, 37], and multi-view subspace clustering [13,45,49]. Multi-view spectral clustering [47] constructs multiple similarity graphs to achieve a common or similar eigenvector matrix on all views, and then generates consensus data partitions, which hinge crucially on the single-view spectral clustering [29]. Due to the straightforward interpretability of matrix factorization [20], multi-view matrix factorization methods [4, 22] integrate information from multiple views towards a compatible common consensus, or decompose the heterogeneous features into specified centroid and cluster indicator matrices. Different from the above strategies, multi-view subspace clustering [13] employs the complementary properties across multiple views to uncover the common latent subspace and quantify the genuine similarities. Some other kernel-based MVIC methods [10,42] exploit a linear or a non-linear kernel on each view. Note that SVIC (*e.g.*, k-means [16] and spectral clustering [29]) can also be leveraged to deal with multi-view clustering problem. A common practice for them is to perform clustering on either any single-view feature or simply concatenated multiple features [47, 48].

Although SVIC and MVIC methods have achieved much progress on small- and middle-scale data, both of them will become intractable (because of unaffordable computation and memory overhead) when dealing with large-scale data with high dimensionality, which is a typical case in the era of 'big data'. As pointed out in [15, 41], we argue that real-valued features are the essential bottleneck restricting the scalability of existing clustering methods. To address this issue, inspired by the recent advances on compact binary coding (*a.k.a.* hashing) [5, 23, 24, 27, 34, 39, 40, 43], we aim to develop a feasible binary clustering technique for large-scale MVIC. Specifically, we transform the original real-valued Euclidean space to the low-dimensional binary Hamming space, based on which an efficient clustering solution can then be devised. In this way, time-consuming Euclidean distance measures (typically of $\mathcal{O}(Nd)$ complexity, where N and d respectively indicate the data size and dimension) for real-valued data can be substantially eliminated by the extremely fast XOR operations (of $\mathcal{O}(1)$ complexity) for compact binary codes. Note that the proposed method is also potentially promising in practical use cases where computation and memory resources are limited (*e.g.*, on wearable or mobile devices).

As shown in Fig. 1, we particularly develop a *Highly-economized Scalable Image Clustering* (HSIC) framework for efficient large-scale MVIC. HSIC jointly learns the effective common binary representations and robust discrete cluster structures. The former can maximally preserve both sharable and view-specific/individual information across multiple views; the latter can significantly promote the computational efficiency and robustness of clustering. The joint learning strategy is superior to separately learning each objective by facilitating the collaboration between both objectives. An efficient alternating optimization algorithm is developed to address the joint discrete optimization problem. The main contributions of this work include:

1) To the best of our knowledge, HSIC is the pioneering work with large-scale MVIC capability, where common binary representations and robust binary cluster structures can be obtained in a unified learning framework.

2) HSIC captures both sharable and view-specific information from multiple views to fully exploit the complementation and individuality of heterogeneous image features. The sparsity-induced ℓ_{21}-norm is imposed on the clustering model to further alleviate its sensitivity against outliers and noise.

3) Extensive experimental results on four image datasets clearly show that HSIC can reduce the *memory footprint* and *computational time* up to **951** and **69.35** times respectively over the classical k-means algorithm, whilst consistently outperform the state-of-the-art approaches.

Notably, two works [15,41] in the literature are most relevant to ours. [15] introduced a two-step binary k-means approach, in which clustering is performed on the binary codes obtained by Iterative Quantization (ITQ) [14], and [41] integrated binary structural SVM and k-means. Our HSIC fundamentally differs from them in the following aspects: 1) [15] and [41] are SVIC methods, while HSIC is specially designed for MVIC; 2) [15] divides the clustering task into two unconnected procedures, which completely eliminate the important tie between the binary coding and cluster structure learning. Meanwhile, the binary codes learned by [41] are too weak to achieve satisfactory results because of lacking adequate representative capability. More importantly, both methods cannot make full use of the complementary properties of multiple views for scalable MVIC, which is also shown in [50].

In the next section, we will introduce the detailed framework of our HSIC and then elaborate on the alternating optimization algorithm. The analysis in terms of computational complexity and memory load will also be presented.

2 Highly-economized Scalable Image Clustering

Suppose we have a set of multi-view image features $\mathcal{X} = \{\boldsymbol{X}^1, \cdots, \boldsymbol{X}^m\}$ from m views, where $\boldsymbol{X}^v = [\boldsymbol{x}_1^v, \cdots, \boldsymbol{x}_N^v] \in \Re^{d_v \times N}$ is the accumulated feature matrix from the v-th view. d_v and N denote the dimensionality and the number of data points in $\boldsymbol{X}^v$, respectively. $\boldsymbol{x}_i^v \in \Re^{d_v \times 1}$ is the i-th feature vector from the v-th view. The main objective of unsupervised MVIC is to partition $\mathcal{X}$ into c groups, where c is the number of clusters. In this work, to address the large-scale MVIC problem, our HSIC aims to perform binary clustering in the much lower-dimensional Hamming space. Particularly, we perform multi-view compression (*i.e.*, project multi-view features onto the common Hamming space) by learning the compatible **common binary representation** via the complimentary characteristics of multiple views. Meanwhile, **robust binary cluster structures** are formulated in the learned Hamming space for efficient clustering.

As a preprocessing step, we first normalize the features from each view as zero-centered vectors. Inspired by [26, 40], in this work, each feature vector is encoded by the simple nonlinear RBF kernel mapping, *i.e.*, $\psi(\boldsymbol{x}_i^v) = [exp(-\|\boldsymbol{x}_i^v - \boldsymbol{a}_1^v\|^2/\gamma), \cdots, exp(-\|\boldsymbol{x}_i^v - \boldsymbol{a}_l^v\|^2/\gamma)]^\top$, where γ is the pre-defined kernel width, and $\psi(\boldsymbol{x}_i^v) \in \Re^{l \times 1}$ denotes an l-dimensional nonlinear embedding for the i-th feature from the v-th view. Similar to [25,26,40], $\{\boldsymbol{a}_i^v\}_{i=1}^l$ are randomly selected l anchor points from $\boldsymbol{X}^v$ ($l = 1000$ is used for each view in this work). Subsequently, we will introduce how to learn the common binary representation and robust binary cluster structure respectively, and finally end up with a joint learning objective.

1) Common Binary Representation Learning. We consider a family of K hashing functions to be learned in HSIC, which quantize each $\psi(\boldsymbol{x}_i^v)$ into a binary representation $\boldsymbol{b}_i^v = [b_{i1}^v, \cdots, b_{iK}^v]^T \in \{-1, 1\}^{K \times 1}$. To eliminate the semantic gaps between different views, HSIC generates the common binary representation by combining multi-view features. Specifically, HSIC simultaneously projects features from multiple views onto a common Hamming space, *i.e.,* $\boldsymbol{b}_i = sgn\big((\boldsymbol{P}^v)^\top \psi(\boldsymbol{x}_i^v)\big)$, where $\boldsymbol{b}_i$ is the common binary code of the i-th features from different views (*i.e.,* $\boldsymbol{x}_i^v$, $\forall v = 1, ..., m$), $sgn(\cdot)$ is an element-wise sign function, $\boldsymbol{P}^v = [\boldsymbol{p}_1^v, \cdots, \boldsymbol{p}_K^v] \in \Re^{l \times K}$ is the mapping matrix for the v-th view and $\boldsymbol{p}_i^v$ is the projection vector for the i-th hashing function. As such, we construct the learning function by minimizing the following quantization loss:

$$\min_{\boldsymbol{P}^v, \boldsymbol{b}_i} \sum_{v=1}^{m} \sum_{i=1}^{N} \|\boldsymbol{b}_i - (\boldsymbol{P}^v)^\top \psi(\boldsymbol{x}_i^v)\|_F^2. \tag{1}$$

Since different views describe the same subject from different perspectives, the projection $\{\boldsymbol{P}^v\}_{v=1}^m$ should capture the shared information that maximizes the similarities of multiple views, as well as the view-specific/individual information that distinguishes individual characteristics between different views. To this end, we decompose each projection into the combination of sharable and individual projections, *i.e.,* $\boldsymbol{P}^v = [\boldsymbol{P}_S, \boldsymbol{P}_I^v]$. Specifically, $\boldsymbol{P}_S \in \Re^{l \times K_S}$ is the shared projection across multiple views, while $\boldsymbol{P}_I^v \in \Re^{l \times K_I}$ is the individual projection for the v-th view, where $K = K_S + K_I$. Therefore, HSIC collectively learns the common binary representation from multiple views using

$$\min_{\boldsymbol{P}^v,\boldsymbol{B},\alpha^v} \sum_{v=1}^{m} (\alpha^v)^r \big(\|\boldsymbol{B} - (\boldsymbol{P}^v)^\top \psi(\boldsymbol{X}^v)\|_F^2 + \lambda_1 \|\boldsymbol{P}^v\|_F^2\big),$$
$$s.t.\ \sum_v \boldsymbol{\alpha}^v = 1, \boldsymbol{\alpha}^v > 0, \boldsymbol{B} = [\boldsymbol{B}_s; \boldsymbol{B}_I] \in \{-1,1\}^{K\times N}, \boldsymbol{P}^v = [\boldsymbol{P}_s, \boldsymbol{P}_I^v], \tag{2}$$

where $\boldsymbol{B} = [\boldsymbol{b}_1, \cdots, \boldsymbol{b}_N]$, $\boldsymbol{\alpha} = [\alpha^1, \cdots, \alpha^m] \in \Re^m$ weighs the importance of different views, $r > 1$ is a constant managing the weight distributions, and λ_1 is a regularization parameter. The second term is a regularizer to control the parameter scales.

Moreover, from the information-theoretic point of view, the information provided by each bit of the binary codes needs to be maximized [2]. Based on this point and motivated by [14, 44], we adopt an additional regularizer for the binary codes $\boldsymbol{B}$ using maximum entropy principle, *i.e.*, max $var[\boldsymbol{B}] = var[sgn\big((\boldsymbol{P}^v)^\top \psi(\boldsymbol{x}_i^v)\big)]$. This additional regularization on $\boldsymbol{B}$ can ensure the *balanced partition* and *reduce the redundancy* of the binary codes. Here we replace the sign function by its signed magnitude, and formulate the relaxed regularization as follows

$$\max \sum_k \mathbb{E}[\|(\boldsymbol{p}_i^v)^\top \psi(\boldsymbol{x}_i^v)\|^2] = \frac{1}{N} tr\big((\boldsymbol{P}^v)^\top \psi(\boldsymbol{X}^v)\psi(\boldsymbol{X}^v)^\top \boldsymbol{P}^v\big) = g(\boldsymbol{P}^v). \tag{3}$$

Finally, we combine problems (2) and (3) together and reformulate the overall common binary representation learning problem as the following

$$\min_{\boldsymbol{P}^v,\boldsymbol{B}} \sum_{v=1}^{m} (\alpha^v)^r \big(\|\boldsymbol{B} - (\boldsymbol{P}^v)^\top \psi(\boldsymbol{X}^v)\|_F^2 + \lambda_1 \|\boldsymbol{P}^v\|_F^2 - \lambda_2 g(\boldsymbol{P}^v)\big)$$
$$s.t.\ \sum_v \alpha^v = 1, \alpha^v > 0, \boldsymbol{B} = [\boldsymbol{B}_s; \boldsymbol{B}_I] \in \{-1,1\}^{K\times N}, \boldsymbol{P}^v = [\boldsymbol{P}_s, \boldsymbol{P}_I^v], \tag{4}$$

where λ_2 is a weighting parameter.

2) Robust Binary Cluster Structure Learning. For binary clustering, HSIC directly factorizes the learned binary representation $\boldsymbol{B}$ into the binary clustering centroids $\boldsymbol{Q}$ and discrete clustering indicators $\boldsymbol{F}$ using

$$\min_{\boldsymbol{Q},\boldsymbol{F}} \|\boldsymbol{B} - \boldsymbol{QF}\|_{21},\ s.t.\ \boldsymbol{Q1} = \boldsymbol{0}, \boldsymbol{Q} \in \{-1,1\}^{K\times c}, \boldsymbol{F} \in \{0,1\}^{c\times N}, \sum_j f_{ji} = 1, \tag{5}$$

where $\|\boldsymbol{A}\|_{21} = \sum_i \|\boldsymbol{a}^i\|_2$, and $\boldsymbol{a}^i$ is the i-th row of matrix $\boldsymbol{A}$. The first constraint of (5) ensures the balanced property on the clustering centroids as with the binary codes. Note that the ℓ_{21}-norm imposed on the loss function can also be replaced by the F-norm, *i.e.*, $\|\boldsymbol{B} - \boldsymbol{QF}\|_F^2$. However, the F-norm based loss function can amplify the errors induced from noise and outliers. Therefore, to achieve more stable and robust clustering performance, we employ the sparsity-induced ℓ_{21}-norm. It is also observed in [12] that the ℓ_{21}-norm not only preserves the rotation invariance within each feature, but also controls the reconstruction error, which significantly mitigates the negative influence of the representation outliers.

3) Joint Objective Function. To preserve the semantic interconnection between the learned binary codes and the robust cluster structures, we incorporate the common binary representation learning and the discrete cluster structure constructing into a joint learning framework. In this way, the unified framework can interactively enhance

Algorithm 1: Highly-economized Scalable Image Clustering (HSIC)

Input : Multi-view features $\{\boldsymbol{X}^v\}_{v=1}^m \in \Re^{d_v \times N}$, $m \geq 3$; code length K; number of centroids c; maximum iterations κ and t; λ_1, λ_2 and λ_3.
Output: Binary representation $\boldsymbol{B}$, cluster centroid $\boldsymbol{Q}$ and cluster indicator $\boldsymbol{F}$.
Initial. : Randomly select l anchor points from each view to calculate the kernelized feature embedding $\psi(\boldsymbol{X}^v) \in \Re^{l \times N}$, and normalize them to have zero-centered mean.
repeat
  $\boldsymbol{P}_S$**-Step**: Update $\boldsymbol{P}_S$ by Eqn.(8);
  $\boldsymbol{P}_I^v$**-Step**: Update $\boldsymbol{P}_I^v$ by Eqn. (9), $\forall v = 1, \cdots, m$;
  $\boldsymbol{B}$**-Step**: Update $\boldsymbol{B}$ by Eqn. (12);
  repeat
    $\boldsymbol{Q}$**-Step:** Iteratively update $\boldsymbol{Q}$ by Eqn. (14);
    $\boldsymbol{F}$**-Step:** Update $\boldsymbol{F}$ by Eqn. (16);
  until *convergence or reach* κ *iterations*;
  $\boldsymbol{\alpha}$**-Step**: Update $\boldsymbol{\alpha}$ by Eqn. (18);
until *convergence or reach* t *iterations*;

the qualities of the learned binary representation and cluster structures. Hence, we have the following joint objective function:

$$\begin{aligned}
&\min_{\boldsymbol{P}^v, \boldsymbol{B}, \boldsymbol{Q}, \boldsymbol{F}, \alpha^v} \sum_{v=1}^{m} (\alpha^v)^r \big(\|\boldsymbol{B} - (\boldsymbol{P}^v)^\top \psi(\boldsymbol{X}^v)\|_F^2 + \lambda_1 \|\boldsymbol{P}^v\|_F^2 - \lambda_2 g(\boldsymbol{P}^v) \big) + \lambda_3 \|\boldsymbol{B} - \boldsymbol{Q}\boldsymbol{F}\|_{21}, \\
&s.t. \sum_v \boldsymbol{\alpha}^v = 1, \boldsymbol{\alpha}^v > 0, \boldsymbol{B} = [\boldsymbol{B}_s; \boldsymbol{B}_I] \in \{-1, 1\}^{K \times N}, \boldsymbol{P}^v = [\boldsymbol{P}_s, \boldsymbol{P}_I^v], \\
&\quad \boldsymbol{Q}\mathbf{1} = \mathbf{0}, \boldsymbol{Q} \in \{-1, 1\}^{K \times c}, \boldsymbol{F} \in \{0, 1\}^{c \times N}, \sum_j f_{ji} = 1,
\end{aligned} \tag{6}$$

where λ_1, λ_2 and λ_3 are trade-off parameters to balance the effects of different terms. To optimize the difficult discrete programming problem, a newly-derived alternating optimization algorithm is developed as shown in the next section.

2.1 Optimization

The solution to problem (6) is non-trivial as it involves a mixed binary integer program with three discrete constraints, which lead to an NP-hard problem. In the following, we introduce an alternating optimization algorithm to iteratively update each variable while fixing others, *i.e.*, update $\boldsymbol{P}_s \to \boldsymbol{P}_I^v \to \boldsymbol{B} \to \boldsymbol{Q} \to \boldsymbol{F} \to \boldsymbol{\alpha}$ in each iteration.

Due to the intractable ℓ_{21}-norm loss function, we first rewrite the last term in (6) as $\lambda_3 tr\big(\boldsymbol{U}^\top \boldsymbol{D}\boldsymbol{U}\big)$, where $\boldsymbol{U} = \boldsymbol{B} - \boldsymbol{Q}\boldsymbol{F}$, and $\boldsymbol{D} \in \Re^{K \times K}$ is a diagonal matrix, the i-th diagonal element of which is defined as $\boldsymbol{d}_{ii} = 1/2\|\boldsymbol{u}^i\|$, where $\boldsymbol{u}^i$ is the i-th row of $\boldsymbol{U}$.
1) $\boldsymbol{P}_s$**-Step**: When fixing other variables, we update the sharable projection by

$$\min_{\boldsymbol{P}_s} \sum_{v=1}^{m} (\alpha^v)^r \big(\|\boldsymbol{B}_s - \boldsymbol{P}_s^\top \psi(\boldsymbol{X}^v)\|_F^2 + \lambda_1 \|\boldsymbol{P}_s\|_F^2 - \frac{\lambda_2}{N} tr\big(\boldsymbol{P}_s^\top \psi(\boldsymbol{X}^v) \psi^\top(\boldsymbol{X}^v) \boldsymbol{P}_s\big)\big). \tag{7}$$

For notational convenience, we rewrite $\psi(\boldsymbol{X}^v)\psi^\top(\boldsymbol{X}^v)$ as $\tilde{\boldsymbol{X}}$. Taking derivation of $\mathcal{L}$ with respect to $\boldsymbol{P}_s$ and let $\frac{\partial \mathcal{L}}{\partial \boldsymbol{P}_s} = 0$, we can obtain the closed-form solution of $\boldsymbol{P}_s$, *i.e.,*

$$\boldsymbol{P}_s = (\boldsymbol{A} + \lambda_1 \sum_{v=1}^{m} (\alpha^v)^r \boldsymbol{I})^{-1} \boldsymbol{T}\boldsymbol{B}^\top, \tag{8}$$

where $\boldsymbol{A} = (1 - \frac{\lambda_2}{N}) \sum_{v=1}^m (\alpha^v)^r \tilde{\boldsymbol{X}}$ and $\boldsymbol{T} = \sum_{v=1}^m (\alpha^v)^r \psi(\boldsymbol{X}^v)$.

2) $\boldsymbol{P}_I^v$-Step: Similarly, when fixing other parameters, the optimal solution of the v-th individual projection matrix can be determined by solving

$$\min_{\boldsymbol{P}_I^v} \|\boldsymbol{B}_I - (\boldsymbol{P}_I^v)^\top \psi(\boldsymbol{X}^v)\|_F^2 + \lambda_1 \|\boldsymbol{P}_I^v\|_F^2 - \frac{\lambda_2}{N} tr\big(\boldsymbol{P}_I^v \tilde{\boldsymbol{X}} (\boldsymbol{P}_I^v)^\top\big), \tag{9}$$

and its closed-form solution can be obtained by $\boldsymbol{P}_I^v = \boldsymbol{W}\psi(\boldsymbol{X}^v)\boldsymbol{B}^\top$, where $\boldsymbol{W} = \Big((1 - \frac{\lambda_2}{N})\tilde{\boldsymbol{X}} + \lambda_1 \boldsymbol{I}\Big)^{-1}$ can be calculated beforehand.

3) $\boldsymbol{B}$-Step: Problem (6) w.r.t. $\boldsymbol{B}$ can be rewritten as:

$$\min_{\boldsymbol{B}} \sum_{v=1}^{m} (\alpha^v)^r \big(\|\boldsymbol{B} - (\boldsymbol{P}^v)^\top \psi(\boldsymbol{X}^v)\|_F^2\big) + \lambda_3 tr\big(\boldsymbol{U}^\top \boldsymbol{D}\boldsymbol{U}\big),\ \ s.t.\ \boldsymbol{B} \in \{-1, 1\}^{K \times N}. \tag{10}$$

Since $\boldsymbol{B}$ only has '1' and '-1' entries and $\boldsymbol{D}$ is a diagonal matrix, both $tr(\boldsymbol{B}\boldsymbol{B}^\top) = tr(\boldsymbol{B}^\top\boldsymbol{B}) = KN$ and $tr\left(\boldsymbol{B}^\top \boldsymbol{D}\boldsymbol{B}\right) = N * tr(\boldsymbol{D})$ are constant terms w.r.t. $\boldsymbol{B}$. Based on this and with some further algebraic computations, (10) can be reformulated as

$$\min_{\boldsymbol{B}} -2tr\left[\boldsymbol{B}^\top \left(\textstyle\sum_{v=1}^m (\alpha^v)^r \big((\boldsymbol{P}^v)^\top \psi(\boldsymbol{X}^v)\big) + \lambda_3 \boldsymbol{Q}\boldsymbol{F}\right)\right] + const, s.t.\ \boldsymbol{B} \in \{-1, 1\}^{K \times N}, \tag{11}$$

where '*const*' denotes the constant terms. This problem has a closed-form solution:

$$\boldsymbol{B} = sgn\left(\sum_{v=1}^{m} (\boldsymbol{\alpha}^v)^r \big((\boldsymbol{P}^v)^\top \psi(\boldsymbol{X}^v)\big) + \lambda_3 \boldsymbol{Q}\boldsymbol{F}\right). \tag{12}$$

4) $\boldsymbol{Q}$-Step: First, we degenerate (6) into the following computationally feasible problem (by removing some irrelevant parameters and discarding the first constraint):

$$\min_{\boldsymbol{Q},\boldsymbol{F}} tr\big(\boldsymbol{U}^\top \boldsymbol{D}\boldsymbol{U}\big) + \nu \|\boldsymbol{Q}^\top \boldsymbol{1}\|_F^2,\ \ s.t.\ \boldsymbol{Q} \in \{-1, 1\}^{K \times c},\ \ \boldsymbol{F} \in \{0, 1\}^{c \times N}, \sum_j f_{ji} = 1. \tag{13}$$

With sufficiently large $\nu > 0$, problems (6) and (13) will be equivalent. Then, by fixing the variable $\boldsymbol{F}$, problem (13) becomes

$$\min_{\boldsymbol{Q}} \mathcal{L}(\boldsymbol{Q}) = -2tr(\boldsymbol{B}^\top \boldsymbol{D}\boldsymbol{Q}\boldsymbol{F}) + \nu \|\boldsymbol{Q}^\top \boldsymbol{1}\|_F^2 + const,\ \ s.t.\ \boldsymbol{Q} \in \{-1, 1\}^{K \times c}. \tag{14}$$

Inspired by the efficient discrete optimization algorithm in [35, 38], we develop an adaptive discrete proximal linearized optimization algorithm, which iteratively updates $\boldsymbol{Q}$ in the $(p{+}1)$-th iteration by $\boldsymbol{Q}^{p+1} = sgn(\boldsymbol{Q}^p - \frac{1}{\eta} \nabla \mathcal{L}(\boldsymbol{Q}^p))$, where $\nabla \mathcal{L}(\boldsymbol{Q})$ is the gradient of $\mathcal{L}(\boldsymbol{Q})$, $\frac{1}{\eta}$ is the learning step size and $\eta \in (C, 2C)$, where C is the Lipschitz constant. Intuitively, for the very special $sgn(\cdot)$ function, if the step size $1/\eta$ is too small/large, the solution of $\boldsymbol{Q}$ will get stuck in a bad local minimum or diverge. To this end, a proper η is adaptively determined by enlarging or reducing based on the changing values of $\mathcal{L}(\boldsymbol{Q})$ between adjacent iterations, which can accelerate its convergence.

5) $\boldsymbol{F}$-Step: Similarly, when fixing $\boldsymbol{Q}$, the problem w.r.t. $\boldsymbol{F}$ turns into

$$\min_{\boldsymbol{f}_i} \sum_{i=1}^{N} \boldsymbol{d}_{ii} \|\boldsymbol{b}_i - \boldsymbol{Q}\boldsymbol{f}_i\|_{21}, \ \ s.t. \ \boldsymbol{f}_i \in \{0,1\}^{c\times 1}, \sum_j f_{ji} = 1. \tag{15}$$

We can divide the above problem into N subproblems, and independently optimize the cluster indicator in a column-wise fashion. That is, one column of $\boldsymbol{F}$ (*i.e.*, $\boldsymbol{f}_i$) is computed at each time. Specifically, we solve the subproblems in an exhaustive search manner, similar to the conventional k-means algorithm. Regarding the i-th column $\boldsymbol{f}_i$, the optimal solution of its j-th entry can be efficiently obtained by

$$f_{ji} = \begin{cases} 1, & j = \arg\min_k H(\boldsymbol{d}_{ii} * \boldsymbol{b}_i, \boldsymbol{q}_{\wp}), \\ 0, & otherwise, \end{cases} \tag{16}$$

where $\boldsymbol{q}_{\wp}$ is the $\wp$-th vector of $\boldsymbol{Q}$, and $H(\cdot,\cdot)$ denotes the Hamming distance metric. Note that computing the Hamming distance is remarkably faster than the Euclidean distance, so the assigned vector $\boldsymbol{f}_i$ will efficiently constitute the matrix $\boldsymbol{F}$.
6) $\boldsymbol{\alpha}$-Step: Let $h^v = \|\boldsymbol{B} - (\boldsymbol{P}^v)^\top \phi(\boldsymbol{X}^v)\|_F^2 + \lambda_1\|\boldsymbol{P}^v\|_F^2 - \lambda_2 g(\boldsymbol{P}^v)$, then problem (6) w.r.t. $\boldsymbol{\alpha}$ can be rewritten as

$$\min_{\alpha^v} \sum_{v=1}^{m} (\alpha^v)^r h^v, \ \ s.t. \ \sum_v \alpha^v = 1, \alpha^v > 0. \tag{17}$$

The Lagrange function of (17) is $\min \mathcal{L}(\alpha^v, \boldsymbol{\zeta}) = \sum_{v=1}^m (\alpha^v)^r h^v - \boldsymbol{\zeta}(\sum_{v=1}^m \alpha^v - 1)$, where $\boldsymbol{\zeta}$ is the Lagrange multiplier. Taking the partial derivatives w.r.t. α^v and $\boldsymbol{\zeta}$, respectively, we can get

$$\begin{cases} \dfrac{\partial \mathcal{L}}{\partial \alpha^v} = & r(\alpha^v)^{r-1} h^v - \boldsymbol{\zeta}, \\ \dfrac{\partial \mathcal{L}}{\partial \boldsymbol{\zeta}} = & \displaystyle\sum_{v=1}^{m} \alpha^v - 1. \end{cases} \tag{18}$$

Following [47], by setting $\nabla_{\alpha^v,\boldsymbol{\zeta}} \mathcal{L} = \mathbf{0}$, the optimal solution of α^v is $\frac{(h^v)^{\frac{1}{1-r}}}{\sum_v (h^v)^{\frac{1}{1-r}}}$.

To obtain the locally optimal solution of problem (6), we update the above six variables iteratively until convergence. To deal with the out-of-example problem in image clustering, HSIC needs to generate the binary code for a new query image $\hat{\boldsymbol{x}}$ from the v-th view (*i.e.*, $\hat{\boldsymbol{x}}^v$) by $\boldsymbol{b}^v = sgn\left((\boldsymbol{P}^v)^\top \psi(\hat{\boldsymbol{x}}^v)\right)$, and then assigns it to the j-th cluster decided by $j = \arg\min_k H(\boldsymbol{b}^v, \boldsymbol{q}_k)$ in the fast Hamming space. For multi-view clustering, the common binary code of $\hat{\boldsymbol{x}}$ is $\boldsymbol{b} = sgn\left(\sum_{v=1}^m (\boldsymbol{\alpha}^v)^r (\boldsymbol{P}^v)^\top \psi(\hat{\boldsymbol{x}}^v)\right)$. Then the optimal cluster assignment of $\hat{\boldsymbol{x}}$ is determined by the solution of $\boldsymbol{F}$. The full learning procedure of HSIC is illustrated in Algorithm 1.

2.2 Complexity and Memory Load Analysis

1) The major computation burden of HSIC lies in the compressive binary representation learning and robust discrete cluster structures learning. The computational complexities of calculating $\boldsymbol{P}_S$ and $\boldsymbol{P}_I^v$ are $\mathcal{O}(K_S l N)$ and $\mathcal{O}(m(K_I l N))$, respectively. Computing $\boldsymbol{B}$ consumes $\mathcal{O}(KlN)$. Similar to [15], constructing the discrete cluster structures needs $\mathcal{O}(N)$ on bit-wise operators for κ iterations, where the distance computation requires only $\mathcal{O}(1)$ per time. The total computational complexity of HSIC is

$\mathcal{O}(t((K_S + mK_I + K)lN + \kappa N))$, where t and κ are empirically set to 10 in all the experiments. In general, the computational complexity of optimizing HSIC is linear to the number of samples, *i.e.*, $\mathcal{O}(N)$. **2)** For memory cost in HSIC, it is unavoidable to store the mapping matrices $\boldsymbol{P}_s$ and $\boldsymbol{P}_I^v$, demanding $\mathcal{O}(lK_S)$ and $\mathcal{O}(lK_I)$ memory costs, respectively. Notably, the learned binary representation and discrete cluster centroids only need the ***bit-wise*** memory load $\mathcal{O}(K(N + c))$, which is much less than that of k-means requiring $\mathcal{O}(d(N + c))$ ***real-valued*** numerical storage footprint.

3 Experimental Evaluation

In this section, we conducted multi-view image clustering experiments on four scalable image datasets to evaluate the effectiveness of HSIC with four frequently-used performance measures. All the experiments are implemented based on Matlab 2013a using a standard Windows PC with an Intel 3.4 GHz CPU.

3.1 Experimental Settings

Datasets and Features: We perform experiments on four image datasets, including **ILSVRC2012 1K** [11], **Cifar-10** [18], **YouTube Faces** (YTBF) [46] and **NUS-WIDE** [9]. Specifically, we randomly select 10 classes from ILSVRC2012 1K with $1,300$ images per class, denoted as *ImageNet-10*, for *middle-scale* multi-view clustering study. *Cifar-10* contains $60,000$ tiny color images in 10 classes, with $6,000$ images per class. A subset of *YTBF* contains $182,881$ face images from 89 different people ($> 1,200$ for each one). Similar to [38], we collect the subset of *NUS-WIDE* including the 21 most frequent concepts, resulting in $195,834$ images with at least $3,091$ images per category. Because some images in NUS-WIDE were labeled by multiple concepts, we only select one of the most representative labels as their true categories for simplicity. Multiple features are extracted on all datasets. Specifically, for ImageNet-10, Cifar-10 and YTBF, we use three different types of features, *i.e.*, 1450-d LBP, 1024-d GIST, and 1152-d HOG. For NUS-WIDE, five publicly available features are employed for experiments, *i.e.*, 64-d color Histogram (CH), 225-d color moments (CM), 144-d color correlation (CORR), 73-d edge distribution (EDH) and 128-d wavelet texture (WT).

Metrics and Parameters: We adopt four widely-used evaluation measures [28] for clustering, including clustering accuracy (ACC), normalized mutual information (NMI), purity, and F-score. In addition, both computational time and memory footprint are compared to show the efficiency of HSIC. To fairly compare different methods, we run the provided codes with default or fine-tuned parameter settings according to the original papers. For binary clustering methods, 128-bit code length is used for all datasets. For hyper-parameters λ_1, $\frac{\lambda_2}{N}$, and λ_3 of HSIC, we first employ the grid search strategy on ImageNet-10 to find the best values (*i.e.*, 10^{-3}, 10^{-3}, and 10^{-5}, respectively), which are then directly adopted on other datasets for simplicity. We empirically set r and $\delta = \frac{K_S}{K}$ (*i.e.*, the ratio of shared binary codes) as 5 and 0.2 respectively in all experiments. The multi-view clustering results are denoted as 'MulView'. We report the averaged clustering results with 10 times randomly initialization for each method.

We conduct the following experiments from ***three*** perspectives. ***Firstly***, we verify various characteristics of HSIC on the *middle-scale* dataset, *i.e.*, ImageNet-10. Here we compare HSIC with both SVIC and MVIC methods (including real-valued and binary ones). ***Secondly***, three large-scale datasets are exploited to evaluate HSIC on the

Table 1: Performance comparisons on ImageNet-10. The bold **black** and **blue** numbers indicate the best single-view and multi-view clustering results, respectively.

	Metric	ACC				NMI				Purity				F-Score			
	Feature	LBP	GIST	HOG	**MulView**	LBP	GIST	HOG	**MulView**	LBP	GIST	HOG	**MulView**	LBP	GIST	HOG	**MulView**
Single-view Alg.	k-means	0.2265	0.3085	0.2492	0.3073	0.1120	**0.1853**	0.1134	0.1803	**0.2361**	0.3098	0.2439	0.3133	0.1628	0.1970	0.1363	0.1996
	k-Medoids	0.1925	0.2634	0.2268	0.2605	0.0755	0.1721	0.1298	0.1461	0.1988	0.2852	0.2329	0.2690	0.1110	**0.1973**	0.1836	0.1874
	Ak-kmeans	0.2159	0.2988	0.2515	0.3113	0.1000	0.1541	0.1279	0.1966	0.2255	0.2805	0.2761	0.3254	0.1662	0.1827	0.1870	0.2122
	Nyström	0.2234	0.2459	0.2544	0.2950	0.0936	0.1222	0.1317	0.1719	0.2181	0.2585	0.2741	0.3320	0.1490	0.1749	0.1639	0.2050
	NMF	0.2178	0.2540	0.2509	0.2737	0.1076	0.1353	0.1434	0.1610	0.2178	0.2614	0.2705	0.2887	0.1571	0.1798	0.1609	0.1854
	LSC-K	**0.2585**	**0.3192**	0.2529	0.3284	0.1356	0.1806	0.1254	0.2215	0.2260	0.2660	0.2797	0.3447	**0.1748**	0.1748	0.1625	0.2301
Multi-view Alg.	AMGL	0.2093	0.2843	0.2516	0.2822	0.1131	0.1301	0.1368	0.2110	0.2149	0.3090	0.2796	0.2902	0.1311	0.1571	**0.2021**	0.2305
	MVKM	0.2321	0.2882	0.2535	0.3058	0.1181	0.1612	0.1372	0.1881	0.2115	0.3091	0.2538	0.3082	0.1461	0.1730	0.1861	0.2161
	MLAN	0.2109	0.2197	0.2127	0.3182	0.1173	0.1255	0.1152	0.1648	0.2117	0.2258	0.2168	0.3248	0.1403	0.1614	0.1813	0.1813
	MultiNMF	0.2113	0.2639	0.2574	0.2632	0.0986	0.1732	**0.1605**	0.1708	0.2202	0.2735	**0.2855**	0.2905	0.1531	0.1789	0.1802	0.1906
	OMVC	0.2062	0.2706	0.2544	0.2739	0.1196	0.1613	0.1222	0.1744	0.1925	0.2611	0.2592	0.2637	0.1333	0.1739	0.1761	0.1885
	MVSC	0.2248	0.2629	**0.2732**	0.3191	0.1293	0.1593	0.1294	0.2097	0.2132	0.3126	0.2828	0.3393	0.1481	0.1909	0.1911	0.2180
Binary Alg.	ITQ+bk-means [15]	0.1861	0.2923	0.2562	0.3101	0.0604	0.1746	0.1200	0.2304	0.1879	0.2842	0.2644	0.3168	0.1214	0.1954	0.1643	0.2032
	CKM [40]	0.1712	0.2382	0.1906	0.2794	0.0394	0.1352	0.0738	0.1823	0.1784	0.2556	0.1962	0.2844	0.1107	0.1687	0.1389	0.1990
	HSIC-TS	0.1829	0.3030	0.2523	0.3568	0.1367	0.1672	0.1013	0.2376	0.1935	0.3247	0.2577	0.3665	0.1194	0.1945	0.1525	0.2309
	HSIC-F	0.1951	0.2923	0.2516	0.3749	0.1289	0.1592	0.1015	0.2411	0.2062	0.3165	0.2625	0.3795	0.1252	0.1832	0.1566	0.2321
	HSIC(ours)	0.2275	0.3128	0.2597	**0.3865**	**0.1396**	0.1692	0.1219	**0.2515**	0.2131	**0.3253**	0.2723	**0.3905**	0.1353	0.1929	0.1739	**0.2530**

For all single-view methods, features from all views are simply concatenated to obtain the 'MulView' results.

Table 2: Time costs (in seconds) of different methods on ImageNet-10.

Alg.	k-means		Ak-kmeans		Nyström		LSC-K		AMGL		MLAN		OMVC		CKM		HSIC-TS		**HSIC (ours)**	
	Time	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup	Tim.	Speedup
LBP	69	1×	16	4.31×	15	4.60×	211	0.33×	1693	0.04×	1431	0.05×	696	0.10×	17	4.06×	18	3.83 ×	**4**	**17.25** ×
GIST	43	1×	11	3.91×	11	3.91×	226	0.19×	1730	0.03×	1557	0.03×	616	0.07×	11	3.91×	16	2.69 ×	**4**	**10.75** ×
HOG	82	1×	11	7.46×	12	6.83×	331	0.25×	1862	0.04×	2226	0.04×	643	0.13×	18	4.56×	16	5.13 ×	**3**	**27.33** ×
MulView	201	1×	21	9.57×	19	10.58×	503	0.40×	3820	0.05×	3336	0.06×	1109	0.18×	27	7.44×	20	10.05 ×	**5**	**40.20** ×

challenging large-scale MVIC problem. **Remark:** Based on the results on ImageNet-10 (see Table 2), the real-valued MVIC methods only obtain comparable results to k-means, but they are very time-consuming. Moreover, when applying those MVIC methods (*e.g.,* AMGL and MLAN) to larger datasets, we encounter the 'out-of-memory' error. Therefore, the real-valued MVIC methods are not compared on the three large-scale datasets. ***Thirdly***, some empirical analyses of our HSIC are also provided.

3.2 Experiments on the Middle-Scale ImageNet-10

We compare HSIC to several state-of-the-art clustering methods including SVIC methods (*i.e.,* k-means [16], k-Medoids [33], Approximate kernel k-means [8], Nyström [6], NMF [20], LSC-K [7]), MVIC methods, (*i.e.,* AMGL [31], MVKM [4], MLAN [30], MultiNMF [22], OMVC [37], MVSC [21]) and two existing binary clustering methods (*i.e.,* ITQ+bk-means [15] and CKM [41]). Additionally, two variants of HSIC are also compared to show its efficacy, *i.e.,* HSIC with F-norm regularized binary clustering (HSIC-F), and HSIC with two separate steps of binary code learning and discrete clustering (HSIC-TS). Similar to [21,22], for all the SVIC methods, we simply concatenate the feature vectors of all views for the 'MulView' clustering.

Table 1 demonstrates the performance of all clustering methods. From Table 1, we can observe in most cases that our HSIC can achieve comparable SVIC results but superior MVIC results in comparison with all the real-valued and binary clustering methods. This indicates the effectiveness of HSIC on the common representation learning and robust cluster structures learning, especially for the MVIC cases. Furthermore, it is clear that HSIC is superior to HSIC-F and HSIC-ST, which demonstrates the robustness and effectiveness of the joint learning framework.

The computational costs are illustrated in Table 2. From its last three columns, we can see that the binary clustering methods can reduce the computational time compared with the real-valued ones such as k-means and LSC-K, due to the highly efficient distance calculation in the Hamming space. Particularly, our HSIC is much faster than the compared real-valued and binary clustering methods, which also proves the superiority of the developed efficient optimization algorithm. Specifically, the speed-up of our HSIC for MVIC is very clear by a margin of 40.20 times compared to k-means. For memory footprint, k-means and our HSIC respectively require 361 MB and 2.73 MB, *i.e.*, ≈ 132 times memory can be reduced using HSIC.

Why does HSIC Outperform the Real-Valued Methods? Table 1 clearly shows that HSIC achieves competitive or superior clustering performance compared to the real-valued clustering methods. The favorable performance mainly comes from: **1)** HSIC greatly benefits from the proposed effective discrete optimization algorithm such that the learned binary representations can eliminate some redundant and noisy information in the original real-valued features. As can be seen in Fig. 2, the similarity structures of the same clusters are enhanced in the coding space, meanwhile, some disturbances from the original features are excluded to refine the learned representation. **2)** For image clustering, binary features are more robust to local changes since small variations caused by varying environments can be eliminated by quantized binary codes. **3)** HSIC is a unified interactive learning framework of the optimal binary codes and clustering structures, which is shown to be better than those disjoint learning approaches (*e.g.*, LSC-K, NMF, MVSC, AMGL and MLAN).

3.3 Experiments on Large-Scale Datasets

To show the strong scalability of HSIC on the large-scale MVIC problem, we compare HSIC with several state-of-the-art scalable clustering methods on three large-scale multi-view datasets. The clustering performance is summarized in Table 3. Given these results, we have the following observations: **1)** Generally, MVIC performs better than SVIC, which implies the necessity of incorporating complementary traits of multiple features for image clustering. Particularly, our HSIC achieves competitive or better SVIC results but consistent best MVIC performance. This mainly owes to the adaptive weights learning strategy and the exploiting of sharable and individual information

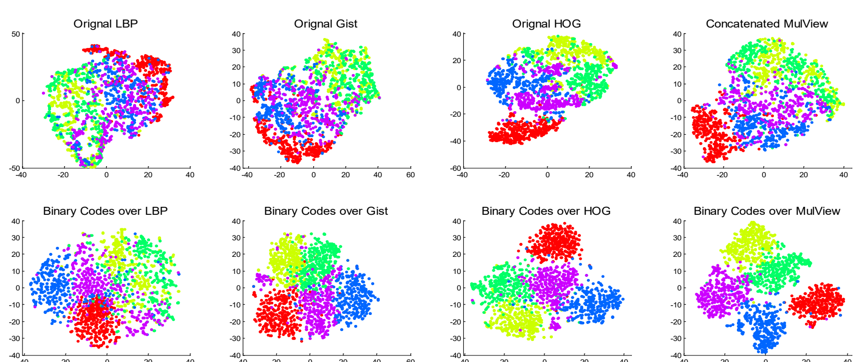


Fig. 2: t-SNE visualization of randomly selected 5 classes from ImageNet-10. The two rows show the real-valued features and 128-bit HSIC-based binary codes, respectively.

Table 3: Performance comparisons on the three large-scale datasets. The bold **black** and **blue** numbers indicate the best single-view and multi-view results, respectively.

Dataset	Metric	Alg.	k-means	k-mean++	k-Medoids	Ak-kmeans	LSC-K	Nyström	ITQ+ bk-means	CKM	HSIC-TS	HSIC-F	**HSIC**
Cifar-10	ACC	LBP	0.2185	0.2182	0.2171	0.2066	0.2550	0.2339	0.2322	0.2225	0.2440	0.2536	**0.2681**
		GIST	0.2842	0.2845	0.2419	0.2847	0.3010	0.2592	0.2777	0.2521	0.3209	0.3456	**0.3595**
		HOG	0.2661	0.2703	0.2456	0.2608	0.2838	0.2408	0.2481	0.2294	0.3178	0.3394	**0.3389**
		MulView	0.2877	0.2882	0.2630	0.2879	0.3488	0.2747	0.2787	0.2703	0.3742	0.3809	**0.3951**
	NMI	LBP	0.1044	0.1044	0.0862	0.1021	0.1303	0.0922	0.0963	0.1092	0.1105	0.1094	**0.1220**
		GIST	0.1692	0.1691	0.1238	0.1692	0.1869	0.1226	0.1502	0.1184	0.2063	0.2134	**0.2299**
		HOG	0.1634	0.1645	0.1328	0.1607	0.1668	0.1415	0.1570	0.1034	0.2053	**0.2199**	0.2170
		MulView	0.1803	0.1805	0.1565	0.1808	0.2382	0.1511	0.1613	0.1499	0.2547	0.2596	**0.2629**
	Purity	LBP	0.2401	0.2400	0.2339	0.2275	0.2768	0.2445	0.2490	0.2476	0.2526	0.2697	**0.2837**
		GIST	0.3056	0.3052	0.2483	0.3054	0.3306	0.2626	0.2882	0.2649	0.3650	0.3651	**0.3828**
		HOG	0.2943	0.2953	0.2561	0.2847	0.3039	0.2655	0.2756	0.2319	0.3199	**0.3589**	0.3481
		MulView	0.3136	0.3138	0.2921	0.3148	0.3787	0.2975	0.2953	0.2846	0.3956	0.4045	**0.4204**
	F-score	LBP	0.1677	0.1676	0.1703	0.1643	0.1692	0.1517	0.1685	0.1509	0.1717	0.1670	**0.1721**
		GIST	0.1866	0.1866	0.1744	0.1867	0.2044	0.1654	0.1808	0.1606	0.2318	0.2318	**0.2397**
		HOG	0.1887	0.1895	0.1808	0.1882	0.1878	0.1680	0.1769	0.1479	0.2221	0.2337	**0.2383**
		MulView	0.1998	0.2001	0.2035	0.2001	0.2477	0.1793	0.1863	0.1807	0.2422	0.2564	**0.2595**
YouTube-Faces (YTBF)	ACC	LBP	0.5870	0.5994	0.5262	0.5584	0.6017	0.5647	0.5765	0.5319	0.5930	0.6208	**0.6471**
		GIST	0.4081	0.4068	0.3584	0.2937	0.4638	0.4497	0.3547	0.3760	0.5432	0.6059	**0.6121**
		HOG	0.5751	0.5821	0.4810	0.5542	0.5830	0.5642	0.5574	0.5584	0.5436	**0.6133**	0.6099
		MulView	0.5927	0.6067	0.5290	0.5562	0.6099	0.6190	0.5852	0.5574	0.5974	0.6315	**0.6547**
	NMI	LBP	0.7473	0.7460	0.6835	0.7251	**0.7725**	0.7515	0.6870	0.6222	0.7256	0.7478	0.7690
		GIST	0.5528	0.5472	0.5062	0.4165	0.6237	0.6630	0.5146	0.5094	0.6889	0.7272	**0.7436**
		HOG	0.7442	0.7375	0.6640	0.7206	**0.7536**	0.7193	0.6827	0.6805	0.6965	0.7342	0.7483
		MulView	0.7492	0.7488	0.6774	0.7215	0.7515	0.7307	0.6921	0.6827	0.7579	0.7785	**0.7899**
	Purity	LBP	0.6744	0.6760	0.6033	0.6155	0.6782	0.6697	0.6529	0.5695	0.6597	0.6600	0.6915
		GIST	0.4641	0.4622	0.4315	0.3157	0.5366	0.5729	0.4405	0.4398	0.6099	0.6530	0.6766
		HOG	0.6499	0.6481	0.5733	0.6218	0.6602	0.6602	0.6257	0.6369	0.6105	0.6606	0.6682
		MulView	0.6712	0.6692	0.5969	0.6376	0.6687	0.6778	0.6642	0.6257	0.6615	0.6955	**0.7023**
	F-score	LBP	0.4240	0.4378	0.4034	0.4412	0.5058	0.4375	0.4421	0.4105	0.4286	0.4982	**0.5123**
		GIST	0.2567	0.2551	0.2310	0.1666	0.3390	0.3455	0.2308	0.2578	0.3367	0.4871	**0.4914**
		HOG	0.4813	0.4572	0.3715	0.4464	0.4627	0.3990	0.4303	0.4663	0.3379	0.4960	**0.5016**
		MulView	0.4886	0.4853	0.4236	0.4209	0.4650	0.4211	0.4650	0.4303	0.4517	0.5113	**0.5425**
NUS-WIDE	ACC	CH	0.1321	0.1370	**0.1433**	0.1351	0.1253	0.1391	0.1193	0.1244	0.1243	0.1314	0.1282
		CM	0.1334	**0.1379**	0.1305	0.1300	0.1297	0.1130	0.1123	0.1202	0.1346	0.1376	0.1360
		CORR	0.1352	**0.1358**	0.1222	0.1301	0.1344	0.1277	0.1143	0.1161	0.1349	0.1253	0.1279
		EDH	0.1402	**0.1425**	0.1382	0.1399	0.1266	0.1129	0.1180	0.1223	0.1343	0.1343	0.1396
		WT	0.1145	0.1182	0.1176	0.1169	0.1110	0.1226	0.1240	0.1172	0.1242	0.1147	**0.1293**
		MulView	0.1434	0.1458	0.1545	0.1499	0.1567	0.1452	0.1295	0.1296	0.1607	0.1639	**0.1661**
	NMI	CH	0.0687	0.0675	0.0706	0.0682	0.0638	0.0684	0.0629	0.0613	0.0668	0.0662	**0.0938**
		CM	0.0755	0.0687	0.0615	0.0747	0.0746	0.0656	0.0625	0.0580	0.0775	0.0870	**0.0944**
		CORR	0.0701	0.0699	0.0639	0.0714	0.0691	0.0661	0.0655	0.0589	0.0784	0.0652	**0.0882**
		EDH	0.0844	0.0877	0.0830	0.0900	0.0866	0.0707	0.0758	0.0731	**0.0961**	0.0872	0.0925
		WT	0.0571	0.0593	0.0559	0.0558	0.0661	0.0711	0.0632	0.0645	0.0878	0.0652	**0.0748**
		MulView	0.0944	0.0967	0.0823	0.0947	0.0980	0.0880	0.0773	0.0696	0.0937	0.0989	**0.1032**
	Purity	CH	0.2459	0.2418	0.2498	0.2439	0.2432	0.2443	0.2422	0.2390	0.2437	0.2397	**0.2589**
		CM	0.2453	0.2459	0.2284	0.2507	0.2516	0.2495	0.2433	0.2414	**0.2601**	0.2371	0.2515
		CORR	0.2370	0.2341	0.2402	0.2413	0.2408	0.2387	0.2404	0.2344	0.2564	0.2337	**0.2589**
		EDH	0.2388	0.2448	0.2365	0.2467	0.2393	0.2193	0.2354	0.2308	0.2451	0.2296	**0.2587**
		WT	0.2256	0.2274	0.2235	0.2237	0.2297	0.2328	0.2273	0.2256	0.2339	0.2306	**0.2393**
		MulView	0.2625	0.2634	0.2446	0.2711	0.2657	0.2546	0.2487	0.2413	0.2647	0.2653	**0.2753**
	F-score	CH	0.1128	0.1134	**0.1147**	0.1095	0.0946	0.1031	0.0867	0.0882	0.0863	0.0901	0.1009
		CM	0.1011	**0.1128**	0.0981	0.0956	0.0896	0.0867	0.0836	0.0879	0.0941	0.1095	0.1010
		CORR	0.1005	**0.1027**	0.0954	0.0947	0.0945	0.0969	0.0854	0.0841	0.0985	0.0888	0.0965
		EDH	**0.1163**	0.1150	0.1079	0.1149	0.0972	0.0865	0.0892	0.0899	0.0966	0.1130	0.1033
		WT	0.0933	0.0949	0.0940	0.0975	0.0893	0.0914	0.0889	0.0892	0.0903	0.0912	**0.1019**
		MulView	0.1106	0.1125	0.1105	0.1061	0.1071	0.1006	0.0905	0.0903	0.1076	0.1055	**0.1216**

For all single-view methods, features from all views are simply concatenated to obtain the 'MulView' results.

from heterogeneous features. **2)** From the last three columns of Table 3, we can observe that HSIC and its variants tend to be better than the real-valued ones. This shows that the binary codes learned by HISC are competitive to the real-valued ones. **3)** When comparing to HSIC-TS and HSIC-F, HSIC in most cases achieves superior

Table 4: Time costs (in seconds) on the three large-scale multi-view datasets.

	Alg.	k-means		k-means++		Ak-kmeans		LSC-K		Nyström		ITQ+bk-means		CKM		HSIC-TS		**HSIC (ours)**	
		Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup	Time	Speedup
Cifar-10	LBP	409	1×	294	1.39×	61	6.71×	112	3.65×	26	15.73×	24	17.04×	29	14.10×	29	14.10×	**10**	**40.90**×
	GIST	305	1×	334	0.91×	56	5.44×	834	0.37×	28	10.89×	23	13.26×	28	10.89×	30	10.17×	**10**	**30.50**×
	HOG	412	1×	266	1.55×	58	7.10×	913	0.45×	32	12.87×	27	15.26×	30	13.73×	25	16.48×	**10**	**41.20**×
	MulView	977	1×	791	1.23×	77	12.69×	1877	0.52×	58	16.85×	48	20.35×	46	21.24×	34	28.74×	**17**	**57.47**×
YTBF	LBP	2344	1×	1974	1.18×	533	4.40×	3546	0.66×	766	3.06×	90	26.04×	141	16.62×	97	24.17×	**40**	**58.60**×
	GIST	2299	1×	1705	1.34×	515	4.46×	3796	0.61×	828	2.78×	107	21.49×	153	15.03×	98	23.46×	**36**	**63.86**×
	HOG	3329	1×	1508	2.21×	523	6.37×	4042	0.83×	870	3.83×	104	32.01×	197	16.90×	105	31.71×	**48**	**69.35**×
	MulView	5879	1×	4250	1.38×	539	10.91×	12546	0.47×	998	5.89×	**110**	**53.45**×	309	19.03×	162	36.29×	139	42.30×
NUS-WIDE	CH	1027	1×	852	1.21×	464	2.21×	1693	0.61×	327	3.14×	91	11.29×	83	12.37×	85	12.08×	**34**	**30.21**×
	CM	1206	1×	937	1.29×	464	2.60×	1987	0.61×	352	3.43×	82	14.71×	93	12.97×	89	13.55×	**35**	**34.46**×
	CORR	1101	1×	876	1.26×	467	2.36×	1854	0.59×	382	2.88×	83	13.27×	83	13.26×	89	12.37×	**35**	**31.46**×
	EDH	1000	1×	829	1.21×	454	2.21×	1825	0.55×	371	2.70×	99	10.10×	91	10.99×	98	10.20×	**34**	**29.41**×
	WT	1206	1×	784	1.54×	491	2.46×	1984	0.61×	427	2.82×	82	14.71×	99	12.18×	81	14.89×	**34**	**35.47**×
	MulView	1711	1×	1147	1.49×	479	3.57×	8978	0.19×	485	3.53×	105	16.30×	142	12.05×	112	15.28×	**81**	**21.12**×

Table 5: Memory footprint of 'MulView' k-means and HSIC on the three large-scale datasets. 'Reduction' denotes the times of memory reduction against k-means.

Datasets	Memory w.r.t. k-means			Memory w.r.t. **HSIC (ours)**			
	Data (Real-valued features)	Centroids	Reduction	Data (128-bit binary codes)	Centroids	Projection	Reduction
Cifar-10 (60,000 images)	1.62GB	0.28MB	1×	0.92MB	0.15×10^{-3}MB	2.53MB	**481**×
YTBF (182,881 images)	4.94GB	2.46MB	1×	2.79MB	1.36×10^{-3}MB	2.53MB	**951**×
NUS-WIDE (195,834 images)	961MB	0.10MB	1×	2.99MB	0.32×10^{-3}MB	2.53MB	**174**×

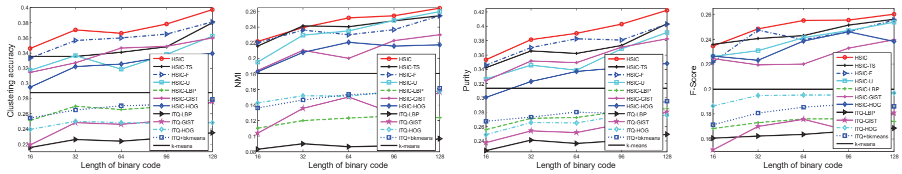
performance. This further reflects the advantages of the unified learning strategy and robust binary cluster structure construction.

The comparisons of running time and memory footprint are illustrated in Tables 4 and 5, respectively. From Table 4, we can observe that our HSIC is the fastest method in most cases. Table 5 shows that HSIC significantly reduces the memory load for large-scale MVIC compared to k-means. The memory cost of HSIC is similar to other binary clustering methods but clearly less than the real-valued methods. Moreover, as shown in Tables 4 and 5, for MVIC on NUS-WIDE with 5 views, HSIC can cluster near one million $(195,834 \times 5)$ features in 81 seconds using only 5.52 MB memory, while k-means needs about 29 minutes with 961 MB memory. Thus, HSIC can effectively address large-scale MVIC with much less computational time and memory footprint.

3.4 Empirical Analysis

Component Analysis: We evaluate the effectiveness of different components of HSIC in Fig. 3. Specifically, in addition to 'HSIC-TS' and 'HSIC-F', we have 'HSIC-U' by removing the balanced and independence constraints on binary codes and clustering centroids. HSIC-'view' and ITQ-'view' respectively refer to the SVIC results obtained using HSIC and ITQ+bk-means on the 'view'-specific features. From Fig. 3, we can observe that each component contributes essentially to the enhanced performance, and lacking any component will deteriorate the performance.

Effect of Code Length: We show our performance changes with the increasing code lengths in Fig. 3. In general, longer codes may provide more information for higher clustering performance. Specifically, both ITQ and HSIC based methods tend to achieve improved performance with increasing numbers of bits. Moreover, HSIC-based methods are superior to the baseline k-means when the code length is larger than 32. The best clustering results are established by HSIC w.r.t. different code lengths, because HSIC



(a) ACC (b) NMI (c) Purity (d) F-Score

Fig. 3: Performance of different clustering methods vs. code lengths on Cifar-10.

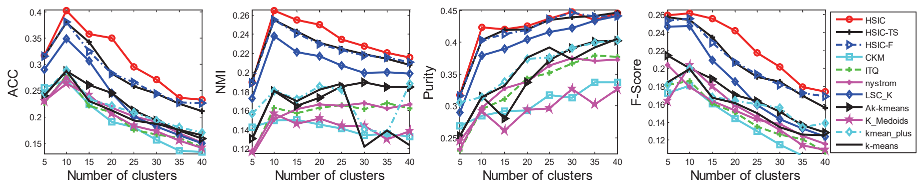


Fig. 4: Performance of different clustering methods vs. numbers of clusters on Cifar-10.

can effectively coordinate the importance of different views and mine the semantic correlations between them.

Effect of Number of Clusters: All the above experiments are evaluated based on the ground-truth cluster numbers. However, if the number of clusters is unknown, how will the performance change with different cluster numbers? To this end, we perform experiments on Cifar-10 to evaluate the stabilities of different methods w.r.t. number of clusters. Fig. 4 illustrates the performance changes by varying the cluster numbers from 5 to 40 with an interval of 5. Interestingly, the performance (*i.e.*, ACC, NMI and F-score) of HSIC-based methods increases when the cluster number increases from 5 to 10, but then sharply drops using more than 10 clusters. This suggests that 10 is the optimal number of clusters. Notably, 'purity' can not trade off the precise clustering evaluation against the number of clusters [28]. Importantly, the clustering performance of HSIC in most cases is better than all the compared methods, and HSIC-based methods hold the first three best results. This shows that HSIC is adaptive to different cluster numbers and can be potentially used to predict the 'optimal' number of clusters.

4 Conclusion

In this paper, we proposed a highly-economized multi-view clustering framework, dubbed HSIC, to jointly learn the compressive binary representations and robust discrete cluster structures. Specifically, HSIC collaboratively integrated the heterogeneous features into the common binary codes, where the sharable and individual information of multiple views were exploited. Meanwhile, a robust cluster structure learning model was developed to improve the clustering performance. Moreover, an effective alternating optimization algorithm was introduced to guarantee the high-quality discrete solutions. Extensive experiments on large-scale multi-view datasets demonstrate the superiority of HSIC over the state-of-the-art methods in terms of clustering performance with significantly reduced computational time and memory footprint.

References

1. Avrithis, Y., Kalantidis, Y., Anagnostopoulos, E., Emiris, I.Z.: Web-scale image clustering revisited. In: ICCV (2015)
2. Baluja, S., Covell, M.: Learning to hash: forgiving hash functions and applications. Data Mining and Knowledge Discovery **17**(3), 402–430 (Dec 2008)
3. Bickel, S., Scheffer, T.: Multi-view clustering. In: ICDM (2004)
4. Cai, X., Nie, F., Huang, H.: Multi-view k-means clustering on big data. In: IJCAI (2013)
5. Chen, J., Wang, Y., Qin, J., Liu, L., Shao, L.: Fast person re-identification via cross-camera semantic binary transformation. In: CVPR (2017)
6. Chen, W.Y., Song, Y., Bai, H., Lin, C.J., Chang, E.Y.: Parallel spectral clustering in distributed systems. IEEE TPAMI **33**(3), 568–586 (2011)
7. Chen, X., Cai, D.: Large scale spectral clustering with landmark-based representation. In: AAAI (2011)
8. Chitta, R., Jin, R., Havens, T.C., Jain, A.K.: Approximate kernel k-means: Solution to large scale kernel clustering. In: SIGKDD (2011)
9. Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: A real-world web image database from national university of singapore. In: ACM International Conference on Image and Video Retrieval (2009)
10. De Sa, V.R., Gallagher, P.W., Lewis, J.M., Malave, V.L.: Multi-view kernel construction. Mach. Learn. **79**(1-2), 47–71 (2010)
11. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Li, F.F.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)
12. Ding, C., Zhou, D., He, X., Zha, H.: R1-pca: rotational invariant $\ell 1$-norm principal component analysis for robust subspace factorization. In: ICML (2006)
13. Gao, H., Nie, F., Li, X., Huang, H.: Multi-view subspace clustering. In: ICCV (2015)
14. Gong, Y., Lazebnik, S., Gordo, A., Perronnin, F.: Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. IEEE TPAMI **35**(12), 2916–2929 (2013)
15. Gong, Y., Pawlowski, M., Yang, F., Brandy, L., Bourdev, L., Fergus, R.: Web scale photo hash clustering on a single machine. In: CVPR (2015)
16. Hartigan, J.A., Wong, M.A.: Algorithm as 136: A k-means clustering algorithm. J. R. Stat. Soc., Series C **28**(1), 100–108 (1979)
17. Jain, A.K.: Data clustering: 50 years beyond k-means. PRL **31**(8), 651–666 (2010)
18. Krizhevsky, A., Hinton, G.: Learning multiple layers of features from tiny images. Technical Report (2009)
19. Kumar, A., Rai, P., Daume, H.: Co-regularized multi-view spectral clustering. In: NIPS (2011)
20. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: NIPS (2001)
21. Li, Y., Nie, F., Huang, H., Huang, J.: Large-scale multi-view spectral clustering via bipartite graph. In: AAAI (2015)
22. Liu, J., Wang, C., Gao, J., Han, J.: Multi-view clustering via joint nonnegative matrix factorization. In: ICDM (2013)
23. Liu, L., Shao, L.: Sequential compact code learning for unsupervised image hashing. IEEE TNNLS **27**(12), 2526–2536 (2016)
24. Liu, L., Yu, M., Shao, L.: Latent structure preserving hashing. IJCV **122**(3), 439–457 (2017)

25. Liu, W., Mu, C., Kumar, S., Chang, S.F.: Discrete graph hashing. In: NIPS (2014)
26. Liu, W., Wang, J., Kumar, S., Chang, S.F.: Hashing with graphs. In: ICML (2011)
27. Lu, J., Liong, V.E., Zhou, J.: Simultaneous local binary feature learning and encoding for homogeneous and heterogeneous face recognition. IEEE TPAMI (2017)
28. Manning, C.D., Raghavan, P., Schütze, H., et al.: Introduction to information retrieval, vol. 1. Cambridge university press Cambridge (2008)
29. Ng, A.Y., Jordan, M.I., Weiss, Y.: On spectral clustering: Analysis and an algorithm. In: NIPS (2002)
30. Nie, F., Cai, G., Li, X.: Multi-view clustering and semi-supervised classification with adaptive neighbours. In: AAAI (2017)
31. Nie, F., Li, J., Li, X., et al.: Parameter-free auto-weighted multiple graph learning: A framework for multiview clustering and semi-supervised classification. In: IJCAI (2016)
32. Otto, C., Wang, D., Jain, A.K.: Clustering millions of faces by identity. IEEE TPAMI **40**(2), 289–303 (2018)
33. Park, H.S., Jun, C.H.: A simple and fast algorithm for k-medoids clustering. Expert Syst. Appl. **36**(2), 3336–3341 (2009)
34. Qin, J., Liu, L., Shao, L., Ni, B., Chen, C., Shen, F., Wang, Y.: Binary coding for partial action analysis with limited observation ratios. In: CVPR (2017)
35. Qin, J., Liu, L., Shao, L., Shen, F., Ni, B., Chen, J., Wang, Y.: Zero-shot action recognition with error-correcting output codes. In: CVPR (2017)
36. Sculley, D.: Web-scale k-means clustering. In: WWW (2010)
37. Shao, W., He, L., Lu, C.t., Philip, S.Y.: Online multi-view clustering with incomplete views. In: ICBD (2016)
38. Shen, F., Zhou, X., Yang, Y., Song, J., Shen, H.T., Tao, D.: A fast optimization method for general binary code learning. IEEE TIP **25**(12), 5610–5621 (2016)
39. Shen, F., Mu, Y., Yang, Y., Liu, W., Liu, L., Song, J., Shen, H.T.: Classification by retrieval: binarizing data and classifier. In: ACM SIGIR (2017)
40. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised discrete hashing. In: CVPR (2015)
41. Shen, X.B., Liu, W., Tsang, I.W., Shen, F., Sun, Q.S.: Compressed k-means for large-scale clustering. In: AAAI (2017)
42. Tzortzis, G., Likas, A.: Kernel-based weighted multi-view clustering. In: ICDM (2012)
43. Wang, J., Zhang, T., Sebe, N., Shen, H.T., et al.: A survey on learning to hash. IEEE TPAMI (2017)
44. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for scalable image retrieval. In: CVPR (2010)
45. Wang, X., Guo, X., Lei, Z., Zhang, C., Li, S.Z.: Exclusivity-consistency regularized multi-view subspace clustering. In: CVPR (2017)
46. Wolf, L., Hassner, T., Maoz, I.: Face recognition in unconstrained videos with matched background similarity. In: CVPR (2011)
47. Xia, T., Tao, D., Mei, T., Zhang, Y.: Multiview spectral embedding. IEEE TCYB **40**(6), 1438–1446 (2010)
48. Xu, C., Tao, D., Xu, C.: A survey on multi-view learning. arXiv preprint (2013)
49. Zhang, C., Hu, Q., Fu, H., Zhu, P., Cao, X.: Latent multi-view subspace clustering. In: CVPR (2017)
50. Zhang, Z., Liu, L., Shen, F., Shen, H.T., Shao, L.: Binary multi-view clustering. IEEE TPAMI (2018)
51. Zhang, Z., Shao, L., Xu, Y., Liu, L., Yang, J.: Marginal representation learning with graph structure self-adaptation. IEEE TNNLS (2018)