Deep Collaborative Multi-View Hashing for Large-Scale Image Search

Lei Zhu, Xu Lu, Zhiyong Cheng, Jingjing Li, and Huaxiang Zhang

***Abstract*— Hashing could significantly accelerate large-scale image search by transforming the high-dimensional features into binary Hamming space, where efficient similarity search can be achieved with very fast Hamming distance computation and extremely low storage cost. As an important branch of hashing methods, multi-view hashing takes advantages of multiple features from different views for binary hash learning. However, existing multi-view hashing methods are either based on shallow models which fail to fully capture the intrinsic correlations of heterogeneous views, or unsupervised deep models which suffer from insufficient semantics and cannot effectively exploit the complementarity of view features. In this paper, we propose a novel *Deep Collaborative Multi-view Hashing* (DCMVH) method to deeply fuse multi-view features and learn multi-view hash codes collaboratively under a deep architecture. DCMVH is a new deep multi-view hash learning framework. It mainly consists of 1) multiple view-specific networks to extract hidden representations of different views, and 2) a fusion network to learn multi-view fused hash code. DCMVH associates different layers with instance-wise and pair-wise semantic labels respectively. In this way, the discriminative capability of representation layers can be progressively enhanced and meanwhile the complementarity of different view features can be exploited effectively. Finally, we develop a fast discrete hash optimization method based on augmented Lagrangian multiplier to efficiently solve the binary hash codes. Experiments on public multi-view image search datasets demonstrate our approach achieves substantial performance improvement over state-of-the-art methods.**

***Index Terms*— Deep multi-view hashing, image search, efficient discrete optimization.**

I. Introduction

HASHING [1] could project the high-dimensional data into low-dimensional binary hash codes. With hashing, large-scale image search can be significantly accelerated by fast Hamming distance computation with low storage cost. Due to its desirable advantages, hashing has been widely applied in image search [2], classification [3], recommendation [4], action analysis [5], and etc.

Currently, most researchers contribute their efforts on single-view hashing [6]–[9], which adopts only one view of feature descriptor for hash learning. In practical image search, it is common that images are represented by several features in multiple views, which characterize the images from different aspects and possess their own characteristics. To handle multi-view features, cross-view hashing [10]–[13] is proposed to learn shared hash codes for cross-view search. At the online search stage, the images in one view are adopted as the queries to retrieve the database images in the other view. However, when both the query and database images are described with multi-view features, existing single-view and cross-view hashing methods cannot work anymore.

Recently, multi-view hashing [14]–[18] is investigated by combining multi-view features at both offline hash code learning and online search stage. It is different from single-view and cross-view hashing methods where only one of the view features is provided as query. Intuitively, single-view hashing can be easily extended to handle multi-view features by concatenating them into a unified feature vector. However, this simple strategy cannot sufficiently exploit the complementarity of multi-view features and fail to alleviate the semantic gap among them.

Many multi-view hashing methods have been proposed in literature. For example, Multiple Feature Hashing (MFH) [19], Compact Kernel Hashing with Multiple Features (MFKH) [15], Multiview Alignment Hashing (MAH) [20], Multi-view Latent Hashing (MVLH) [14], Multi-view Discrete Hashing (MvDH) [21], Discrete Multi-view Hashing (DMVH) [22]. Despite their success, they still suffer from two important problems:

A. Limited Semantic Representation Capability

Most existing multi-view hashing methods are still based on shallow models. Intrinsically, raw features are low-level representation on representing the semantics, and there also exists heterogeneous semantic gap between different view features. Hence, these shallow hashing models cannot fully capture the high-level semantics and the intrinsic correlations of heterogeneous view features [23]. Although deep learning has already significantly reshaped the research fields of single-view and cross-view hashing [11], [24], it is still a challenging problem that how to learn discriminative binary hash codes by combing heterogeneous multi-view data with nowadays advanced deep learning framework.

Manuscript received May 29, 2019; revised November 13, 2019 and January 6, 2020; accepted February 5, 2020. Date of publication February 21, 2020; date of current version February 27, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61802236, Grant 61772322, and Grant U1836216, in part by the Natural Science Foundation of Shandong, China, under Grant ZR2019QF002, in part by the Major Fundamental Research Project of Shandong, China, under Grant ZR2019ZD03, in part by the Youth Innovation Project of Shandong Universities, China, under Grant 2019KJN040, and in part by the Taishan Scholar Project of Shandong, China, under Grant ts20190924. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Yannick Berthoumieu. *(Corresponding author: Huaxiang Zhang.)*

Lei Zhu, Xu Lu, and Huaxiang Zhang are with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China (e-mail: huaxzhang@hotmail.com).

Zhiyong Cheng is with the Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China.

Jingjing Li is with the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China.

Digital Object Identifier 10.1109/TIP.2020.2974065

1057-7149 © 2020 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

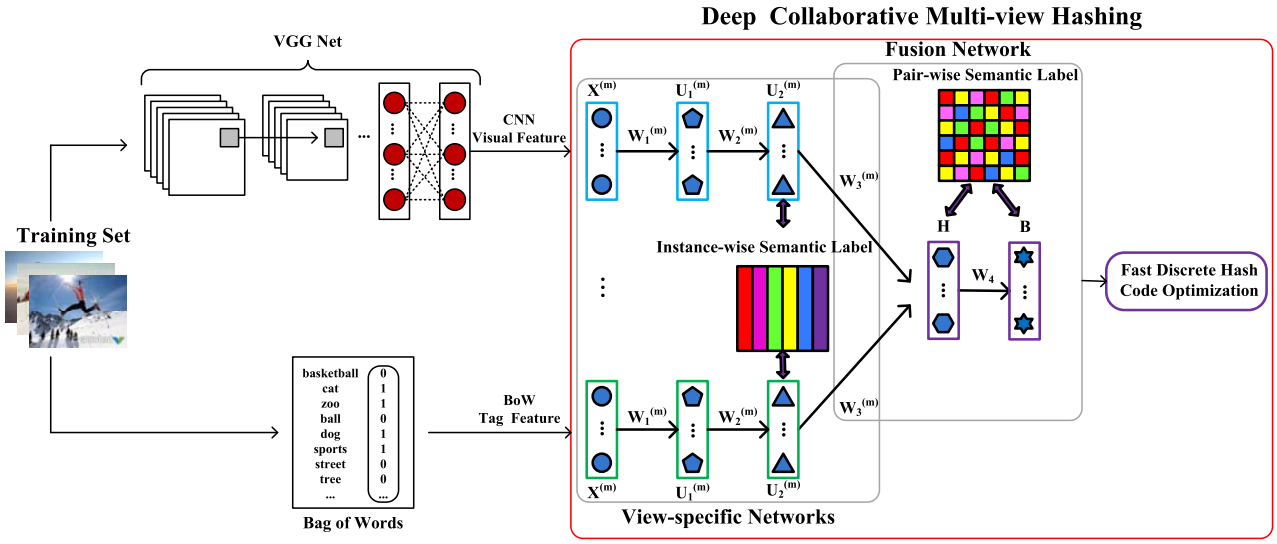

Fig. 1. The basic framework of the proposed DCMVH. We firstly obtain CNN visual feature with VGG Net and BoW tag feature from images and texts respectively. Then, we import them into the DCMVH model. DCMVH deeply fuses multi-view features and learns hash codes in a collaborative manner. DCMVH is comprised of 1) multiple *view-specific networks* for different views to extract hidden representations and 2) a *fusion network* to fuse them into a consensus multi-view factor **H**, based on which the multi-view hash code **B** is learned. The 1st layer is a feature selection layer to filter the feature noises from raw data. The 2nd layer is associated with the instance-wise semantic label to enhance its discriminative capability. The 3rd layer seeks a common latent subspace from multiple view-specific networks and learn the consensus multi-view factor. The 4th layer transforms the consensus multi-view factor to multi-view hash code and correlates it with the pair-wise semantic matrix. The last layer outputs the binary hash codes. Furthermore, a fast discrete hash code optimization module is designed to directly learn the binary multi-view hash codes.

To the best of our knowledge, only two deep multi-view hashing methods are proposed in literature: Deep Multi-View Hashing (DMVH-4layers) [16] and Multi-view Hashing with Orthogonal Regularization (DMHOR) [25]. These two methods are all unsupervised. As a result, the learned hash codes still suffer from limited semantics. In addition, they cannot sufficiently exploit the complementarity of different view features with unsupervised learning [26], [27].

B. Relaxed or Time-Consuming Discrete Optimization

Discrete optimization has achieved impressive results in single-view [28], [29] and cross-view hashing [30], [31]. However, its development in multi-view hashing is severely lagging behind. Existing multi-view hashing adopt two kinds of optimization strategies. The first is two-step relaxing+rounding relaxed optimization [19], [20], [25]–[27], [32], which relaxes the discrete constraints first and then obtains binary codes by simple thresholding. As indicated by recent works [14], [21], this simplified hash optimization strategy could cause significant quantization errors and thus lead to sub-optimal solutions. The other one is the bit-by-bit hash optimization based on Discrete Cyclic Coordinate Descent (DCC) [14], [22]. This strategy is still time-consuming since that only one bit is optimized in one step and learning all hash bits needs lots of iterations.

In this paper, we propose a *Deep Collaborative Multi-view Hashing* (DCMVH) method to address the above problems in a novel supervised deep learning model. Our method deeply fuses multi-view features and learns hash codes in a collaborative manner. Besides, the representation learning process is supervised by dual-level semantics, which progressively enhances the discriminative capability of the binary hash codes. Moreover, we propose a fast discrete hash optimization method to directly learn binary hash codes with avoiding the quantization errors. The basic framework of the proposed DCMVH is illustrated in Fig. 1. We firstly obtain CNN visual feature with VGG Net [33] and Bag-of-Words (BoW) [34] tag feature from images and texts respectively. Then, we import them into the DCMVH model. In our model, multiple *view-specific networks* for different views are developed to extract hidden representations, where the 1st layer is a feature selection layer to filter the feature noises from raw data, the 2nd layer is associated with the instance-wise semantic label to enhance its discriminative capability, and the 3rd layer seeks a common latent subspace from multiple view-specific networks and learn the consensus multi-view factor. Then, a *fusion network* aims to fuse them into a consensus multi-view factor, based on which the multi-view hash code is learned. In this network, the 4th layer transforms the consensus multi-view factor to multi-view hash code and correlates it with the pair-wise semantic matrix, and the last layer outputs the binary hash codes. Furthermore, a fast discrete hash code optimization module is designed to directly learn the binary multi-view hash codes. The main contributions of this paper are:

- We propose a novel deep multi-view hashing architecture to deeply fuse multi-view features and collaboratively learn binary hash codes. Our model exploits the complementarity of multi-view features with discriminative semantic label supervision. To the best of our knowledge, it is the first supervised deep multi-view hashing model.
- In our model, the representation layers first filter the feature noises, and are progressively supervised by dual-level semantic labels. The learned hash codes can not only capture the low-level multi-view feature distributions, but also be correlated with high-level semantics to gradually enhance the semantic representation capability.
- A fast discrete optimization method based on the augmented Lagrangian multiplier is proposed to directly and

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

efficiently solve the binary hash codes with a single hash code-solving operation. Experiments demonstrate the substantial performance gains of the proposed method over state-of-the-art techniques.

The rest of this paper is arranged as follows. Section II reviews the related work of existing hashing methods. The details of the proposed method are introduced in Section III. Section IV presents the experiments. Finally, Section V concludes the paper.

II. Related Work

According to the view features employed in hash learning and online search, existing hashing methods are categorized into three major families: single-view hashing [6], [9], [35], cross-view hashing [10], [11], [13], and multi-view hashing [14]–[16].

A. Single-View Hashing

Single-view hashing [6], [7], [9], [28], [29], [35]–[38] is designed to generate compact binary codes for single-view data represented by one view of features. Iterative Quantization (ITQ) [6] is a typical method of this kind. It decreases the quantization error by finding an orthogonal rotation to project training data. Supervised Discrete Hashing (SDH) [7] is a supervised hashing method, which employs DCC to solve the discrete optimization without any relaxations. Fast Supervised Discrete Hashing (FSDH) [28] directly regresses the class labels of training images to the corresponding hash code to accelerate the hash learning process. Scalable Supervised Discrete Hashing (SSDH) [35] efficiently learns the discrete hash codes by making full use of training images and semantic information. Attribute Hashing (AH) [36] is a supervised single-view hashing method to facilitate zero-shot image retrieval. It is a multi-layer hash learning method to model the relationships among visual features, binary codes and labels. Similarity-Adaptive Deep Hashing (SADH) [37] is a unsupervised hashing framework within the deep architecture. Different from many two-step hashing methods, SADH uses the output of the learned deep model to update the similarity graph matrix, so as to improve the subsequent hash code optimization.

B. Cross-View Hashing

Cross-view hashing [10], [11], [13], [39]–[43] retrieves the most relevant objects represented by one view for a given query represented by the other view. Collective Matrix Factorization Hashing (CMFH) [39] is an unsupervised cross-view hashing method, which learns the unified hash codes by collective matrix factorization from different views of one instance. Semantic Correlation Maximization (SCM) [40] is a supervised cross-view hashing method, which avoids explicitly computing the pairwise similarity matrix and utilizes all the supervised information for training with linear-time complexity. Semantics-Preserving Hashing (SePH) [41] transforms the supervised semantic affinities into a probability distribution and approximates it with to-be-learnt hash codes in Hamming space by minimizing the Kullback-Leibler divergence. Label Consistent Matrix Factorization Hashing (LCMFH) [10] is a supervised cross-view hashing method. It transforms heterogeneous data into the latent spaces where the data images from the same category share the same representation. Collective Reconstructive Embeddings (CRE) [11] maps heterogeneous features into a common Hamming space, where the cross-view similarities can be approximately measured with the Hamming distance. Specifically, to address the heterogeneity challenge, CRE exploits domain-specific methods to model different views. Self-supervised Adversarial Hashing (SSAH) [13] incorporates adversarial learning [44] into cross-view hashing in a self-supervised fashion. It leverages two adversarial networks to maximize the semantic correlation of different views, and a self-supervised semantic network to discover high-level semantic information with multi-label annotations.

The main objective of cross-view hashing is to discover the shared Hamming space between different views so that the search process across different views can be achieved accordingly. Similar to single-view hashing, only one view of feature is provided during the online search stage. However, existing cross-view hashing methods cannot work when the queries are described by multi-view features.

C. Multi-View Hashing

Multi-view hashing [14]–[16], [19]–[22], [32] fuses multi-view features for hash learning. Most existing multi-view hashing methods adopt unsupervised learning to learn hash codes. They usually construct a graph to model the relationships between images, and then perform the subsequent hash learning. Multiple Feature Hashing (MFH) [19] not only preserves the local structure information of each view, but also considers all the local structures during the optimization. Multi-view Alignment Hashing (MAH) [20] formulates a multi-graph regularized nonnegative matrix factorization framework, where hash codes are learned by uncovering the hidden semantics and capturing the joint probability distribution of data. Multi-view Discrete Hashing (MvDH) [21] performs spectral clustering to learn cluster labels and applies them in the hash code learning process to generate discriminant hash codes. Multi-view Latent Hashing (MVLH) [14] learns multi-view shared hash codes from an unified kernel feature space. The aforementioned unsupervised multi-view hashing methods exploit no semantic label, thus the learned hash codes suffer from limited discriminative capability.

Supervised multi-view hashing employs semantic labels as supervision in the hash learning process. Compact Kernel Hashing with Multiple Features (MFKH) [15] preserves the semantic similarity between different images and learns the hash codes with optimal linearly-combined multiple kernels. Discrete Multi-view Hashing (DMVH) [22] is a discrete supervised multi-view hashing method. It constructs similarity graph based on Locally Linear Embedding (LLE) [45], [46], which cannot only preserve local similarity structure, but also keep the semantic similarity between data pairs.

The aforementioned multi-view hashing methods are all shallow models. They may fail to capture the high-level semantics and the intrinsic complex correlations of heterogeneous views. Only a few multi-view deep hashing

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

TABLE I
KEY CHARACTERISTICS OF REPRESENTATIVE MULTI-VIEW HASHING METHODS AND THE PROPOSED DCMVH

Methods	Model	Learning	Optimization
MFKH [15]	shallow	supervised	relaxation
DMVH-4layers [16]	deep	unsupervised	relaxation
MFH [19]	shallow	unsupervised	relaxation
MVAGH [32]	shallow	unsupervised	relaxation
MAH [20]	shallow	unsupervised	relaxation
DMHOR [25]	deep	unsupervised	relaxation
MVLH [14]	shallow	unsupervised	discrete
DMVH [22]	shallow	supervised	discrete
MvDH [21]	shallow	unsupervised	discrete
Ours	**deep**	**supervised**	**discrete**

methods are developed in literature. Deep Multi-View Hashing (DMVH-4layers) [16] exploits a deep network with a set of view-specific hidden nodes and a set of shared hidden nodes to yield binary codes at its top layer nodes. Deep Multi-view Hashing with Orthogonal Regularization (DMHOR) [25] exploits the intra-view and inter-view correlation to learn hash codes. It specially reduces the redundant information among different views by imposing the orthogonal regularizer on the weighting matrices of the model. These two methods are all unsupervised, and they simply adopt relaxed strategy to solve the hash codes. As illustrated above, they will suffer from the problems of limited discriminative capability and significant quantization loss. Different from them, in this paper, we propose a novel *Deep Collaborative Multi-view Hashing* (DCMVH), which directly learns discrete hash codes progressively in a deep architecture with discriminative dual-level semantic supervision. The key characteristics of representative hashing methods and the proposed method are summarized in Table I.

III. THE PROPOSED METHODOLOGY

A. Notations

Throughout this paper, we utilize bold uppercase letters to represent matrices and bold lowercase letters to represent vectors. For any matrix $\mathbf{A}$, $\mathbf{a}_{i\cdot}$ means the i-th row vector of $\mathbf{A}$, $\mathbf{a}_{\cdot j}$ means the j-th column vector of $\mathbf{A}$, A_{ij} denotes the (i, j)-element of $\mathbf{A}$ and $\mathtt{tr}(\mathbf{A})$ is the trace of $\mathbf{A}$ if $\mathbf{A}$ is square. $\mathbf{A}^{\mathrm{T}}$ denotes the transposed matrix of $\mathbf{A}$. The Frobenius norm of a matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$ is defined as $\|\mathbf{A}\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n} A_{ij}^2 = \mathtt{tr}(\mathbf{A}^{\mathrm{T}}\mathbf{A})$. The $\ell_{2,1}$-norm for $\mathbf{A}$ is defined as $\|\mathbf{A}\|_{2,1} = \sum_{i=1}^{m} \|\mathbf{a}_{i\cdot}\|_2$.

Suppose that $\mathbf{O} = \{\mathbf{o}_i\}_{i=1}^{n}$ is the training set, which contains n training image instances. Each image instance is represented with V different view features. The v-th view feature is $\{\mathbf{X}^{(v)} = [\mathbf{x}_1^{(v)}, \ldots, \mathbf{x}_n^{(v)}] \in \mathbb{R}^{p_v\times n}\}_{v=1}^{V}$, where p_v is the dimensionality of the v-th view. V is the number of views. Different views of one image instance $\mathbf{o}_i$ belong to the same category. The category label of each image instance $\mathbf{y}_i \in \{0, 1\}^c$ is given and $\mathbf{Y} \in \mathbb{R}^{c\times n}$ is the instance-wise semantic label, where c is the number of categories. $\mathbf{S} \in \{-1, 1\}^{n\times n}$ is a pair-wise semantic matrix, where the element in the i-th row and j-th column is defined as

$$s_{ij} = \begin{cases} 1, & \text{if } \mathbf{x}_i \text{ and } \mathbf{x}_j \text{ belong to the same category,} \\ -1, & \text{vice versa.} \end{cases}$$

Our method is to learn hash code $\mathbf{B} \in \{-1, 1\}^{r\times n}$ to represent multi-view image instances, where r is the length of hash code. The important notations in this paper are summarized in Table II.

B. Deep Multi-View Hash Learning Model

Meanwhile, different views have complex internal relationships to each other, and there exists heterogeneous semantic gap between different views [47]–[50]. To capture these characteristics, in this paper, we develop a novel deep hash learning model to deeply fuse different view features and collaboratively learn multi-view hash codes. Our deep architecture consists of 1) *multiple view-specific networks* to extract hidden representations, and 2) *a fusion network* to fuse them into a consensus multi-view factor $\mathbf{H} \in \mathbb{R}^{r\times n}$, based on which the multi-view hash code $\mathbf{B} \in \{-1, 1\}^{r\times n}$ is learned. In the deep architecture, each representation layer has a different function.

- Feature selection layer. The real data inevitably contains noises. To alleviate the adverse effects of those noises, we design the 1st layer to filter the feature noises from raw data. Formally, it works as $\mathbf{U}_1^{(v)} = \mathbf{W}_1^{(v)}\mathbf{X}^{(v)}$, where $\mathbf{U}_1^{(v)} \in \mathbb{R}^{d_1\times n}$ is the output of the 1st layer, $\mathbf{W}_1^{(v)} \in \mathbb{R}^{d_1\times p_v}$ is the corresponding feature selection matrix of the 1st layer in the v-th view-specific network.
- The 2nd layers are represented as $\mathbf{U}_2^{(v)} = \mathbf{W}_2^{(v)}\mathbf{U}_1^{(v)}$, where $\mathbf{U}_2^{(v)} \in \mathbb{R}^{c\times n}$ is the output of the 2nd layer, $\mathbf{W}_2^{(v)} \in \mathbb{R}^{c\times d_1}$ is the transformation matrix of the 2nd layer of the v-th view-specific network. Besides, the output of the 2nd layer is associated with the instance-wise semantic label $\mathbf{Y} \in \mathbb{R}^{c\times n}$ to enhance its discriminative capability.
- The 3rd layers seek a common latent subspace from multiple view-specific network and learn the consensus multi-view factor $\mathbf{H} \in \mathbb{R}^{r\times n}$. It can be represented as $\mathbf{H} = \sum_{v=1}^{V} \mu_v \mathbf{W}_3^{(v)}\mathbf{U}_2^{(v)}$, where the consensus multi-view factor $\mathbf{H}$ is the output of the 3rd layer, μ_v is the explicit weight and it measures the importance of the v-th view features, $\mathbf{W}_3^{(v)} \in \mathbb{R}^{r\times c}$ is the transformation matrix of the 3rd layer in the v-th view-specific network. Note that, different from unsupervised methods, these weights can be effectively learned with the semantic label supervision.
- The 4th layer transforms the consensus multi-view factor $\mathbf{H}$ to multi-view hash code $\mathbf{B} \in \{-1, 1\}^{r\times n}$, and correlates it with the pair-wise semantic matrix $\mathbf{S} \in \{-1, 1\}^{n\times n}$ to further enhance its semantic representation capability. It can be represented as $\mathbf{B} = (\mathtt{sgn}(\mathbf{W}_4\mathbf{H}) + 1)/2$, where $\mathbf{W}_4 \in \mathbb{R}^{r\times r}$ is orthogonal transformation matrix of the 4th layer of the fusion network.
- The 5th layer outputs the binary hash codes $\mathbf{B}$.

As a consequence, the hash codes can be generated through the following subsequent steps

$$\begin{aligned} \mathbf{B} &= (\mathtt{sgn}(\mathbf{W}_4\mathbf{H}) + 1)/2, \\ \mathbf{H} &= \sum_{v=1}^{V} \mu_v \mathbf{W}_3^{(v)}\mathbf{U}_2^{(v)}, \\ \mathbf{U}_2^{(v)} &= \mathbf{W}_2^{(v)}\mathbf{U}_1^{(v)}, \quad v = 1, \ldots, V, \\ \mathbf{U}_1^{(v)} &= \mathbf{W}_1^{(v)}\mathbf{X}^{(v)}, \quad v = 1, \ldots, V. \end{aligned} \tag{1}$$

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

TABLE II
IMPORTANT NOTATIONS USED IN THIS PAPER

Notation	Dimensionality	Description
$\mathbf{X}^{(v)}$	$p_v \times n$	the feature matrix of the v-th view data
$\mathbf{Y}$	$c \times n$	the instance-wise semantic label matrix
$\mathbf{S}$	$n \times n$	the pair-wise semantic matrix
$\mathbf{W}_1^{(v)}$	$d_1 \times p_v$	the feature selection matrix of the 1st layer in the v-th view-specific network
$\mathbf{U}_1^{(v)}$	$d_1 \times n$	the output of the 1st layer
$\mathbf{W}_2^{(v)}$	$c \times d_1$	the transformation matrix of the 2nd layer in the v-th view-specific network
$\mathbf{U}_2^{(v)}$	$c \times n$	the output of the 2nd layer
$\mathbf{W}_3^{(v)}$	$r \times c$	the transformation matrix of the 3rd layer in the v-th view-specific network
$\mathbf{H}$	$r \times n$	the consensus multi-view factor
$\mathbf{W}_4$	$r \times r$	the orthogonal transformation matrix of the 4th layer of the fusion network
$\mathbf{B}$	$\{-1,1\}^{r\times n}$	the hash code matrix
μ_v		the weight of the v-th view
p_v		the dimensionality of the v-th view data
n		the number of training samples
V		the number of views
c		the number of semantic categories
r		the length of hash code
d_1		the dimensionality of the 1st layer in the view-specific network
d_2		the dimensionality of the 2nd layer in the view-specific network

The overall objective function is derived as

$$\begin{aligned}\min_{\mathbf{W}_k^{(v)}|_{k=1,\ldots,3,v=1,\ldots,V},\mathbf{W}_4,\mathbf{H},\mathbf{B}} & \beta\|\mathbf{B}-\mathbf{W}_4\mathbf{H}\|_F^2+\alpha\|r\mathbf{S}-\mathbf{B}^T\mathbf{W}_4\mathbf{H}\|_F^2\\ &+\sum_{v=1}^{V}(\mu_v\|\mathbf{H}-\mathbf{W}_3^{(v)}\mathbf{W}_2^{(v)}\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}\|_F^2\\ &+\theta\|\mathbf{W}_2^{(v)}\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}-\mathbf{Y}\|_F^2\\ &+\gamma\|\mathbf{W}_1^{(v)}\|_{2,1})+\delta\sum_{v=1}^{V}\sum_{k=2}^{3}\|\mathbf{W}_k^{(v)}\|_F^2,\\ s.t.\ \mathbf{B}\in\{-1,1\}^{r\times n},&\quad \mathbf{W}_4^T\mathbf{W}_4=\mathbf{I}_r,\\ &\sum_{v=1}^{V}\mu_v=1,\quad \mu_v\geq 0. \end{aligned}\tag{2}$$

where β, α, θ, γ, and δ are balance parameters. The first term transforms the multi-view hash code $\mathbf{B}$ from the consensus multi-view factor $\mathbf{H}$ with orthogonal rotation matrix. The second term learns hash code with the supervision of pair-wise semantic matrix. Note that, the traditional supervised hashing learning strategy $\min_{\mathbf{B}\in\{-1,1\}^{r\times n}}\|r\mathbf{S}-\mathbf{B}^T\mathbf{B}\|_F^2$ is very challenging due to the symmetric matrix factorization and discrete constraint. To solve this problem, we substitute one of $\mathbf{B}$ with the output of the previous layer $\mathbf{W}_4\mathbf{H}$. The third term models the transformation from heterogeneous view features into the consensus multi-view factor $\mathbf{H}$ with projection consistence. The fourth term associates the output of the 2nd layers with the instance-wise semantic label $\mathbf{Y}\in\mathbb{R}^{c\times n}$. The fifth term indicates that the 1st layers are feature selection layers, which can remove the redundant or noisy features [51]. The last term is a regularization term to avoid over-fitting.

C. Discrete Hash Code Optimization

It is nontrivial to solve binary hash codes due to the discrete constraint. In this paper, with the support of objective formulation, we propose to directly learn the discrete multi-view hash code based on augmented Lagrangian multiplier (ALM) [52], [53]. Our idea is to introduce auxiliary variables to separate constraints, and transform the objective function to an equivalent one that can be tackled more easily. Specifically, we add auxiliary variables $\mathbf{Z}_w\in\mathbb{R}^{r\times r}$ and $\mathbf{Z}_b\in\{-1,1\}^{r\times n}$, and set $\mathbf{Z}_w=\mathbf{W}_4$ and $\mathbf{Z}_b=\mathbf{B}$. With the substitution of variables, we derive the following iterative optimization steps to solve Eq.(2).

Step 1: update $\mu_v|_{v=1}^V$. We denote $h_v=\|\mathbf{H}-\mathbf{W}_3^{(v)}\mathbf{W}_2^{(v)}\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}\|_F^2+\theta\|\mathbf{W}_2^{(v)}\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}-\mathbf{Y}\|_F^2+\gamma\|\mathbf{W}_1^{(v)}\|_{2,1}+\delta\sum_{k=2}^{3}\|\mathbf{W}_k^{(v)}\|_F^2$. In some cases, $\mu_v=1$ corresponding to the minimum h_v over all views, and $\mu_v=0$ otherwise, leading to that only one view is selected but other views are ignored. Thus we have $\mu_v\leftarrow\mu_v^t$, $t>1$. The reformulated objective function is $\min_{\mu_v\geq 0,\sum_{v=1}^V\mu_v=1}\sum_{v=1}^V\mu_v^t h_v$. μ_v is calculated as

$$\mu_v=\frac{(1/h_v)^{1/(t-1)}}{\sum_{v=1}^{V}(1/h_v)^{1/(t-1)}}.\tag{3}$$

Step 2: update $\mathbf{W}_1^{(v)}|_{v=1}^V$. The objective function with respect to $\mathbf{W}_1^{(v)}$ can be represented as

$$\begin{aligned}\min_{\mathbf{W}_1^{(v)}}&\sum_{v=1}^{V}\mu_v\|\mathbf{H}-\mathbf{W}_3^{(v)}\mathbf{W}_2^{(v)}\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}\|_F^2\\ &+\gamma\|\mathbf{W}_1^{(v)}\|_{2,1}+\delta\|\mathbf{W}_1^{(v)}\|_F^2,\\ s.t.\ \mathbf{B}&\in\{-1,1\}^{r\times n}.\end{aligned}\tag{4}$$

By calculating the derivative of Eq.(2) with respect to $\mathbf{W}_1^{(v)}$ and setting it to zero, the updating formula for $\mathbf{W}_1^{(v)}$ can be derived as

$$\begin{aligned}\mathbf{W}_1^{(v)}=&(\mu_v\mathbf{W}_2^{(v)T}\mathbf{W}_3^{(v)T}\mathbf{W}_3^{(v)}\mathbf{W}_2^{(v)}+\theta\mathbf{W}_2^{(v)T}\mathbf{W}_2^{(v)}\\ &+\gamma\,\mathbf{D}^{(v)})^{-1}(\mu_v\mathbf{W}_2^{(v)T}\mathbf{W}_3^{(v)T}\mathbf{H}\mathbf{X}^{(v)T}\end{aligned}$$

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

$$+\theta\mathbf{W}_2^{(v)T}\mathbf{Y}\mathbf{X}^{(v)T})((\mu_v+\theta)\mathbf{X}^{(v)}\mathbf{X}^{(v)T}+\gamma\,\mathbf{I}_{p_v})^{-1}, \tag{5}$$

where $\mathbf{D}^{(v)} \in \mathbb{R}^{d_1\times d_1}$ is a diagonal matrix with $\mathbf{D}_{ii}^{(v)} = \frac{1}{2\|(\mathbf{W}_1^{(v)})_{i\cdot}\|_2}$. It is worth noting that in practice, $\|(\mathbf{W}_1^{(v)})_{i\cdot}\|_2$ could be close to zero. Theoretically, it could be zeros. For this case, we can regularize $\mathbf{D}_{ii}^{(v)} = \frac{1}{2(\|(\mathbf{W}_1^{(v)})_{i\cdot}\|_2+\epsilon)}$, where ϵ is very small constant.

Step 3: update $\mathbf{W}_2^{(v)}|_{v=1}^V$. By calculating the derivative of Eq.(2) with respect to $\mathbf{W}_2^{(v)}$ and setting it to zero, the updating formula for $\mathbf{W}_2^{(v)}$ can be derived as

$$\begin{aligned}\mathbf{W}_2^{(v)} = &(\mu_v\mathbf{W}_3^{(v)T}\mathbf{W}_3^{(v)}+(\theta+\delta)\mathbf{I}_c)^{-1}\\ &\times(\mu_v\mathbf{W}_3^{(v)T}\mathbf{H}\mathbf{X}^{(v)T}\mathbf{W}_1^{(v)T}+\theta\mathbf{Y}\mathbf{X}^{(v)T}\mathbf{W}_1^{(v)T})\\ &\times((\mu_v+\theta)\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}\mathbf{X}^{(v)T}\mathbf{W}_1^{(v)T}+\delta\mathbf{I}_{d_1})^{-1}.\end{aligned} \tag{6}$$

Step 4: update $\mathbf{W}_3^{(v)}|_{v=1}^V$. By calculating the derivative of Eq.(2) with respect to $\mathbf{W}_3^{(v)}$ and setting it to zero, the updating formula for $\mathbf{W}_3^{(v)}$ can be derived as

$$\begin{aligned}\mathbf{W}_3^{(v)} = &(\mu_v\mathbf{H}\mathbf{X}^{(v)T}\mathbf{W}_1^{(v)T}\mathbf{W}_2^{(v)T})\\ &\times(\mu_v\mathbf{W}_2^{(v)}\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}\mathbf{X}^{(v)T}\mathbf{W}_1^{(v)T}\mathbf{W}_2^{(v)T}+\delta\mathbf{I}_c)^{-1}.\end{aligned} \tag{7}$$

Step 5: update $\mathbf{W}_4$. By fixing the other variables, the optimization formula for $\mathbf{W}_4$ can be derived as

$$\begin{aligned}\min_{\mathbf{W}_4^T\mathbf{W}_4=\mathbf{I}_r}\ &\beta\,\mathtt{tr}(-2\mathbf{W}_4^T\mathbf{B}\mathbf{H}^T+\mathbf{W}_4^T\mathbf{W}_4\mathbf{H}\mathbf{H}^T)\\ &+\alpha\,\mathtt{tr}(-2r\mathbf{W}_4^T\mathbf{B}\mathbf{S}\mathbf{H}^T+\mathbf{W}_4^T\mathbf{B}\mathbf{B}^T\mathbf{W}_4\mathbf{H}\mathbf{H}^T).\end{aligned} \tag{8}$$

With the auxiliary variable $\mathbf{Z}_w \in \mathbb{R}^{r\times r}$ and $\mathbf{Z}_w = \mathbf{W}_4$, $\mathbf{Z}_w^T\mathbf{Z}_w = \mathbf{I}_r$, Eq.(8) could be transformed as

$$\begin{aligned}\min_{\mathbf{W}_4^T\mathbf{W}_4=\mathbf{I}_r}\ &\beta\,\mathtt{tr}(-2\mathbf{W}_4^T\mathbf{B}\mathbf{H}^T+\mathbf{W}_4^T\mathbf{Z}_w\mathbf{H}\mathbf{H}^T)\\ &+\alpha\,\mathtt{tr}(-2r\mathbf{W}_4^T\mathbf{B}\mathbf{S}\mathbf{H}^T+\mathbf{W}_4^T\mathbf{B}\mathbf{B}^T\mathbf{Z}_w\mathbf{H}\mathbf{H}^T)\\ &+\frac{\rho}{2}\|\mathbf{W}_4-\mathbf{Z}_w+\frac{\mathbf{G}_w}{\rho}\|_F^2,\end{aligned}$$

where $\mathbf{G}_w \in \mathbb{R}^{r\times r}$ measures the difference between the target and auxiliary variable, $\rho > 0$ adjusts the balance between terms. The above equation can be represented as

$$\max_{\mathbf{W}_4^T\mathbf{W}_4=\mathbf{I}_r}\ \mathtt{tr}(\mathbf{W}_4^T\mathbf{C}_w), \tag{9}$$

where $\mathbf{C}_w = 2\beta\mathbf{B}\mathbf{H}^T-\beta\mathbf{Z}_w\mathbf{H}\mathbf{H}^T+2\alpha r\mathbf{B}\mathbf{S}\mathbf{H}^T-\alpha\mathbf{B}\mathbf{B}^T\mathbf{Z}_w\mathbf{H}\mathbf{H}^T+\rho\mathbf{Z}_w-\mathbf{G}_w$. The optimal $\mathbf{W}_4$ is defined as $\mathbf{W}_4 = \mathbf{P}_w\mathbf{Q}_w^T$, where $\mathbf{P}_w$ and $\mathbf{Q}_w$ are comprised of left-singular and right-singular vectors of $\mathbf{C}_w$ respectively [54].

Note that, the $\mathbf{S} \in \mathbb{R}^{n\times n}$ is included in the term $\mathbf{B}\mathbf{S}\mathbf{H}^T$ when updating $\mathbf{W}_4$. If we compute $\mathbf{S}$ explicitly, the computational complexity is $\mathcal{O}(n^2)$. In this paper, we utilize $c\times n$ matrix $\tilde{\mathbf{Y}}$ (c is the number of semantic categories) to store the label information instead of directly calculating $\mathbf{S}$. This can reduce the computational complexity to $\mathcal{O}(n)$. Let $\tilde{\mathbf{Y}}_{ki} = \frac{l_{ki}}{\|l_i\|_2}$ be the element at the k-th row and the i-th column in the matrix $\tilde{\mathbf{Y}}$. Then we can get the similarity matrix $\tilde{\mathbf{S}} = \tilde{\mathbf{Y}}^T\tilde{\mathbf{Y}}$. The semantic similarity matrix $\mathbf{S}$ can be calculated as

$$\mathbf{S} = 2\tilde{\mathbf{S}}-\mathbf{E} = 2\tilde{\mathbf{Y}}^T\tilde{\mathbf{Y}}-\mathbf{1}_n\mathbf{1}_n^T, \tag{10}$$

where $\mathbf{1}_n$ is an all-one column vector with length n, and $\mathbf{E}$ is a matrix with all elements as 1. Then we can get

$$\mathbf{B}\mathbf{S}\mathbf{H}^T = 2(\mathbf{B}\mathbf{Y}^T)(\mathbf{H}\mathbf{Y}^T)^T-(\mathbf{B}\mathbf{1}_n)(\mathbf{H}\mathbf{1}_n)^T,$$

which consumes $\mathcal{O}(2crn+r^2c)$.

Step 6: update H. By calculating the derivative of Eq.(2) with respect to $\mathbf{H}$ and setting it to zero, the updating formula for $\mathbf{H}$ can be derived as

$$\begin{aligned}\mathbf{H} = &(\alpha\mathbf{W}_4^T\mathbf{B}\mathbf{B}^T\mathbf{W}_4+\beta\mathbf{W}_4^T\mathbf{W}_4+\mathbf{I}_r)^{-1}(\sum_{v=1}^V\mu_v\mathbf{W}_3^{(v)}\mathbf{W}_2^{(v)}\\ &\times\mathbf{W}_1^{(v)}\mathbf{X}^{(v)}+\beta\mathbf{W}_4^T\mathbf{B}+\alpha r\mathbf{W}_4^T\mathbf{B}\mathbf{S}),\end{aligned} \tag{11}$$

where $\mathbf{S}$ is also transformed using Eq.(10), then we have

$$\mathbf{W}_4^T\mathbf{B}\mathbf{S} = 2\mathbf{W}_4^T\mathbf{B}\tilde{\mathbf{Y}}^T\tilde{\mathbf{Y}}-\mathbf{W}_4^T\mathbf{B}\mathbf{1}_n\mathbf{1}_n^T.$$

The time complexity of computing $\mathbf{W}_4^T\mathbf{B}\mathbf{S}$ is reduced to $\mathcal{O}(crn+r^2n)$.

Step 7: update B. By fixing the other variables, the optimization formula with respect to $\mathbf{B}$ can be represented as

$$\begin{aligned}\min_{\mathbf{B}\in\{-1,1\}^{r\times n}}\ &-2\beta\,\mathtt{tr}(\mathbf{B}^T\mathbf{W}_4\mathbf{H})+\alpha\,\mathtt{tr}(-2r\mathbf{B}^T\mathbf{W}_4\mathbf{H}\mathbf{S}^T\\ &+\mathbf{B}^T\mathbf{W}_4\mathbf{H}\mathbf{H}^T\mathbf{W}_4^T\mathbf{B}).\end{aligned} \tag{12}$$

With the auxiliary variable $\mathbf{Z}_b \in \{-1,1\}^{r\times n}$ and $\mathbf{Z}_b = \mathbf{B}$, Eq.(12) could be transformed as

$$\begin{aligned}\min_{\mathbf{B}\in\{-1,1\}^{r\times n}}\ &-2\beta\,\mathtt{tr}(\mathbf{B}^T\mathbf{W}_4\mathbf{H})+\alpha\,\mathtt{tr}(-2r\mathbf{B}^T\mathbf{W}_4\mathbf{H}\mathbf{S}^T\\ &+\mathbf{B}^T\mathbf{W}_4\mathbf{H}\mathbf{H}^T\mathbf{W}_4^T\mathbf{Z}_b)+\frac{\rho}{2}\|\mathbf{B}-\mathbf{Z}_b+\frac{\mathbf{G}_b}{\rho}\|_F^2,\end{aligned} \tag{13}$$

where $\mathbf{G}_b \in \mathbb{R}^{r\times n}$ measures the difference between the target and auxiliary variable. Eq.(13) is equivalent to

$$\begin{aligned}\min_{\mathbf{B}\in\{-1,1\}^{r\times n}}\ &\mathtt{tr}(-\mathbf{B}^T(2\beta\mathbf{W}_4\mathbf{H}+2\alpha r\mathbf{W}_4\mathbf{H}\mathbf{S}^T\\ &-\alpha\mathbf{W}_4\mathbf{H}\mathbf{H}^T\mathbf{W}_4^T\mathbf{Z}_b+\rho\mathbf{Z}_b-\mathbf{G}_b)).\end{aligned} \tag{14}$$

The solution of $\mathbf{B}$ can be obtained as the following closed form

$$\begin{aligned}\mathbf{B} = \mathtt{sgn}(&2\beta\mathbf{W}_4\mathbf{H}+2\alpha r\mathbf{W}_4\mathbf{H}\mathbf{S}^T\\ &-\alpha\mathbf{W}_4\mathbf{H}\mathbf{H}^T\mathbf{W}_4^T\mathbf{Z}_b+\rho\mathbf{Z}_b-\mathbf{G}_b),\end{aligned} \tag{15}$$

where $\mathbf{S}$ is also transformed using Eq.(10), then we have

$$\mathbf{W}_4\mathbf{H}\mathbf{S}^T = 2\mathbf{W}_4\mathbf{H}\tilde{\mathbf{Y}}^T\tilde{\mathbf{Y}}-\mathbf{W}_4\mathbf{H}\mathbf{1}_n\mathbf{1}_n^T,$$

and the time complexity is $\mathcal{O}(crn+r^2n)$.

The hash codes $\mathbf{B}$ are directly solved with a single hash code-solving operation, which is faster than iterative hash code-solving step in DCC.

Step 8: update $\mathbf{Z}_w$. By fixing the other variables, the optimization formula for $\mathbf{Z}_w$ is

$$\max_{\mathbf{Z}_w^T\mathbf{Z}_w=\mathbf{I}_r}\ \mathtt{tr}(\mathbf{Z}_w^T\mathbf{C}_z), \tag{16}$$

where $\mathbf{C}_z = -\beta\mathbf{W}_4\mathbf{H}\mathbf{H}^T-\alpha\mathbf{B}\mathbf{B}^T\mathbf{W}_4\mathbf{H}\mathbf{H}^T+\rho\mathbf{W}_4+\mathbf{G}_w$. Similar to the updating of $\mathbf{W}_4$, the optimal $\mathbf{Z}_w$ is solved as $\mathbf{Z}_w = \mathbf{P}_{zw}\mathbf{Q}_{zw}^T$, where $\mathbf{P}_{zw}$ and $\mathbf{Q}_{zw}$ are comprised of left-singular and right-singular vectors of $\mathbf{C}_z$ respectively.

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Key Steps of Discrete Hash Code Optimization

Input: Training set $\{\mathbf{X}^{(v)} \in \mathbb{R}^{p_v \times n}\}_{v=1}^{V}$, instance-wise semantic label $\mathbf{Y} \in \mathbb{R}^{c \times n}$, pair-wise semantic matrix $\mathbf{S} \in \mathbb{R}^{n \times n}$, hash code length r, parameters β, α, θ, γ, δ and d_1.

Output: Transformation matrices $\mathbf{W}_k^{(v)}|_{k=1,...,3}$, $\mathbf{W}_4$, weight μ_v, $v = 1, ..., V$.

Initialize $\mu_v = \frac{1}{V}$, initialize $\mathbf{B}$ and $\mathbf{Z}_b$ as $\{-1, 1\}^{r \times n}$ matrix.
Initialize $\mathbf{W}_1^{(v)}|_{v=1,...,V}$ as $d_1 \times p_v$ random variable.
Initialize $\mathbf{W}_2^{(v)}|_{v=1,...,V}$ as $c \times d_1$ random variable.
Initialize $\mathbf{W}_3^{(v)}|_{v=1,...,V}$ as $r \times c$ random variable.
Initialize $\mathbf{W}_4$ and $\mathbf{Z}_w$ as $r \times r$ random variables.
Initialize $\mathbf{G}_b = \mathbf{B} - \mathbf{Z}_b$ and $\mathbf{G}_w = \mathbf{W}_4 - \mathbf{Z}_w$.
Pre-training
$\mathbf{U}_1^{(v)} = \mathbf{W}_1^{(v)}\mathbf{X}^{(v)}$, $v = 1, ..., V$.
$\mathbf{U}_2^{(v)} = \mathbf{W}_2^{(v)}\mathbf{U}_1^{(v)}$, $v = 1, ..., V$.
$\mathbf{H} = \sum_{v=1}^{V} \mu_v \mathbf{W}_3^{(v)}\mathbf{U}_2^{(v)}$, $v = 1, ..., V$.
$\mathbf{B} = \mathtt{sgn}(\mathbf{W}_4\mathbf{H})$.
repeat
  for $v = 1, ..., V$ **do**
    Update μ_v by Eq.(3).
    for $k = 1, ..., 3$ **do**
      Update $\mathbf{W}_k^{(v)}$ by Eq.(5), (6), (7).
    end for
  end for
  Update $\mathbf{W}_4$ by Eq.(9).
  Update $\mathbf{H}$ by Eq.(11).
  Update $\mathbf{B}$ by Eq.(15)
  Update $\mathbf{Z}_w$ and $\mathbf{Z}_b$ by Eq.(16) and (17), respectively.
  Update $\mathbf{G}_w$ and $\mathbf{G}_b$ by Eq.(18).
until convergence.

Step 9: update $\mathbf{Z}_b$. $\mathbf{Z}_b$ can be solved with the following closed form

$$\mathbf{Z}_b = \mathtt{sgn}(-\alpha\mathbf{W}_4\mathbf{H}\mathbf{H}^T\mathbf{W}_4^T\mathbf{B} + \rho\mathbf{B} + \mathbf{G}_b). \quad (17)$$

Step 10: update $\mathbf{G}_w$ and $\mathbf{G}_b$. The update rules are

$$\mathbf{G}_w = \mathbf{G}_w - \rho(\mathbf{W}_4 - \mathbf{Z}_w),$$
$$\mathbf{G}_b = \mathbf{G}_b - \rho(\mathbf{B} - \mathbf{Z}_b). \quad (18)$$

Algorithm 1 summarizes the main procedures of the discrete hash code optimization. The convergence criterion is that $(O_{iter} - O_{iter-1}) < \xi$, where O_{iter} is the value of the objective function in the $iter$-th iteration, and convergence error ξ is usually set to 10^{-4}.

D. Online Hashing for New Queries

In the online hashing process, we aim to map the new coming query instances into binary hash codes for fast image search. With the learned transformation matrices and the weight of each view $\mu_v|_{v=1}^{V}$ from Algorithm 1, the new query represented with multi-view features $\mathbf{X}_q^{(v)}|_{v=1}^{V}$ can be projected into hash codes $\mathbf{B}_q$ as

$$\mathbf{B}_q = (\mathtt{sgn}(\mathbf{W}_4 \sum_{v=1}^{V} \mu_v \mathbf{W}_3^{(v)}\mathbf{W}_2^{(v)}\mathbf{W}_1^{(v)}\mathbf{X}_q^{(v)}) + 1)/2. \quad (19)$$

E. Discussions

1) Complexity Analysis: This section provides the computational complexity analysis of the proposed DCMVH. We assume that $n \gg d_1$, $n \gg p_v$, $p_v \gg r$, $p_v \gg c$. It takes $\mathcal{O}(p_v r n)$ for updating $\mu^{(v)}$. Updating $\mathbf{W}_1^{(v)}|_{v=1}^{V}$ costs $\mathcal{O}((d_1 + p_v)p_v n)$. Updating $\mathbf{W}_2^{(v)}|_{v=1}^{V}$ costs $\mathcal{O}((d_1 + c)p_v n)$. Updating $\mathbf{W}_3^{(v)}|_{v=1}^{V}$ costs $\mathcal{O}((c + r)p_v n)$. Updating $\mathbf{W}_4$ costs $\mathcal{O}((c + r)rn)$. Updating $\mathbf{H}$ costs $\mathcal{O}(rp_v n)$. Updating $\mathbf{B}$ requires $\mathcal{O}((c+r)rn)$. Updating $\mathbf{Z}_w$ costs $O(r^2 n)$. Updating $\mathbf{Z}_b$ costs $O(r^2 n)$. Updating $\mathbf{G}_w$ costs $O(r^2)$, Updating $\mathbf{G}_b$ costs $O(rn)$. Generally, the computational complexity of the iterative discrete optimization process in Algorithm 1 is $\mathcal{O}(iter \times n)$, where $iter$ is the number of iteration. This process scales linearly with n. At online search process, it takes $\mathcal{O}(iter \times p_v r n)$ to generate binary hash code for a newly coming query instance.

Please note that, in the discrete optimization process, we avoid explicitly computing the pair-wise similarity matrix $\mathbf{S}$. $\mathbf{S}$ is calculated with the formula based on $\tilde{\mathbf{Y}}$, which is a $c \times n$ matrix to store label information. This strategy successfully reduces the computational complexity and space cost to $\mathcal{O}(n)$.

In a sum, the computational and space complexity of the proposed DCMVH is linearly related to the size of the dataset. Our method has well scalability.

2) Convergence Analysis: Algorithm 1 iteratively updates variables. Specifically, we update each variable by fixing the others. The optimization of a variable at each step will lead to a lower or equal value of the objective function.[1] Therefore, each iteration of the optimization method in Algorithm 1 monotonously reduces the objective function value. After several iterations, the objective function will reach a local minimum. In addition, we empirically verify the convergence of the proposed DCMVH on three benchmark datasets in experiments.

IV. Experiments

A. Evaluation Datasets

In experiments, we evaluate the proposed DCMVH for large-scale multi-view image search tasks. We adopt three publicly available datasets: MIR Flickr [55], MS COCO [56] and NUS-WIDE [57]. These datasets have been widely used for evaluating the image search performance [20], [22]. Without loss of generality, in each dataset, we extract heterogeneous visual features and the corresponding tag features from each image instance as existing works [20], [22]. The statistics of three datasets used in experiments are summarized in Table III, and their detailed descriptions are as follows:

- **MIR Flickr**[2] is comprised of 25,000 images collected from Flickr.[3] Each image is annotated with several user assigned tags, which are taken from 24 labels. We randomly select 100 images from each category to comprise the query set. After removing the repeated images without textual tags or manually annotated labels, we have 2,243 images as the query set and the remaining images are treated as the database set. We randomly select 5,000 images from the database for training. We extract

[1] In Step 7 and 9, we can directly obtain closed form solution.
[2] http://lear.inrialpes.fr/people/guillaumin/data.php
[3] https://www.flickr.com/

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

TABLE III
GENERAL STATISTICS OF THREE DATASETS. CNN FEATURE IS OBTAINED BY CAFFE IMPLEMENTATION OF VGG NET. BOW DENOTES THE BAG-OF-WORDS FEATURE

Datasets	Training Size	Retrieval Size	Query Size	Categories	Visual Feature	Tag Feature
MIR Flickr [55]	5,000	17,772	2,243	24	CNN (4,096-D)	BoW (1,386-D)
MS COCO [56]	18,000	82,783	5,981	80	CNN (4,096-D)	BoW (2,000-D)
NUS-WIDE [57]	21,000	193,749	2,085	21	CNN (4,096-D)	BoW (1,000-D)

4096-dimensional CNN feature from VGG-Net [33] to represent image contents, and 1,386 dimensional bag-of-words tag feature to represent the texts.

- **MS COCO**[4] contains more than 300,000 images for object detection, segmentation, and captioning tasks. In experiments, we adopt MS COCO 2014 dataset, which contains 82,783 training images and 40,504 validation images. Each image is associated with at least one of 80 object categories. In our experiments, we randomly select 80 images for each category from the validation images to form the query set. Besides, from the training images, we select the 82,783 images as the database and randomly select 18,000 images within them for training. After pruning the images without any label or tag information, we obtain the query set with 5,981 images. We extract 4096-dimensional CNN feature from VGG-Net to represent image contents, and 2,000 dimensional bag-of-words tag feature to represent the texts.
- **NUS-WIDE**[5] consists of 269,648 Flickr images with 81 ground-truth semantic concepts. In experiments, we select 21 most common concepts and the corresponding 195,834 images. We randomly select 100 images for each category, thus the corresponding 2,085 images are selected to comprise the query set. The remaining 193,749 images are treated as the database. 21,000 images from the database are randomly selected for training. We extract 4096-dimensional CNN feature from VGG-Net to represent image contents, and 1,000 dimensional bag-of-words tag feature to represent the texts.

B. Evaluation Metrics

In experiments, similar to existing hashing methods [14], [20], [21], [39], we evaluate the image search performance by two standard evaluation metrics, Mean Average Precision (MAP) [39] and topK-precision [21]. We measure the similarity between the binary codes of the query set and that of retrieval set with Hamming distance. Two images are considered to be semantically similar when they share at least one semantic label. Given a query sample q, average precision (AP) is defined as

$$AP(q) = \frac{1}{L_q} \sum_{r=1}^{R} P_q(r)\delta_q(r), \tag{20}$$

where L_q is the number of the true neighbors in the retrieved list, $P_q(r)$ denotes the precision of the top r retrieved results, and $\delta_q(r) = 1$ if the r-th instance is the true neighbor and 0 otherwise. For all the queries, we first calculate their APs and then use the average value as mAP. For both evaluation protocols, larger value indicates better performance.

[4]http://cocodataset.org/

[5]http://lms.comp.nus.edu.sg/research/NUS-WIDE.htm

C. Evaluation Baselines

We compare the proposed method with several state-of-the-art multi-view hashing methods, including Multiple Feature Hashing (MFH) [19], Multiple Feature Kernel Hashing (MFKH) [15], Multi-view Alignment Hashing (MAH) [20], Multi-view Latent Hashing (MVLH) [14], Multi-view Discrete Hashing (MVDH) [21], Discrete Multi-view Hashing (DMVH) [22]. These methods have been discussed in the related works in Section II. Source codes of MFH and MFKH are publicly available, and the other methods are implemented by ourselves.

D. Implementation Details

The proposed DCMVH has several parameters: β, α, θ, γ and δ in Eq.(2), and ρ in the discrete hash code optimization, and the dimension of the 1st layer d_1. In the experiments, the best performance is achieved when $\{\beta = 10^{-1}, \alpha = 10^{-5}, \theta = 10^{-5}, \gamma = 10^{3}, \delta = 10^{-1}, \rho = 10^{5}, d_1 = 2^{11}\}$, $\{\beta = 10^{-3}, \alpha = 10^{-5}, \theta = 10^{-5}, \gamma = 10^{1}, \delta = 10^{1}, \rho = 10^{5}, d_1 = 2^{11}\}$, and $\{\beta = 10^{-5}, \alpha = 10^{-5}, \theta = 10^{-5}, \gamma = 10^{3}, \delta = 10^{-1}, \rho = 10^{5}, d_1 = 2^{11}\}$ on MIR Flickr, NUS-WIDE and MS COCO, respectively.

We carefully tune the parameters of all the baselines and finally report their best results for performance comparison. On three datasets, we conduct five successive experiments with randomly partitioned datasets and report the average results. All our experiments are conducted on a workstation with a 3.40 GHz Intel(R) Core(TM) i7-6700 CPU and 64 GB RAM.

E. Comparison Results

The MAP values of all compared methods varying with different hash code lengths (16 bits, 32 bits, 64 bits, and 128 bits) on three datasets (MIR Flickr, NUS-WIDE and MS COCO) are presented in Table IV. From the results, we can find that the proposed DCMVH outperforms all the compared multi-view hashing methods. For example, compared with the second best multi-view hashing method DMVH, the average mAP score of our method has been increased by 7.49%, 3.34%, and 10.98% on MIR Flickr, NUS-WIDE, and MS COCO respectively. Especially, on the MIR Flickr and MS COCO, the mAP of DCMVH on 16 bits is higher than that of DMVH on 128 bits. On NUS-WIDE, the mAP of DCMVH on 16 bits is higher than that of DMVH on 64 bits. The main reasons for these superior results come from three aspects: 1) The proposed deep multi-view hash learning architecture could deeply fuse the

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

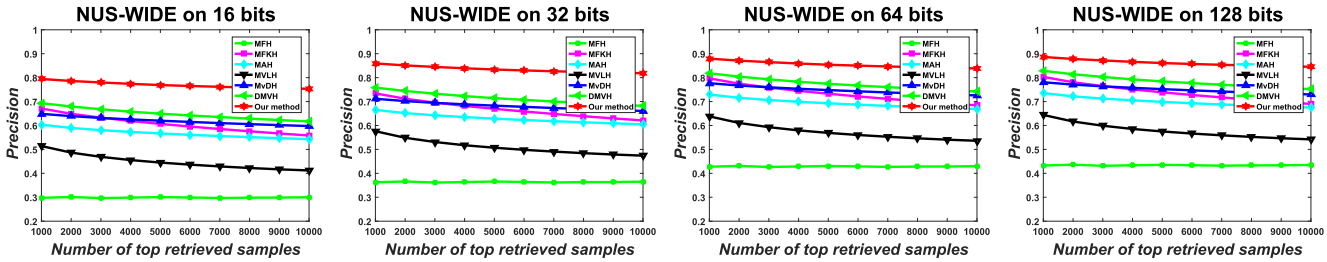

Fig. 2. topK-precision curves on NUS-WIDE.

TABLE IV
MAP COMPARISON RESULTS ON MIR FLICKR, MS COCO AND NUS-WIDE

Methods	MIR Flickr				MS COCO				NUS-WIDE			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
MFH	0.5795	0.5824	0.5831	0.5836	0.3948	0.3966	0.3960	0.3980	0.3603	0.3611	0.3625	0.3629
MFKH	0.6369	0.6128	0.5985	0.5807	0.4216	0.4211	0.4230	0.4229	0.4768	0.4359	0.4342	0.3956
MAH	0.6488	0.6649	0.6990	0.7114	0.3967	0.3943	0.3966	0.3988	0.4633	0.4945	0.5381	0.5476
MVLH	0.6541	0.6421	0.6044	0.5982	0.3993	0.4012	0.4065	0.4099	0.4182	0.4092	0.3789	0.3897
MVDH	0.6828	0.7210	0.7344	0.7527	0.3978	0.3966	0.3977	0.3998	0.4947	0.5661	0.5789	0.6122
DMVH	0.7231	0.7326	0.7495	0.7641	0.4123	0.4288	0.4355	0.4563	0.5676	0.5883	0.6092	0.6279
DCMVH	**0.8097**	**0.8279**	**0.8354**	**0.8467**	**0.5378**	**0.5427**	**0.5490**	**0.5576**	**0.6509**	**0.6625**	**0.6905**	**0.7023**

TABLE V
COMPARISON OF TRAINING AND QUERY TIME (SECONDS) WHEN THE HASH CODE LENGTH IS FIXED TO 128 BITS

Methods	Training Time (s)			Query Time (s)		
	MIR Flickr	MS COCO	NUS-WIDE	MIR Flickr	MS COCO	NUS-WIDE
MFH	240.51	7115.26	7524.72	18.59	175.51	236.68
MFKH	225.12	7013.72	7420.03	128.43	1248.06	1634.96
MAH	238.74	7099.29	7498.32	129.37	1250.07	1647.57
MVLH	88.25	304.00	363.16	23.29	226.18	295.44
MvDH	262.79	7523.91	7920.01	604.44	5782.36	7695.25
DMVH	201.43	7002.27	7407.08	126.68	1235.81	1613.24
Ours	31.97	137.50	154.22	10.77	101.46	137.08

multi-view features into the binary hash codes. 2) Dual-level semantic supervision enhance the discriminative capability of hash codes. 3) The proposed discrete optimization method avoids the binary quantization loss.

Moreover, Fig. 2 shows the corresponding topK-precision curves with the increasing number of the retrieved instances on NUS-WIDE. It can be observed that the precision of the proposed DCMVH consistently outperforms all the compared baselines. These results clearly validate the superiority of DCMVH over the baselines.

We further conduct comparison experiments to validate training and query efficiency of the proposed method. Specifically, we fix the hash code length to 128 bits, and record the training and query time (seconds) of all approaches in Table V. From this table, we can find that our method consumes the least time at both training and query stages. In particular, at the training stage, the training time of our method is 56.28s, 166.50s and 208.94s less than that of the second best method MVLH on all three datasets. This is because that MVLH learn the hash codes bit-by-bit, while our method directly obtains the hash codes with a simple and efficient one-step operation. On MIR Flickr, compared with the baselines MFH, MFKH, MAH, MvDH, and DMVH, the training efficiency of our method is improved by 7.52%, 7.04%, 7.47%, 8.22%, and 8.43%, respectively. The efficiency improvement is attributed to the reason that: the computation complexity of these compared approaches is $\mathcal{O}(n^2)$, while that of our method is $\mathcal{O}(n)$. At the query stage, the training time of our method is 7.82s, 61.17s and 99.6s less than that of the second best method MFH on three datasets. The reason is that: MFH first concatenates feature vectors into one vector and projects this vector into the Hamming space with a unified projection matrix. In contrast, our method generates query hash codes with the view-specific transformation matrices that reduce the computation complexity. In addition, we find that the query efficiency of MvDH is significantly higher than other baselines because it updates query hash codes bit-by-bit when new query samples comes.

F. Ablation Analysis

1) Effects of Deep Multi-View Hash Learning Architecture: In this paper, deep hash learning model is proposed to deeply fuse different view features into binary hash codes. To evaluate the effects of our designed deep architecture, we compare with three variants of the proposed DCMVH.

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

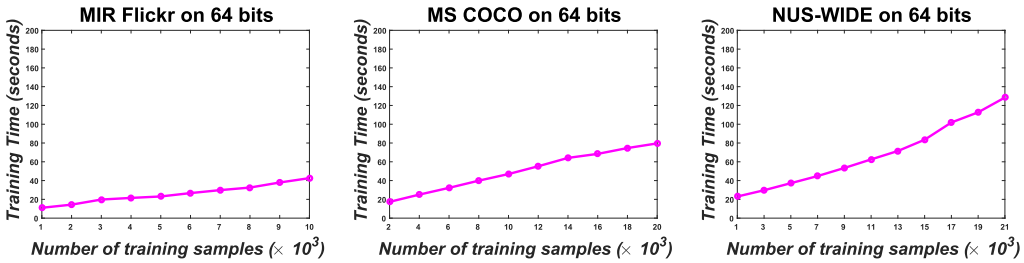

Fig. 3. Variations of training time with training data size on MIR FLickr, MS COCO and NUS-WIDE, when the code length is fixed to 64 bits.

TABLE VI
ABLATION EXPERIMENTS ON THREE DATASETS

Methods	MIR Flickr				MS COCO				NUS-WIDE			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
DCMVH-*selec*	0.6424	0.6655	0.6711	0.6768	0.4933	0.4989	0.5028	0.5082	0.4704	0.4846	0.4932	0.5047
DCMVH-*inst*	0.7418	0.7696	0.7820	0.7912	0.5192	0.5210	0.5222	0.5258	0.5817	0.6107	0.6194	0.6230
DCMVH-*pair*	0.7022	0.7306	0.7452	0.7505	0.4034	0.4041	0.4049	0.4092	0.4898	0.5110	0.5321	0.5597
DCMVH-*relax*	0.7075	0.7684	0.7802	0.7859	0.5040	0.5208	0.5249	0.5312	0.4814	0.5039	0.5329	0.6022
DCMVH-*img*	0.6037	0.6159	0.6224	0.6330	0.3911	0.3973	0.4025	0.4047	0.5722	0.5977	0.5987	0.6073
DCMVH-*txt*	0.5885	0.5988	0.5997	0.6025	0.4016	0.4020	0.4048	0.4094	0.4641	0.4723	0.4894	0.4952
DCMVH-*con*	0.6069	0.6185	0.6224	0.6363	0.3927	0.3955	0.3973	0.3995	0.6165	0.6048	0.5937	0.5904
DCMVH	**0.8097**	**0.8279**	**0.8354**	**0.8467**	**0.5378**	**0.5427**	**0.5490**	**0.5576**	**0.6509**	**0.6625**	**0.6905**	**0.7023**

- DCMVH-*selec*: it directly takes the original feature as the input of the 2nd layer. This variant is to evaluate the effects of the feature selection layer (the 1st layer) on filtering the feature noises from the original space.
- DCMVH-*inst*: it takes the output of the 1st layer as the input of the 3rd layer. This variant is to evaluate the effects of the instance-wise semantic label (in the 2nd layer) on enhancing the discriminative capability of hash codes.
- DCMVH-*pair*: it takes the output of the 5th layer as the final hash code without pair-wise semantic supervision. This variant is to evaluate the effects of the pair-wise semantic matrix (in the 4th layer) on enhancing the discriminative capability of hash codes.

Besides, we design three variants of the proposed DCMVH to evaluate the complementarity of multi-view features.

- DCMVH-*img*: it only inputs the visual feature into hash learning, hash codes, hash functions.
- DCMVH-*txt*: it only inputs the textual feature into hash learning, hash codes, hash functions.
- DCMVH-*con*: it inputs the concatenated feature into hash learning, hash codes, hash functions.

We conduct experiments on all three datasets with the code length varying from 16 bits, 32 bits, 64 bits and 128 bits. The comparison results on image search accuracy are presented in Table VI. From the results, we can observe that the proposed DCMVH can achieve consistent performance improvement. This is because: each layer in the proposed deep multi-view hash learning architecture is designed to enhance the image search performance. The 1st layer supports feature selection, which reduces the effects of adverse noises. The 2nd layer enhances the discriminative capability with instance-wise label. The 3rd layer exploits the complimentary of multi-view features. The 4th layer improves the semantic representation capability of hash codes further by correlating with pair-wise semantic matrix. This deep architecture can improve the accuracy of image search.

2) Effects of Discrete Hash Code Optimization: In this paper, to solve binary hash codes with the discrete constraint, we propose to directly learn them in a discrete optimization manner. To evaluate the proposed discrete hash code optimization, we design a variant of the proposed DCMVH, named DCMVH-*relax*. This variant first relaxes the discrete constraints to learn hash codes and then obtains binary codes by mean-thresholding. We conduct the experiments on all three datasets with the code length varying from 16 to 128 bits. The comparison results on retrieval accuracy are presented in Table IV. We can find that the proposed DCMVH achieves higher retrieval precision than the variant DCMVH-*relax*. These results validate that the proposed discrete hash optimization method can reduce the binary quantization loss and improve the retrieval performance.

In addition, we records the training time variations with training size on MIR Flickr, NUS-WIDE, and MS COCO, when the code length is fixed to 64 bits. The results are shown in Fig.3. The figure show that the training time increases almost linearly with training size. These results validate that the linear scalability of DCMVH and empirically confirm that DCMVH is suitable for large-scale datasets.

G. Parameter and Convergency Analysis

We conduct experiments to observe the performance variations with the involved parameters β, α, θ, γ, δ, and ρ. We report the results on MIR Flickr when the code length is fixed to 32 bits. The values of these parameters are selected from the range of $\{10^{-5}, 10^{-3}, 10^{-1}, 10, 10^{3}, 10^{5}\}$ while fixing the others. Besides, d_1 is the dimension of the 1st layer. the value of d_1 is taken in the range

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

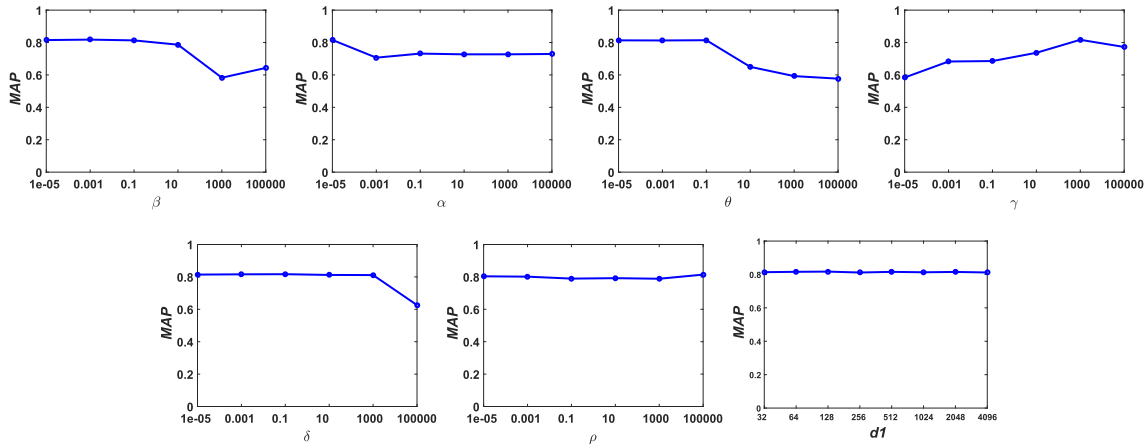

Fig. 4. Performance variations with the parameters on MIR Flickr when the code length is fixed to 32 bits.

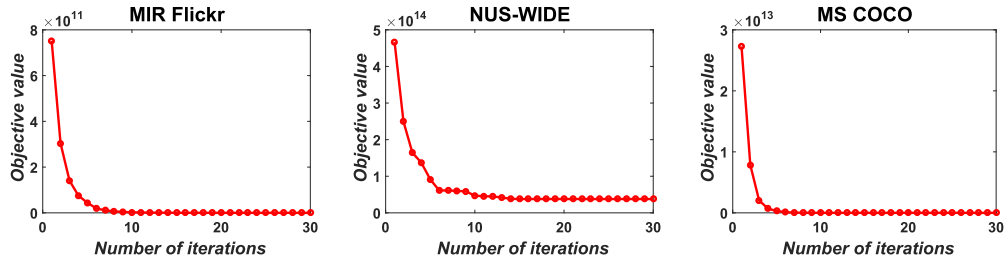

Fig. 5. Convergency analysis on MIR Flickr, NUS-WIDE, and MS COCO when the code length is fixed to 64 bits.

of [32, 64, 128, 256, 512, 1024, 2048, 4096]. Detailed experimental results are presented in Fig. 4. We can find that the performance is relatively stable when β is in a range of $\{10^{-5}, 10^{-3}, 10^{-1}, 10^{1}\}$, α is fixed to 10^{-5}, θ is in a range of $\{10^{-5}, 10^{-3}, 10^{-1}\}$, γ is fixed to $\{10^{3}\}$, δ is in a range of $\{10^{-5}, 10^{-3}, 10^{-1}, 10^{1}, 10^{3}\}$, ρ is fixed to 10^{5}, and d_1 is in a range of [32, 64, 128, 256, 512, 1024, 2048, 4096].

In addition, we conduct convergency analysis on MIR Flickr, NUS-WIDE and MS COCO with the code length is fixed to 64 bits. The convergency curves are shown in Fig. 5. As shown in the figures, on three datasets, the objective function value decreases gradually with the number of iterations, and finally reaches a local minimum after several iterations. We can find from the figures that, on three datasets, our method converges within ten iterations.

V. Conclusion and Future Work

In this paper, we propose a deep collaborative multi-view hashing method to fuse discriminative information in multi-view features into hash codes. We design a new deep multi-view hash learning model, where discriminative capability of representation layers are progressively enhanced. Besides, a fast discreet optimization method is proposed to efficiently and directly learn hash codes without relaxing binary quantization errors. Experiments on several public multi-view image search datasets demonstrate the state-of-the-art performance of the proposed method. In the future, we will further: 1) reduce the number of model parameters and make the proposed method more applicable in large-scale datasets. 2) evaluate the performance of the proposed method on testing datasets with more views.

Acknowledgment

The authors would like to thank the anonymous reviewers for their constructive and helpful suggestions.

References

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

[2] J. Lu, V. E. Liong, and J. Zhou, "Deep hashing for scalable image search," *IEEE Trans. Image Process.*, vol. 26, no. 5, pp. 2352–2367, May 2017.

[3] F. Shen *et al.*, "Classification by retrieval: Binarizing data and classifiers," in *Proc. ACM Int. Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2017, pp. 595–604.

[4] Z. Zhang, Q. Wang, L. Ruan, and L. Si, "Preference preserving hashing for efficient recommendation," in *Proc. ACM Int. Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2014, pp. 183–192.

[5] J. Qin *et al.*, "Binary coding for partial action analysis with limited observation ratios," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 6728–6737.

[6] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 35, no. 12, pp. 2916–2929, Dec. 2013.

[7] X. Zhou *et al.*, "Graph convolutional network hashing," *IEEE Trans. Cybern.*, to be published.

[8] F. Shen, X. Zhou, J. Yu, Y. Yang, L. Liu, and H. T. Shen, "Scalable zero-shot learning via binary visual-semantic embeddings," *IEEE Trans. Image Process.*, vol. 28, no. 7, pp. 3662–3674, Jul. 2019.

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

[9] X. Luo, P.-F. Zhang, Z. Huang, L. Nie, and X.-S. Xu, "Discrete hashing with multiple supervision," *IEEE Trans. Image Process.*, vol. 28, no. 6, pp. 2962–2975, Jun. 2019.

[10] D. Wang, X. Gao, X. Wang, and L. He, "Label consistent matrix factorization hashing for large-scale cross-modal similarity search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 10, pp. 2466–2479, Oct. 2019.

[11] M. Hu, Y. Yang, F. Shen, N. Xie, R. Hong, and H. T. Shen, "Collective reconstructive embeddings for cross-modal hashing," *IEEE Trans. Image Process.*, vol. 28, no. 6, pp. 2770–2784, Jun. 2019.

[12] L. Zhu, Z. Huang, Z. Li, L. Xie, and H. T. Shen, "Exploring auxiliary context: Discrete semantic transfer hashing for scalable image retrieval," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 11, pp. 5264–5276, Nov. 2018.

[13] C. Li, C. Deng, N. Li, W. Liu, X. Gao, and D. Tao, "Self-supervised adversarial hashing networks for cross-modal retrieval," in *Proc. IEEE Int. Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2018, pp. 4242–4251.

[14] X. Shen, F. Shen, Q.-S. Sun, and Y.-H. Yuan, "Multi-view latent hashing for efficient multimedia search," in *Proc. ACM Int. Conf. Multimedia (MM)*, 2015, pp. 831–834.

[15] X. Liu, J. He, D. Liu, and B. Lang, "Compact kernel hashing with multiple features," in *Proc. ACM Int. Conf. Multimedi (MM)*, 2012, pp. 881–884.

[16] Y. Kang, S. Kim, and S. Choi, "Deep learning to hash with multiple representations," in *Proc. IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2012, pp. 930–935.

[17] L. Zhu, Z. Huang, X. Liu, X. He, J. Sun, and X. Zhou, "Discrete multimodal hashing with canonical views for robust mobile landmark search," *IEEE Trans. Multimedia*, vol. 19, no. 9, pp. 2066–2079, Sep. 2017.

[18] X. Lu, L. Zhu, J. Li, H. Zhang, and H. T. Shen, "Efficient supervised discrete multi-view hashing for large-scale multimedia search," *IEEE Trans. Multimedia*, to be published.

[19] J. Song, Y. Yang, Z. Huang, H. T. Shen, and J. Luo, "Effective multiple feature hashing for large-scale near-duplicate video retrieval," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 1997–2008, Dec. 2013.

[20] L. Liu, M. Yu, and L. Shao, "Multiview alignment hashing for efficient image search," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 956–966, Mar. 2015.

[21] X. Shen, F. Shen, L. Liu, Y.-H. Yuan, W. Liu, and Q.-S. Sun, "Multiview discrete hashing for scalable multimedia search," *TISTACM Trans. Intell. Syst. Technol.*, vol. 9, no. 5, pp. 1–21, Jun. 2018.

[22] R. Yang, Y. Shi, and X.-S. Xu, "Discrete multi-view hashing for effective image retrieval," in *Proc. ACM Int. Conf. Multimedia Retr. (ICMR)*, 2017, pp. 175–183.

[23] T. Baltrusaitis, C. Ahuja, and L.-P. Morency, "Multimodal machine learning: A survey and taxonomy," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 2, pp. 423–443, Feb. 2019.

[24] N. Li, C. Li, C. Deng, X. Liu, and X. Gao, "Deep joint semantic-embedding hashing," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2018, pp. 2397–2403.

[25] D. Wang, P. Cui, M. Ou, and W. Zhu, "Deep multimodal hashing with orthogonal regularization," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2015, pp. 2291–2297.

[26] F. Nie, G. Cai, J. Li, and X. Li, "Auto-weighted multi-view learning for image clustering and semi-supervised classification," *IEEE Trans. Image Process.*, vol. 27, no. 3, pp. 1501–1511, Mar. 2018.

[27] F. Nie, J. Li, and X. Li, "Self-weighted multiview clustering with multiple graphs," in *Proc. Int. Joint Conf. Artif. Intell. (IJCAI)*, 2017, pp. 2564–2570.

[28] J. Gui, T. Liu, Z. Sun, D. Tao, and T. Tan, "Fast supervised discrete hashing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 2, pp. 490–496, Feb. 2018.

[29] D. Hu, F. Nie, and X. Li, "Discrete spectral hashing for efficient similarity retrieval," *IEEE Trans. Image Process.*, vol. 28, no. 3, pp. 1080–1091, Mar. 2019.

[30] X. Luo, X.-Y. Yin, L. Nie, X. Song, Y. Wang, and X.-S. Xu, "SDMCH: Supervised discrete manifold-embedded cross-modal hashing," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, Jul. 2018, pp. 2518–2524.

[31] C.-X. Li *et al.*, "SCRATCH: A scalable discrete matrix factorization hashing for cross-modal retrieval," in *Proc. ACM Multimedia Conf. Multimedia Conf. (MM)*, 2018, pp. 1–9.

[32] S. Kim and S. Choi, "Multi-view anchor graph hashing," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2013, pp. 3123–3127.

[33] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*. [Online]. Available: http://arxiv.org/abs/1409.1556

[34] Y. Ko, "A study of term weighting schemes using class information for text classification," in *Proc. ACM Int. Conf. Res. Develop. Inf. Retr. (SIGIR)*, 2012, p. 1029–1030.

[35] X. Luo, Y. Wu, and X.-S. Xu, "Scalable supervised discrete hashing for large-scale search," in *Proc. World Wide Web Conf. World Wide Web (WWW)*, 2018, pp. 1603–1612

[36] Y. Xu, Y. Yang, F. Shen, X. Xu, Y. Zhou, and H. T. Shen, "Attribute hashing for zero-shot image retrieval," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Jul. 2017, pp. 133–138.

[37] F. Shen, Y. Xu, L. Liu, Y. Yang, Z. Huang, and H. T. Shen, "Unsupervised deep hashing with similarity-adaptive and discrete optimization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 3034–3044, Dec. 2018.

[38] M. Hu, Y. Yang, F. Shen, N. Xie, and H. T. Shen, "Hashing with angular reconstructive embeddings," *IEEE Trans. Image Process.*, vol. 27, no. 2, pp. 545–555, Feb. 2018.

[39] G. Ding, Y. Guo, and J. Zhou, "Collective matrix factorization hashing for multimodal data," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2014, pp. 2083–2090.

[40] D. Zhang and W. Li, "Large-scale supervised multimodal hashing with semantic correlation maximization," in *Proc. AAAI Int. Conf. Artif. Intell. (AAAI)*, 2014, pp. 2177–2183.

[41] Z. Lin, G. Ding, J. Han, and J. Wang, "Cross-view retrieval via probability-based semantics-preserving hashing," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4342–4355, Dec. 2017.

[42] X. Shen, W. Liu, I. W. Tsang, Q.-S. Sun, and Y.-S. Ong, "Multilabel prediction via cross-view search," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 9, pp. 4324–4338, Sep. 2018.

[43] Z. Chen, W. Yu, C. Li, L. Nie, and X. Xu, "Dual deep neural networks cross-modal hashing," in *Proc. AAAI Int. Conf. Artif. Intell. (AAAI)*, 2018, pp. 274–281.

[44] B. Wang, Y. Yang, X. Xu, A. Hanjalic, and H. T. Shen, "Adversarial cross-modal retrieval," in *Proc. ACM Multimedia Conf. (MM)*, 2017, pp. 154–162.

[45] Y. Hou, P. Zhang, X. Xu, X. Zhang, and W. Li, "Nonlinear dimensionality reduction by locally linear inlaying," *IEEE Trans. Neural Netw.*, vol. 20, no. 2, pp. 300–315, Feb. 2009.

[46] L. K. Saul and S. T. Roweis, "Think globally, fit locally: Unsupervised learning of low dimensional manifold," *J. Mach. Learn. Res.*, vol. 4, pp. 119–155, 2003.

[47] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Transfer independently together: A generalized framework for domain adaptation," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2144–2155, Jun. 2019.

[48] J. Li, K. Lu, Z. Huang, L. Zhu, and H. T. Shen, "Heterogeneous domain adaptation through progressive alignment," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 5, pp. 1381–1391, May 2019.

[49] C. Cui, H. Liu, T. Lian, L. Nie, L. Zhu, and Y. Yin, "Distribution-oriented aesthetics assessment with semantic-aware hybrid network," *IEEE Trans. Multimedia*, vol. 21, no. 5, pp. 1209–1220, May 2019.

[50] Z. Cheng, X. Chang, L. Zhu, R. C. Kanjirathinkal, and M. Kankanhalli, "MMALFM: Explainable recommendation by leveraging reviews and images," *TOISACM Trans. Inf. Syst.*, vol. 37, no. 2, pp. 16:1–16:28, Jan. 2019.

[51] F. Nie, H. Huang, X. Cai, and C. H. Q. Ding, "Efficient and robust feature selection via joint l21-norms minimization," in *Proc. Int. Conf. Neural Inf. Process. Syst. (NIPS)*, 2010, pp. 1813–1821.

[52] K. G. Murty, "Nonlinear programming theory and algorithms," *Technometrics*, vol. 49, no. 1, p. 105, Feb. 2007.

[53] Z. Lin, M. Chen, and Y. Ma, "The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices," *CoRR*, vol. abs/1009.5055, 2010. [Online]. Available: http://arxiv.org/abs/1009.5055

[54] H. Wang, F. Nie, and H. Huang, "Multi-view clustering and feature learning via structured sparsity," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2013, pp. 352–360.

[55] M. J. Huiskes and M. S. Lew, "The MIR flickr retrieval evaluation," in *Proc. 1st ACM Int. Conf. Multimedia Inf. Retr. (MIR)*, 2008, pp. 39–43.

[56] T. Lin *et al.*, "Microsoft COCO: Common objects in context," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2014, pp. 740–755.

[57] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng, "NUS-WIDE: A real-world Web image database from National University of Singapore," in *Proc. ACM Int. Conf. Image Video Retr. (CIVR)*, 2009.

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.

Lei Zhu received the B.S. degree from the Wuhan University of Technology in 2009, and the Ph.D. degree from the Huazhong University of Science and Technology in 2015. He is currently a Full Professor with the School of Information Science and Engineering, Shandong Normal University, China. He was a Research Fellow at The University of Queensland from 2016 to 2017, and at the Singapore Management University from 2015 to 2016. His research interests are in the area of large-scale multimedia content analysis and retrieval.

Xu Lu received the bachelor's degree from Shandong Normal University, China, in 2016, where she is currently pursuing the Ph.D. degree with the School of Information Science and Engineering. She is under the supervision of Prof. H. Zhang at Shandong Normal University since 2016.

Zhiyong Cheng received the B.E. and M.E. degrees from the Huazhong University of Science and Technology (HUST) and Xi'an Jiaotong University (XJTU), in 2007 and 2010, respectively, and the Ph.D. degree in computer science from Singapore Management University, Singapore. From 2014 to 2015, he was a visiting student at the School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA. In January 2019, he joined SDAI, Shandong Academy of Sciences, as a Professor. His research interests mainly focus on large-scale multimedia content analysis and retrieval.

Jingjing Li received the M.Sc. and Ph.D. degrees in computer science from the University of Electronic Science and Technology of China in 2013 and 2017, respectively. He is currently an Associate Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China. His research interest is in machine learning, especially transfer learning, subspace learning, and recommender systems.

Huaxiang Zhang received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2004. He was an Associate Professor with the Department of Computer Science, Shandong Normal University, Jinan, China, from 2004 to 2005, where he is currently a Professor with the School of Information Science and Engineering. His current research interests include machine learning, pattern recognition, evolutionary computation, and Web information processing.

Authorized licensed use limited to: University of Canberra. Downloaded on June 06,2020 at 03:37:42 UTC from IEEE Xplore. Restrictions apply.