Deep Factorized Multi-view Hashing for Image Retrieval

Chenyang Zhu, Wenjue He, and Zheng Zhang(✉)

Harbin Institute of Technology, Shenzhen 518055, China
darrenzz219@gmail.com

Abstract. Multi-view hashing has been paid much attention to due to its computational efficiency and lower memory overhead in similarity measurement between instances. However, a common drawback of these multi-view hashing methods is the lack of ability to fully explore the underlying correlations between different views, which hinders them from producing more discriminative hash codes. In our work, we propose the principled Deep Factorized Multi-view Hashing (DFMH) framework, including interpretable robust representation learning, multi-view fusion learning, and flexible semantic feature learning, to deal with the challenging multi-view hashing problem. Specifically, instead of directly projecting the features to a common representation space, we construct an adaptively weighted deep factorized structure to preserve the heterogeneity between different views. Furthermore, the visual space and semantic space are interactively learned to form a reliable hamming space. Particularly, the flexible semantic representation is obtained by learning regressively from semantic labels. Importantly, a well-designed learning strategy is developed to optimize the objective function efficiently. DFMH as well as compared methods is tested on benchmark datasets to validate the efficiency and effectiveness of our proposed method. The source codes of this paper are released at: https://github.com/chenyangzhu1/DFMH.

Keywords: Multi-view hashing · Deep factorization · Image retrieval · Learning to hash

1 Introduction

With the increasing usage of big data and multimedia, there has been rising need for large-scale data analytic methods, especially methods measuring the similarities or distances between items, with low computational complexity and memory usage. Hashing provides an elegant solution to this challenging problem by representing the original high-dimensional features as binary codes which preserve the original similarity of features.

C. Zhu and W. He—Co-first authors;
Supported by National Natural Science Foundation of China (62002085).

S.-H. Wang and Y.-D. Zhang (Eds.): ICMTEL 2022, LNICST 446, pp. 628–643, 2022.
https://doi.org/10.1007/978-3-031-18123-8_49

While some early methods [4,5,7] use fixed hash functions to obtain the binary codes, most of the recent methods [1,3,10,22] implement a learning-to-hash strategy, which borrows the ideas from machine learning and deep learning to obtain more flexible hash codes. These methods are no longer independent of data, and are expected to yield better results. More specifically, there are two types of learning-to-hash methods, *i.e.*, unsupervised and supervised, according to whether semantic information is used. Among them, supervised methods [12, 16] are usually preferred to unsupervised ones [15,24] since they explore the semantic information and thus can provide a more promising result.

However, all of the aforementioned methods are designed for single-view situations, which cannot solve the real-world problems that usually appear in a multi-view form. For instance, multiple features, such as HOG, GIST, and LBP, can be extracted from a picture, every feature is taken as a view. An event can be reported in different forms including texts, photos, and videos, and each of the forms is considered a view. The multi-view circumstances usually contain more information than the summation of the information within every single view, since connections between views provide additional information. Unfortunately, existing single-view algorithms are not able to utilize such complementary information, resulting in unsatisfying clustering results. Therefore, a number of multi-view hashing methods are put forward to explore the heterogeneous information under the multiple views. Utilizing non-negative matrix factorization (NMF), [8] fuses information from multiple views and drops the useless and even noisy ones. [20] solves the near-duplicate video retrieval (NDVR) problems by exploring information from multiple feature types extracted from video keyframes, and learns hash codes and the hash functions simultaneously to deal with the large-scale data. [17] measures the local similarities as well as the semantic similarities by a Locally Linear Embedding (LLE) based method.

Although there has been some encouraging progress in multi-view hashing, most existing methods still encounter many challenges, which prohibit them from providing accurate and discriminative hash codes. First, the conventional multi-view hashing always learns the common representations from multiple views based on feature projecting from the original feature space, which lacks enough interpretability for robust representation learning. Moreover, the underlying characteristics across multiple views are missing in common feature learning. Furthermore, the complementary property of multi-view learning is underexplored to generate a unified representation. Additionally, most of the existing works employ fixed discrete labels for classification or regression in hash code learning, which lacks enough flexibility in semantic knowledge discovery.

To deal with the drawbacks of existing methods mentioned above, a novel Deep Factorized Multi-view Hashing (DFMH) method is proposed to produce hash codes which not only preserve the latent characteristics between views, but also capture the categorical information from the regressive semantic labels (Fig. 1). Specifically speaking, we implement a novel deep factorized structure to learn a consensus representation space instead of using the traditional factorize method, and employ an orthogonal rotation strategy to convert the real values

into the hash codes. In this way, more characteristics, especially the complementary information between views, are incorporated in the learned hash codes. Furthermore, adaptively learned weights are applied to the real-value learning process to automatically make the trade-off between different views, making sure that informative views contribute more to the representation space and the noisier ones contribute less. Additionally, learning from semantic labels, instead of directly using fixed labels, ensures that the flexibility is preserved and underlying information is fully explored.

The main contributions of the DFMH method are listed as following:

1. We propose a novel multi-view hashing method to capture the hidden characteristics from multi-view data for discriminative multi-view information retrieval. In the method, deep factorized common representation learning, adaptively-weighted multi-view fusion, and flexible semantic feature learning are jointly considered in one unified learning framework.
2. An adaptively weighted deep factorized learning scheme is designed to explore the heterogeneous properties of multi-view data, and the consensus representations are further transformed to the discrete hash space based on an orthogonal rotation strategy.
3. The flexible semantic representation is learned based on the regressive semantic labels, which can guide the intrinsic semantic transfer in joint binary code learning.
4. Extensive experiments validate the effectiveness of the proposed method in multi-view fusion and semantic-preserving multi-view hashing compared with multiple state-of-the-art multi-view hashing methods.

2 Related Work

Multi-view hashing is an efficient method for multimedia retrieval. Unlike single-view methods which focus on one view only, multi-view hashing not only utilizes information within each individual view but also employs the complementary information between views to improve its hashing performance. Existing multi-view hashing methods can be classified into two types: those do not use labelled information, *i.e.*, unsupervised methods, and those take advantage of labelled information *i.e.*, supervised methods.

Unsupervised methods are intended to generate hash codes and hash functions independent of semantic labels. For example, [20] generates hash codes under individual structure as well as global structure. In [18], Shen *et al.* generate hash codes using matrix factorization methods with an adaptively learned kernel space. Going one step further, Lu *et al.* [13] propose a method that has no additional parameter for the regularization term to shorten training time, making it possible to deal with the retrieval of large-scale multi-view data. Although encouraging improvement has been made, hash codes generated without the supervision of semantic labels are usually not discriminative enough.

Contrarily, information underlying in the semantic labels has been proven very useful in discovering similarities between views, and thus many supervised

multi-view methods are proposed to make full use of it. For instance, [11] designs a model which combines different kernels of different features to generate hash codes. In [23], Yang *et al.* propose a novel method to explore the local as well as semantic similarity and optimizes the resulting objective function without relaxing the constraint. However, in most of the existing methods, different views share the same weight, while the real-world case is that some views usually contain more important information than others.

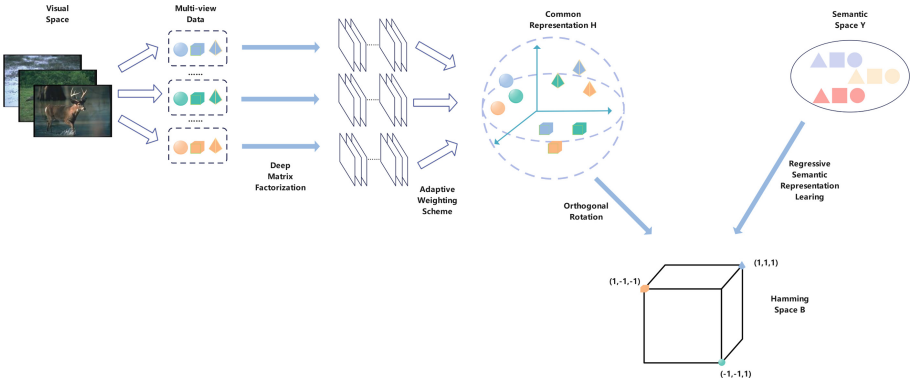


Fig. 1. The general structure of our DFMH method. The learning framework is composed of a deep matrix factorization scheme to extract the features from complex structures and an adaptive weighting scheme to balance each view. Moreover, our framework adopts orthogonal rotation strategy and regressive semantic representation learning to fuse visual and semantic information together in Hamming space.

3 The Proposed Method

3.1 Notation and Problem Definition

In this paper, matrices are represented by upper case letters such as $\boldsymbol{X}$ and vectors are represented by lowercase letters, such as $\boldsymbol{\alpha}$. The multi-view data training set is denoted as $\mathcal{X} = \left\{\boldsymbol{X}^{(v)}\right\}_{v=1}^{m}$, where $\boldsymbol{X}^{(v)} = [x_1^{(v)}, \cdots, x_n^{(v)}] \in \Re^{d_v \times n}$ is the feature matrix of the v-th view, d_v is the feature dimension of v-th view, and n is the number of instances. Based on this, each x_i has a corresponding label vector $y_i \in \{0,1\}^c$ indicating its class, where c is the class number and $y_{ij} = \begin{cases} 1 & \text{if } x_i \text{ is an instance of the j-th class} \\ 0 & \text{otherwise} \end{cases}$. $\boldsymbol{Y} = [y_1, \cdots, y_n] \in \Re^{c \times n}$ is the class label matrix. $\boldsymbol{B} = [b_1, \cdots, b_n] \in \{-1,1\}^{l \times n}$ is the learned hash code, where l is the hash code length. DFMH aims at learning the hash function $\mathcal{H} : \mathcal{X} \rightarrow \boldsymbol{B}$, which enables the proposed method to return a series of binary codes which preserves the similarity of original instances, *i.e.*, the more similarity instances share, the closer the distance between the learned hash codes of these instances is.

3.2 The Proposed Deep Factorized Multi-view Hashing Method

Unlike most of the aforementioned methods which usually use a shallow learning method, deep matrix factorization is employed to learn a consensus multi-view representation $\boldsymbol{H} \in \Re^{l \times n}$, where l is the length of the hash code.

Inspired by [21], we conduct matrix factorization on $\boldsymbol{X}^{(v)}$, which writes $\boldsymbol{X}^{(v)} \approx \boldsymbol{U}_1^{(v)} \boldsymbol{H}_1^{(v)}$, so that the latent representation of the first layer is obtained. After that, the first layer is represented as $\boldsymbol{H}_1^{(v)} = \boldsymbol{U}_2^{(v)} \boldsymbol{H}_2^{(v)}$ in the same manner. Such process is repeated for $(m-1)$ times to get $\boldsymbol{H}_{m-1}^{(v)}$.

The factorization process can be listed as following:

$$\begin{aligned} &\boldsymbol{X}^{(v)} \approx \boldsymbol{U}_1^{(v)} \boldsymbol{H}_1^{(v)} \\ &\boldsymbol{X}^{(v)} \approx \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \boldsymbol{H}_2^{(v)} \\ &\cdots \\ &\boldsymbol{X}^{(v)} \approx \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_{m-1}^{(v)} \boldsymbol{H}_{m-1}^{(v)} \\ &\text{s.t. } \boldsymbol{H}_i^{(v)} \geq 0. \end{aligned} \tag{1}$$

The last layer is specially treated in order to learn the final consensus multi-view representation $\boldsymbol{H}$:

$$\begin{aligned} &\boldsymbol{H}_{m-1}^{(v)} = \boldsymbol{U}_m^{(v)} \boldsymbol{H}, v = 1, 2, \cdots, m \\ &\text{s.t. } \boldsymbol{H} \geq 0. \end{aligned} \tag{2}$$

As such, the common representation learning process can be expressed in the following equation:

$$\min_{\boldsymbol{U}_i^{(v)}, \boldsymbol{H}} \sum_{v=1}^{m} \left\| \boldsymbol{X}^{(v)} - \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_m^{(v)} \boldsymbol{H} \right\|_F^2 \quad \text{s.t. } \boldsymbol{H} \geq 0, \tag{3}$$

where $\boldsymbol{U}_i^{(v)} \in \Re^{p_{i-1} \times p_i}$ is the layer basis matrix of the i-th layer and v-th view, $\boldsymbol{H}_m^{(v)}$ denotes the latent representation of the m-th layer and v-th view, and p_i represents the dimension of the i-th layer with $p_0 = d_v$, $p_m = r$.

Since the physical meanings of features vary from one to another, different weights should be assigned to different views to explore as much information in provided data as possible. To balance the information between views, an adaptive weighting scheme is proposed, which writes:

$$\begin{aligned} \min_{\boldsymbol{U}_i^{(v)}, \boldsymbol{H}, \alpha^{(v)}} & \sum_{v=1}^{m} \left(\alpha^{(v)} \right)^r \left\| \boldsymbol{X}^{(v)} - \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_m^{(v)} \boldsymbol{H} \right\|_F^2 \\ \text{s.t. } & \sum_v \alpha^{(v)} = 1, 0 < \alpha^{(v)} < 1, \boldsymbol{H} \geq 0, \end{aligned} \tag{4}$$

where $\alpha^{(v)}$ is the adaptive weighting parameter of the v-th view. r controls the distribution of the weight, and is usually set to 3 or 5.

The space rotation strategy is introduced to make sure that the real-valued common representation space $\boldsymbol{H}$ and the binary code space $\boldsymbol{B}$ are of enough similarity:

$$\begin{aligned}\min_{\boldsymbol{U}_i^{(v)},\boldsymbol{H},\alpha^{(v)},\boldsymbol{B},\boldsymbol{R}} & \sum_{v=1}^{m}\left(\alpha^{(v)}\right)^r\left\|\boldsymbol{X}^{(v)}-\boldsymbol{U}_1^{(v)}\boldsymbol{U}_2^{(v)}...\boldsymbol{U}_m^{(v)}\boldsymbol{H}\right\|_F^2 \\ & +\lambda\left\|\boldsymbol{B}-\boldsymbol{R}\boldsymbol{H}\right\|_F^2 \\ & \text{s.t. } \sum_v \alpha^{(v)}=1, 0<\alpha^{(v)}<1, \boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I}, \boldsymbol{H}\geq 0,\end{aligned} \tag{5}$$

where λ is a balance parameter and $\boldsymbol{R}$ is the orthogonal rotation matrix.

Notably, hash code generated from the Eq. (5) only contain the visual information, while the semantic information which is also significant is ignored. Therefore, a regressive semantic representation learning step is introduced to extract the flexible semantics from the fixed one-hot binary class label matrix $\boldsymbol{Y}$:

$$\begin{aligned}\min_{\boldsymbol{U}_i^{(v)},\boldsymbol{H},\alpha^{(v)},\boldsymbol{B},\boldsymbol{R},\boldsymbol{W}} & \sum_{v=1}^{m}\left(\alpha^{(v)}\right)^r\left\|\boldsymbol{X}^{(v)}-\boldsymbol{U}_1^{(v)}\boldsymbol{U}_2^{(v)}...\boldsymbol{U}_m^{(v)}\boldsymbol{H}\right\|_F^2 \\ & +\lambda\left\|\boldsymbol{B}-\boldsymbol{R}\boldsymbol{H}\right\|_F^2+\beta\left\|\boldsymbol{B}-\boldsymbol{W}\boldsymbol{Y}\right\|_F^2 \\ & \text{s.t. } \sum_v \alpha^{(v)}=1, 0<\alpha^{(v)}<1, \boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I}, \boldsymbol{H}\geq 0,\end{aligned} \tag{6}$$

where β is a trade-off parameter.

Adding a fourth term to avoid trivial solutions of regressive semantic encoding, the overall objective function in DFMH is shown below:

$$\begin{aligned}\min_{\boldsymbol{U}_i^{(v)},\boldsymbol{H},\alpha^{(v)},\boldsymbol{B},\boldsymbol{R},\boldsymbol{W}} & \sum_{v=1}^{m}\left(\alpha^{(v)}\right)^r\left\|\boldsymbol{X}^{(v)}-\boldsymbol{U}_1^{(v)}\boldsymbol{U}_2^{(v)}...\boldsymbol{U}_m^{(v)}\boldsymbol{H}\right\|_F^2 \\ & +\lambda\left\|\boldsymbol{B}-\boldsymbol{R}\boldsymbol{H}\right\|_F^2+\beta\left\|\boldsymbol{B}-\boldsymbol{W}\boldsymbol{Y}\right\|_F^2 \\ & +\gamma\left\|\boldsymbol{W}\right\|_F^2 \\ & \text{s.t. } \sum_v \alpha^{(v)}=1, 0<\alpha^{(v)}<1, \boldsymbol{R}^\top\boldsymbol{R}=\boldsymbol{I}, \boldsymbol{H}\geq 0.\end{aligned} \tag{7}$$

The first term of this function learns a common representation $\boldsymbol{H}$, the second term minimizes the gap between $\boldsymbol{H}$ and the binary code space $\boldsymbol{B}$, and the third and forth term obtain flexible semantics from the class label matrix $\boldsymbol{Y}$.

4 Optimization

Optimizing problem Eq. (7) is a non-convex problem whose closed-form solution can hardly be achieved. Therefore, a feasible iterative optimization strategy is employed to address the problem Eq. (7).

Step 1: Set other variables as constant values to update $\boldsymbol{U}_i^{(v)}$. The subproblem for $\boldsymbol{U}_i^{(v)}$ refers to:

$$\boldsymbol{C} = \left\| \boldsymbol{X}^{(v)} - \boldsymbol{\phi} \boldsymbol{U}_i^{(v)} \boldsymbol{H}_i^{(v)} \right\|_F^2 \text{ s.t. } \boldsymbol{H}_i^{(v)} \geq 0. \tag{8}$$

where $\boldsymbol{\phi} = \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_{i-1}^{(v)}$. Setting $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{U}_i^{(v)}} = 0$, $\boldsymbol{U}_{(i)}^v$ can be expressed in the following form:

$$\boldsymbol{U}_i^{(v)} = \boldsymbol{\phi}^\dagger \boldsymbol{X}^{(v)} \boldsymbol{H}_i^{(v)\dagger} \tag{9}$$

where $\boldsymbol{X}^\dagger$ means the pseudo inverse of matrix $\boldsymbol{X}$.

Step 2: Set other variables as constant values to update $\boldsymbol{H}_i^{(v)}$: The optimization of $\boldsymbol{U}_i^{(v)}$ requires an up-to-date $\boldsymbol{H}_i^{(v)}$, so here we update $\boldsymbol{H}_i^{(v)}$ though it does not appear in the final objective function. The subproblem for $\boldsymbol{H}_i^{(v)}$ refers to:

$$\boldsymbol{C} = \| \boldsymbol{X}^{(v)} - \boldsymbol{\Phi} \boldsymbol{H}_i^{(v)} \|_F^2 \text{ s.t. } \boldsymbol{H}_i^{(v)} \geq 0. \tag{10}$$

where $\boldsymbol{\Phi} = \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_i^{(v)}$. Following [2], $\boldsymbol{H}_i^{(v)} (i < m)$ is updated with the following equation:

$$\boldsymbol{H}_i^{(v)} = \boldsymbol{H}_i^{(v)} \odot \sqrt{\frac{[\boldsymbol{\Phi}^\top \boldsymbol{X}^{(v)}]^+ + [\boldsymbol{\Phi}^\top \boldsymbol{\Phi} \boldsymbol{H}_i^{(v)}]^-}{[\boldsymbol{\Phi}^\top \boldsymbol{X}^{(v)}]^- + [\boldsymbol{\Phi}^\top \Phi \boldsymbol{H}_i^{(v)}]^+}}, \tag{11}$$

where $[\boldsymbol{A}]^+ = (|\boldsymbol{A}| + \boldsymbol{A})/2, [\boldsymbol{A}]^- = (|\boldsymbol{A}| - \boldsymbol{A})/2$.

Step 3: Set other variables as constant values to update $\boldsymbol{W}$. The subproblem for $\boldsymbol{W}$ refers to:

$$\boldsymbol{C} = \beta \| \boldsymbol{B} - \boldsymbol{W}\boldsymbol{Y} \|_F^2 + \gamma \| \boldsymbol{W} \|_F^2 \tag{12}$$

Setting $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{W}} = 0$, the following solution for $\boldsymbol{W}$ is obtained:

$$\boldsymbol{W} = \beta \boldsymbol{B}\boldsymbol{Y}^\top (\beta \boldsymbol{Y}\boldsymbol{Y}^\top + \gamma \boldsymbol{I})^{-1} \tag{13}$$

Step 4: Set other variables as constant values to update $\boldsymbol{R}$. The subproblem for $\boldsymbol{R}$ refers to:

$$\begin{aligned} &\max_{\boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}} tr(\boldsymbol{R}^\top \boldsymbol{B} \boldsymbol{H}^\top) \\ &\text{s.t. } \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}, \boldsymbol{H} \geq 0. \end{aligned} \tag{14}$$

According to [9], the solution for $\boldsymbol{R}$ is:

$$\boldsymbol{R} = \boldsymbol{P}_R \boldsymbol{Q}_R^\top \tag{15}$$

where $\boldsymbol{P}_R$ is a matrix constructed of the left-singular vectors of $\boldsymbol{B}\boldsymbol{H}^\top$, and $\boldsymbol{Q}_R$ is a matrix constructed of the right-singlular vectors of $\boldsymbol{B}\boldsymbol{H}^\top$.

Step 5: Set other variables as constant values to update $\boldsymbol{H}$. The subproblem for $\boldsymbol{H}$ refers to:

$$\begin{aligned} \boldsymbol{C} = & \sum_{v=1}^{m} \left(\alpha^{(v)}\right)^r \left\| \boldsymbol{X}^{(v)} - \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_m^{(v)} \boldsymbol{H} \right\|_F^2 \\ & + \lambda \left\| \boldsymbol{B} - \boldsymbol{R}\boldsymbol{H} \right\|_F^2 \\ & \text{s.t. } \sum_v \alpha^{(v)} = 1, 0 < \alpha^{(v)} < 1, \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}, \boldsymbol{H} \geq 0. \end{aligned} \tag{16}$$

By setting $\frac{\partial \boldsymbol{C}}{\partial \boldsymbol{H}} = 0$, the following solution can be obtained:

$$\boldsymbol{H} = (\sum_{v=1}^{m} (\alpha^{(v)})^r \boldsymbol{Z}^\top \boldsymbol{Z} + \lambda \boldsymbol{R}^\top \boldsymbol{R})^{-1} (\sum_{v=1}^{m} (\alpha^{(v)})^r \boldsymbol{Z}^\top \boldsymbol{X} + \lambda \boldsymbol{R}^\top \boldsymbol{B}) \tag{17}$$

where $\boldsymbol{Z} = \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_m^{(v)}$.

Step 6: Set other variables as constant values to update $\boldsymbol{B}$. The subproblem for $\boldsymbol{B}$ refers to:

$$\begin{aligned} \max_{\boldsymbol{B} \in \{-1,1\}} & tr(\boldsymbol{B}^\top (2\lambda \boldsymbol{R}\boldsymbol{H} + 2\beta \boldsymbol{W}\boldsymbol{Y})) \\ & \text{s.t. } \boldsymbol{R}^\top \boldsymbol{R} = \boldsymbol{I}, \boldsymbol{H} \geq 0. \end{aligned} \tag{18}$$

The solution for $\boldsymbol{B}$ is:

$$\boldsymbol{B} = sgn(2\lambda \boldsymbol{R}\boldsymbol{H} + 2\beta \boldsymbol{W}\boldsymbol{Y}). \tag{19}$$

Step 7: Set other variables as constant values to update $\alpha^{(v)}$. For convenience, we set $h_v = \left\| \boldsymbol{X}^{(v)} - \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_m^{(v)} \boldsymbol{H} \right\|_F^2$. The subproblem for α refers to:

$$\begin{aligned} \min_{\alpha} & \sum_{v=1}^{m} (\alpha^{(v)})^r h_v \\ & \text{s.t. } \sum_v \alpha^{(v)} = 1, 0 < \alpha^{(v)} < 1. \end{aligned} \tag{20}$$

We can then construct the Lagrange function as following:

$$L(\alpha, \varepsilon) = \sum_{v=1}^{m} (\alpha^{(v)})^r h_v - \varepsilon (\sum_{v=1}^{m} \alpha^{(v)} - 1), \tag{21}$$

then we need

$$\bigtriangledown_{\alpha^{(v)}} L = r(\alpha^{(v)})^{r-1} h_v - \varepsilon = 0 \tag{22}$$

$$\bigtriangledown_{\varepsilon} L = \sum_{v=1}^{m} \alpha^{(v)} - 1 = 0, \tag{23}$$

where $(v = 1, 2 \cdots m)$. Combining Eq. (22) and Eq. (23), we have:

$$\sum_{v=1}^{m} (\frac{\varepsilon}{rh_v})^{\frac{1}{r-1}} = 1, \tag{24}$$

which can be reformulated as

$$(\frac{\varepsilon}{r})^{\frac{1}{r-1}} = \frac{1}{\sum_{v=1}^{m} (\frac{1}{h_v})^{\frac{1}{r-1}}}. \tag{25}$$

Thus, Eq. (25) can be transformed into:

$$\alpha^{(v)} = \frac{(\frac{1}{h_v})^{\frac{1}{r-1}}}{\sum_{v=1}^{m} (\frac{1}{h_v})^{\frac{1}{r-1}}}. \tag{26}$$

Notably, the adaptive weight $\alpha^{(v)}$ is determined by different multi-view data $\mathcal{X}$.

The proposed optimization algorithm is run for several times until its convergence. According to our experiments, the number of iterations needed is usually less than 10. The whole learning process of DFMH is concluded in Algorithm 1.

Algorithm 1. DFMH

Input: $\mathcal{X}$: set of multi-view data; $\boldsymbol{Y}$: Label set; λ, β, γ: parameters; h: length of binary codes; T: maximum number of iterations.

Output: $\alpha^{(v)}$: Adaptive weight parameters; $\boldsymbol{U}_1^{(v)}\boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_m^{(v)}$: layer basis matrices; $\boldsymbol{B}$: ideal binary code.

Initialize $\boldsymbol{H}$, $\boldsymbol{U}_i^{(v)}$, $\boldsymbol{B}$, $\boldsymbol{R}$ by random matrix.
for Not Convergence or k=1:T **do**
  Update $\boldsymbol{U}_i^{(v)}$ via solving Eq. (9)
  Update $\boldsymbol{H}_i^{(v)} (i < m)$ via solving Eq. (11)
  Update $\boldsymbol{W}$ via solving Eq. (13)
  Update $\boldsymbol{R}$ via solving Eq. (15)
  Update $\boldsymbol{H}$ via solving Eq. (17)
  Update $\boldsymbol{B}$ via solving Eq. (19)
  Update $\alpha^{(v)}$ via solving Eq. (26)
end for

5 Nonlinear Feature Embedding and Out-of-Sample Cases

5.1 Nonlinear Feature Embeddings

High-quality multi-view features are indispensable in the generation of accountable hash codes. However, to avoid redundancies and noise in the handcraft features is of great difficulty, in this work we employ a kernelized method to solve

this problem. According to [15], Gaussian Mixed Model (GMM) makes great contribution to the production of nonlinear feature embeddings. The function writes:

$$\varphi(x_i^{(v)}) = [exp(-\left\|x_I^{(v)} - a_1^{(v)}\right\|^2/\sigma_v), \cdots, exp(-\left\|x_i^{(v)} - a_k^{(v)}\right\|^2/\sigma_v)]^\top, \quad (27)$$

where $\varphi(x_i^{(v)}) \in \Re^{k\times 1}$ is a feature of the i-th sample from v-th view learned using the Radial Basis Function (RBF) kernel. $a_i^{(v)}(i = 1, 2, \cdots, k)$ is k randomly selected anchor samples in view v. σ_m is the m-th view's kernel width which means the median of the distance between $a^{(v)}$ and $\boldsymbol{X}^{(v)}$. Replacing $\boldsymbol{X}^{(v)}$ by nonlinear feature $\varphi(\boldsymbol{X}^{(v)}) = [\varphi(x_1^{(v)}), \cdots, \varphi(x_n^{(v)})] \in \Re^{k\times n}$, the above optimization methods can be applied to this learning algorithm.

5.2 Out-of-Sample Extension

In this section, we extend the proposed DFMH method to out-of-sample cases, in which data are not included in the training stage. Generally, a two-step method is adopted to generate the final hash function [9,25]. Typically, for the given training samples, we can obtain their common latent representation $\boldsymbol{H}$ by using Eq. (17). Based on this, we can formulate the hash function from the latent representation $\boldsymbol{H}$ to the objective binary code matrix $\boldsymbol{B}$ by using the following simple linear regression function:

$$\min_{\boldsymbol{W}} \left\|\boldsymbol{W}^\top \boldsymbol{H} - \boldsymbol{B}\right\|_F^2 + \|\boldsymbol{W}\|_F^2, \quad (28)$$

which has closed-form solution:

$$\boldsymbol{W} = (\boldsymbol{H}\boldsymbol{H}^\top + \boldsymbol{I})^{-1}\boldsymbol{H}\boldsymbol{B}^\top. \quad (29)$$

Similarly, for any query sample $\boldsymbol{x}_q^{(v)}$, we first embed it into the nonlinear feature vector $\varphi(\boldsymbol{x}_q^{(v)})$ by using Eq. (27). Subsequently, we learn the shared latent representation $\hat{h}$ of the multi-view query samples based on the following equation:

$$\hat{h} = \sum_{v=1}^{m} (\alpha^{(v)})^r \boldsymbol{U}_1^{(v)} \boldsymbol{U}_2^{(v)} \cdots \boldsymbol{U}_m^{(v)} \varphi(\boldsymbol{x}_q^{(v)}), \quad (30)$$

Hence, the corresponding hash codes can be learned by using:

$$b = sgn(\boldsymbol{W}^\top \hat{h}). \quad (31)$$

6 Experiments

6.1 Datasets

To evaluate our model, we conduct experiments on two publicly available benchmark datasets, *i.e.*, CIFAR-10 [6] and MIRFlickr [8]. CIFAR-10[1] is consisted of

[1] https://www.cs.toronto.edu/~kriz/cifar.html.

Table 1. MAP values with different code length on **CIFAR-10** and **MIRFlickr**

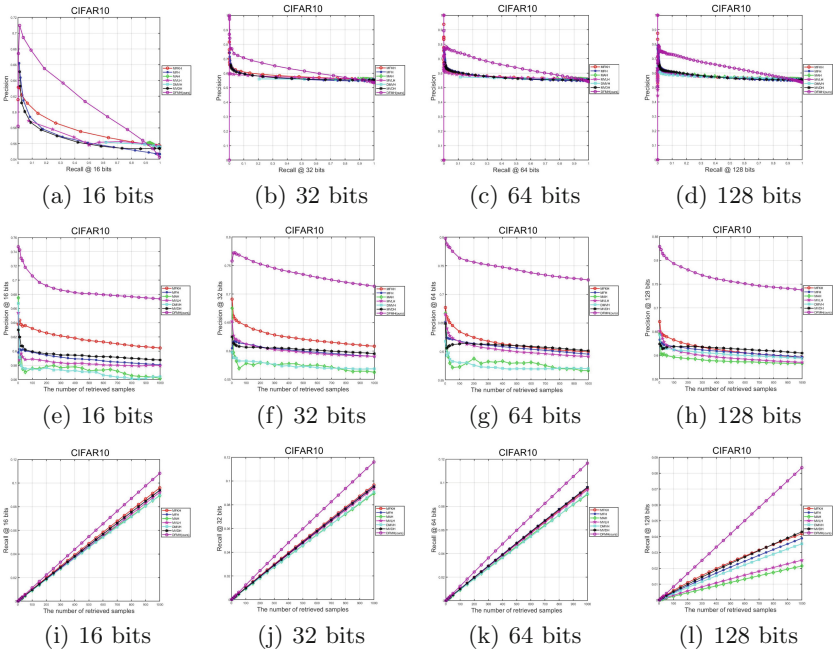
CIFAR-10					
Methods	8	16	32	64	128
MFKH [11]	0.1943	0.1892	0.1654	0.1467	0.1330
MFH [20]	0.1235	0.1349	0.1443	0.1649	0.1798
MAH [8]	0.1051	0.1117	0.1104	0.1089	0.1509
MVLH [19]	0.1103	0.1189	0.1238	0.1146	0.1103
DMVH [23]	0.1162	0.1510	0.1484	0.1549	0.1599
MVDH [17]	0.1524	0.1691	0.1796	0.1877	0.1952
DFMH (ours)	**0.3590**	**0.4265**	**0.4820**	**0.5892**	**0.6362**
MIRFlickr					
Methods	8	16	32	64	128
MFKH [11]	0.5763	0.5805	0.5827	0.5773	0.5745
MFH [20]	0.5596	0.5643	0.5673	0.5692	0.5703
MAH [8]	0.5626	0.5622	0.5629	0.5653	0.5714
MVLH [19]	0.5638	0.5670	0.5744	0.5729	0.5700
DMVH [23]	0.5673	0.5623	0.5619	0.5631	0.5739
MVDH [17]	0.5660	0.5693	0.5723	0.5745	0.5766
DFMH (ours)	**0.6176**	**0.6414**	**0.6701**	**0.6753**	**0.6936**

10 classes with 6000 images each, *i.e.*, 60000 images in total. Three features, *i.e.*, LBP, Gist, and HOG, with dimensions of 1450, 1024, and 1152 respectively, are extracted as 3 views to evaluate the performances of the methods. Following [9], 12500 labeled images from 38 categories are extracted from MIRFlickr dataset[2] in our experiment. Gist (512 dimensions), Hue histogram (100 dimensions) and SIFT bag of visual words (BoVW) (1000 dimensions) serve as three views.

6.2 Experimental Settings and Evaluation Matrics

Our DFMH method is compared with six different state-of-the-art multi-view hashing methods, *i.e.*, MFKH [11], MFH [20], MAH [8], MVLH [19], DMVH [23], and MVDH [17]. All these experiments are conducted on MATLEB R2021a on a computer with Intel (R) Core (TM) i5-9300HF CPU @ 2.40 GHz and 16 GB RAM. The original code of the compared methods are from the corresponding authors. The parameters in each compared method are set to the recommended values if it is mentioned in corresponding papers, or in other cases, default values. For the CIFAR-10 dataset, 1000 samples are randomly selected from 10 views with 100 samples from each view as a query set. Similarly, MIRFlickr is separated into a query set with 1000 images and a retrieval set containing the rest of the images. Two pictures are considered relevant if at least one label exists in both

[2] http://lear.inrialpes.fr/people/guillaumin/data.php.



(a) 16 bits (b) 32 bits (c) 64 bits (d) 128 bits

(e) 16 bits (f) 32 bits (g) 64 bits (h) 128 bits

(i) 16 bits (j) 32 bits (k) 64 bits (l) 128 bits

Fig. 2. Precision-recall curves, Precision @TopN curves, and Recall @TopN curves of DFMH as well as compared methods on the CIFAR-10 with different code lengths

of their label sets. We randomly select 1000 images as anchor graphs for anchor-based methods. Gaussian kernel is used for the kernel-based methods [9].

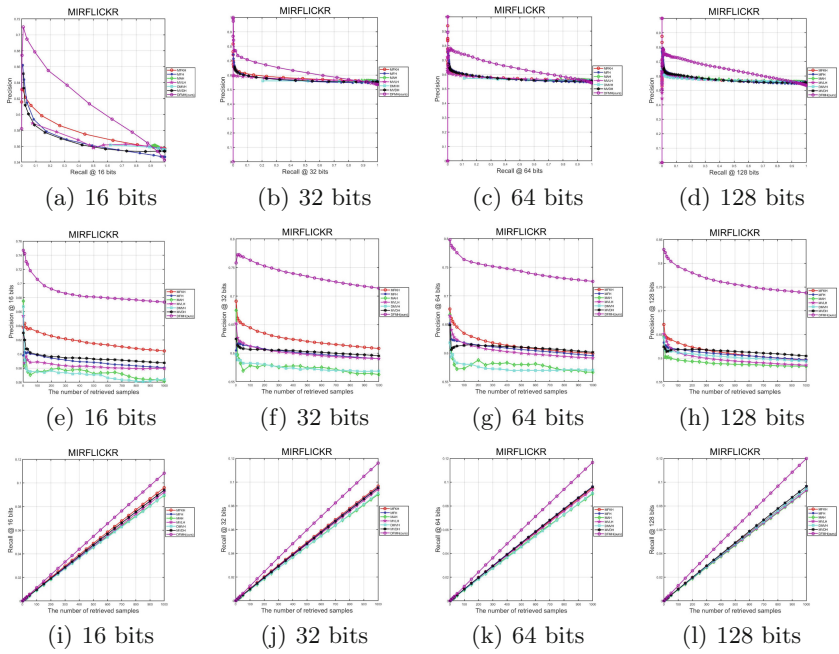
Following previous works [14,26], we use metrics including mean average precision (mAP), precision-recall curves, recall @TopN and precision @TopN to evaluate the performances of tested methods.

6.3 Experimental Results

Table 1 reports the mAPs of DFMH as well as compared algorithms on CIFAR-10 and MIRFlickr when code length is 8, 16, 32, 64, and 128, respectively. The following observations can be made from Table 1. 1) With the increment of code length, the mAPs generally rise, which is intuitive since longer hash codes usually contain more information. 2) Due to the lack of semantic information, unsupervised methods normally have inferior performances than the supervised ones. 3) MFKH performs the best among all the compared methods. 4) Our method defeats all the other methods dramatically on both datasets. For CIFAR-10, when the hash code is consisted of 8 bits, our method has an mAP 16.47% higher than that of the second-placed method MFKH, and the mAP of our method is more than two times as high as that of the second highest method in the 16, 32, 64 or 128 bits situations. On the MIRFlickr dataset, DFMH defeats all the compared methods, and as the hash code length increases, the advantage of our

method over the compared methods becomes even larger. For instance, when the hash codes is only 8 bits, the mAP of our method is 4.13% higher than the second highest method, while when the hash code is lengthened to 128 bits, our method outperforms the second-placed method by 11.7%. This indicates that with the same code length increment, our method incorporates more additional information.

The precision-recall relationships of all the tested methods with different length of hash code on CIFAR-10 and MIRFlickr are reported in figure (a)–(d) in Fig. 2 and Fig. 3, respectively. The P-R curve of our method lies on the upper right side of the other curves in almost all situations, which strongly justifies the effectiveness of our proposed method. The relationship between precision and the number of retrieved images on both datasets are illustrated in figure (e)–(h) in Fig. 2 and Fig. 3, and figure (i)–(l) demonstrate how recall change with the number of retrieved images. It can be concluded from the graphs that with the number of image retrieved fixed, our method always yields a better precision and recall.

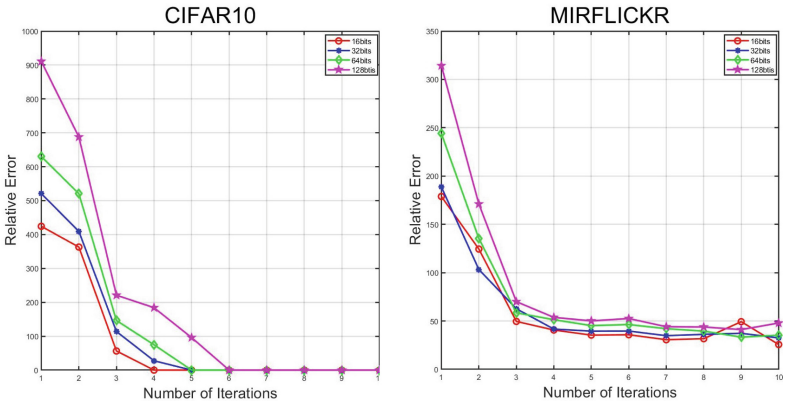


(a) 16 bits (b) 32 bits (c) 64 bits (d) 128 bits

(e) 16 bits (f) 32 bits (g) 64 bits (h) 128 bits

(i) 16 bits (j) 32 bits (k) 64 bits (l) 128 bits

Fig. 3. Precision-recall curves, Precision @TopN curves, and Recall @TopN curves of DFMH as well as compared methods on the MIRFlickr with different code lengths



(a) CIFAR-10 (b) MIRFlickr

Fig. 4. How retrieval error of DFMH change with iteration times on CIFAR-10 and MIRFlickr when the code lengths are different.

6.4 Convergence and Computational Complexity Analysis

Figure 4 shows how $\|\boldsymbol{B}_t - \boldsymbol{B}_{t-1}\|_F$ of different code length changes with the number of iterations, where $\boldsymbol{B}_t$ denotes $\boldsymbol{B}$ gained from the t-th iteration. It can be observed that the curve has a downward trend, while in the first two iterations the difference drops sharply, and in the following iterations the curve flattens down until the final convergence. These figures also show that our objective function converges within a reasonable number of iterations, which validates the proposed optimization method.

Table 2 compares the running time of different methods on MIRFlickr learning a 32-bit hash code. While outperforming other methods on mAP, our method is also among the fastest methods which are able to finish training within 10 seconds. This shows that our method keeps a balance between efficiency and effectiveness.

Table 2. Comparison of training time on **MIRFlickr** dataset with 32-bit hash code.

Methods	Training time (s)
MFKH [11]	2.03
MVLH [19]	5.13
MFH [20]	6.42
MVDH [17]	13.14
MAH [8]	62.15
DMVH [23]	408.97
DFMH (ours)	9.69

7 Conclusion

We proposed a novel *Deep Factorized Multi-view Hash* (DFMH) method for efficient and effective multi-view image retrieval in this work. DFMH first learned a common representation space through an adaptively weighted deep factorization scheme and then fused it with the semantic information space to learn a hamming space. Furthermore, a feasible algorithm was put forward to optimize the non-convex problem. The superiority of the proposed DFMH method in both accuracy and computational complexity was validated by extensive experiments on benchmark datasets.

References

1. Chen, Y., Wang, S., Lu, J., Chen, Z., Zhang, Z., Huang, Z.: Local graph convolutional networks for cross-modal hashing. In: Proceedings of the 29th ACM International Conference on Multimedia, pp. 1921–1928 (2021)
2. Ding, C.H.Q., Li, T., Jordan, M.I.: Convex and semi-nonnegative matrix factorizations. IEEE Trans. Pattern Anal. Mach. Intell. **32**(1), 45–55 (2008)
3. Liong, V.E., Lu, J., Wang, G., Moulin, P., Zhou, J.: Deep hashing for compact binary codes learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2475–2483 (2015)
4. Gorisse, D., Cord, M., Precioso, F.: Locality-sensitive hashing for Chi2 distance. IEEE Trans. Pattern Anal. Mach. Intell. **34**(2), 402–409 (2011)
5. Ji, J., Li, J., Yan, S., Zhang, B., Tian, Q.: Super-bit locality-sensitive hashing. In: Advances in Neural Information Processing Systems, vol. 25 (2012)
6. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
7. Kulis, B., Grauman, K.: Kernelized locality-sensitive hashing. IEEE Trans. Pattern Anal. Mach. Intell. **34**(6), 1092–1104 (2011)
8. Liu, L., Mengyang, Yu., Shao, L.: Multiview alignment hashing for efficient image search. IEEE Trans. Image Process. **24**(3), 956–966 (2015)
9. Liu, L., Zhang, Z., Huang, Z.: Flexible discrete multi-view hashing with collective latent feature learning. Neural Process. Lett. **52**(3), 1765–1791 (2020)
10. Liu, W., Wang, J., Kumar, S., Chang, S.-F.: Hashing with graphs. In: International Conference on Machine Learning (2011)
11. Liu, X., He, J., Liu, D., Lang, B.: Compact kernel hashing with multiple features. In: Proceedings of the 20th ACM International Conference on Multimedia, pp. 881–884 (2012)
12. Liu, X., Yadong, M., Zhang, D., Lang, B., Li, X.: Large-scale unsupervised hashing with shared structure learning. IEEE Trans. Cybern. **45**(9), 1811–1822 (2014)
13. Lu, X., Zhu, L., Li, J., Zhang, H., Shen, H.T.: Efficient supervised discrete multi-view hashing for large-scale multimedia search. IEEE Trans. Multimedia **22**(8), 2048–2060 (2019)
14. Schütze, H., Manning, C.D., Raghavan, P.: Introduction to Information Retrieval, vol. 39. Cambridge University Press, Cambridge (2008)
15. Shen, F., Shen, C., Liu, W., Shen, H.T.: Supervised discrete hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 37–45 (2015)

16. Shen, F., Xu, Y., Liu, L., Yang, Y., Huang, Z., Shen, H.T.: Unsupervised deep hashing with similarity-adaptive and discrete optimization. IEEE Trans. Pattern Anal. Mach. Intell. **40**(12), 3034–3044 (2018)
17. Shen, X., Shen, F., Liu, L., Yuan, Y.-H., Liu, W., Sun, Q.-S.: Multiview discrete hashing for scalable multimedia search. ACM Trans. Intell. Syst. Technol. **9**(5), 1–21 (2018)
18. Shen, X., Shen, F., Sun, Q.-S., Yuan, Y.-H.: Multi-view latent hashing for efficient multimedia search. In: Proceedings of the 23rd ACM International Conference on Multimedia, pp. 831–834 (2015)
19. Shen, X., Shen, F., Sun, Q.-S., Yuan, Y.H.: Multi-view latent hashing for efficient multimedia search. In: Proceedings of the 23rd ACM International Conference on Multimedia, pp. 831–834 (2015)
20. Song, J., Yang, Y., Huang, Z., Shen, H.T., Luo, J.: Effective multiple feature hashing for large-scale near-duplicate video retrieval. IEEE Trans. Multimedia **15**(8), 1997–2008 (2013)
21. Trigeorgis, G., Bousmalis, K., Zafeiriou, S., Schuller, B.W.: A deep matrix factorization method for learning attribute representations. IEEE Trans. Pattern Anal. Mach. Intell. **39**(3), 417–429 (2016)
22. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: Advances in Neural Information Processing Systems, vol. 21 (2008)
23. Yang, R., Shi, Y., Xu, X.-S.: Discrete multi-view hashing for effective image retrieval. In: Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval, pp. 175–183 (2017)
24. Zhang, Z., et al.: Scalable supervised asymmetric hashing with semantic and latent factor embedding. IEEE Trans. Image Process. **28**(10), 4803–4818 (2019)
25. Zhang, Z., Luo, H., Zhu, L., Lu, G., Shen, H.T.: Modality-Invariant asymmetric networks for cross-modal hashing. IEEE Trans. Knowl. Data Eng. (2022). https://doi.org/10.1109/TKDE.2022.3144352
26. Zhang, Z., Xie, G., Li, Y., Li, S., Huang, Z.: SADIH: semantic-aware discrete hashing. Proc. AAAI Conf. Artif. Intell. **33**, 5853–5860 (2019)