Applied Intelligence (2023) 53:28954–28973
https://doi.org/10.1007/s10489-023-05028-y

Deep cross-modal hashing with fine-grained similarity

Yangdong Chen[1] · Jiaqi Quan[1] · Yuejie Zhang[1] · Rui Feng[1] · Tao Zhang[2]

Accepted: 18 September 2023 / Published online: 19 October 2023

Abstract
Cross-modal hashing has attracted noticeable attention in the multimedia community. Two-stage methods often show impressive performance by first learning hash codes for data instances from different modalities, then learning hash functions that map the original multimodal data to the low dimensional hash codes. However, most existing two-stage methods can hardly obtain satisfactory hash codes at the first stage, as the commonly used coarse-grained similarity matrix fails to capture the differentiated similarity relationships between the original data instances. Besides, such methods cannot obtain satisfactory hash functions at the second stage, where the learning of hash functions is treated as a multi-binary classification problem. In this paper, we propose a novel two-stage hashing method for cross-modal retrieval. At the first stage, we capture the differentiated similarity relationships between data instances by designing a fine-grained similarity matrix and add an Autoencoder to mine the semantic information. At the second stage, we introduce a similarity sensitivity learning strategy under the guidance of the similarity matrix to train the hash functions. This strategy makes the training process sensitive to the similar and hard pairs, boosting the retrieval performance. Comprehensive experiments on three benchmark datasets validate the effectiveness of our method.

Keywords Cross-modal retrieval · Deep hashing · Fine-grained similarity · Focal loss

1 Introduction

With the explosive growth of multimedia data, multimodal applications are attracting increasing attention, such as Image Captioning [17], Visual Question Answering [39], and Cross-modal Retrieval [27, 32, 44, 56]. Unlike unimodal retrieval [1], cross-modal retrieval aims to search semantically similar samples from one modality with a given query from another modality (e.g., from image to text). However, it is nearly impossible to perform exact nearest neighbor search for real-world large-scale multimedia data due to the cost of time and storage involved. Many methods have been proposed to address this problem, such as Approximate Nearest Neighbor (ANN) search. Hashing is one of the most promising techniques among the ANN-based methods [35, 45]. Recently, hashing-based ANN search has become popular because of its low computational cost and fast retrieval speed. The aim of hashing is to learn hash functions to project data instances from the original feature space into a compact Hamming space, where the data instances from different modalities can be represented by binary hash codes of a certain length. In the Hamming space, the similarity between original data instances can be represented by the Hamming distance. This requires the hash codes to preserve the similarity relationships between the data instances in the original feature space.

Data instances from different modalities are heterogeneous. Their features may have different distributions. To narrow such a gap, many cross-modal hashing methods have been proposed to mine the semantic relationships

✉ Yuejie Zhang
yjzhang@fudan.edu.cn

Yangdong Chen
chenyd19@fudan.edu.cn

Jiaqi Quan
18210240157@fudan.edu.cn

Rui Feng
fengrui@fudan.edu.cn

Tao Zhang
taozhang@mail.shufe.edu.cn

[1] School of Computer Science, Shanghai Key Lab of Intelligent Information Processing, Fudan University, Yangpu, Shanghai 200433, China

[2] School of Information Management and Engineering, Shanghai University of Finance and Economics, Yangpu, Shanghai 200433, China

between different modalities and preserve them in a common Hamming space. According to whether supervised information (i.e., semantic labels) is used or not, recent existing methods can be generally divided into unsupervised hashing [46] and supervised hashing [55]. Supervised methods usually achieve superior performance to unsupervised methods by learning the correlations between multimodal data via semantic labels. However, most of them are shallow hashing methods which use hand-craft features. The hand-crafted features may not be optimal discriminative representations of data instances.

Deep learning based methods have meanwhile achieved promising performance. Typical deep learning based methods, such as DVSH [5], CMDVH [16] and TVDH [42], train hash codes and hash functions simultaneously. They are also called one-stage hashing methods with end-to-end training scheme. One advantage of such methods is that the joint optimization of hash codes and hash functions can lead to better retrieval accuracy compared to two-stage methods. However, the hash codes must be retrained when the hash functions are changed. Moreover, the alternating training of hash codes and hash functions makes them impact each other, potentially leading to suboptimal performance. The pairwise-based [19, 21, 55] or triplet-based [12] training strategies also limit them to preserving only local similarities due to computational constraints.

To improve the flexibility and address the above limitations, two-stage methods are proposed, which decouple the training of hash codes and hash functions [22, 26, 31, 48]. Hash codes are first learned independently. Hash functions are then trained to map new queries to the pre-trained hash codes. This makes it easier to optimize hash codes and hash functions separately. At first, the retrieval accuracy of such methods is not as good as that of the one-stage methods. Nearly all of the two-stage methods only use a coarse-grained similarity matrix $\boldsymbol{S}^c = s_{ij}^c$, where $s_{ij}^c = 1$ means that the data instances x_i^c and x_j^c are similar and $s_{ij}^c = 0$ means dissimilar. Obviously, it will cause substantial loss of information before generating the hash codes, because this similarity matrix cannot capture the differentiated similarity relationships between the training data at the first stage. At the second stage, most of the two-stage methods treat the learning of hash functions as a multi-binary classification problem. In fact, each bit of a hash code does not necessarily correspond to a partition in the feature space. The learning of hash functions at this stage should focus on preserving as much discriminative information as possible. In other words, this should be a result of similarity calculation rather than classification. In addition, separate training is no longer limited to mini-batches, enabling preservation of global similarities.

Based on the above observations, we propose a novel two-stage method for cross-modal retrieval, called Deep Cross-modal Hashing with Fine-grained Similarity (DFGH). It effectively explores the differentiated similarity relationships between the training data and utilizes them to assist the learning of hash codes and hash functions. When performing hash code inference at the first stage, we use semantic labels to build a fine-grained similarity matrix by assuming two instances are more similar if they share more labels. In this way, we can represent the differentiated similarity relationships between the training data. When performing hash code mapping at the second stage, we design a similarity sensitivity strategy to learn the hash functions. In this strategy, we employ a mapping loss to fit the hash codes and a classification loss to retain the semantic information. Importantly, we further design a similarity preserving loss based on the above fine-grained similarity matrix to preserve the differentiated similarity information.

The main contributions of our work are summarized as:

1) We propose a novel two-stage hashing method for cross-modal retrieval with the fine-grained similarity matrix. The learning of hash codes and hash functions are linked up without affecting each other by utilizing the matrix.
2) At the first stage of the method, we preserve the distinguishable similarity relationships between data instances with the similarity matrix. At the second stage, the similarity-preserving loss is designed so that the hash functions can preserve the fine-grained similarity information.
3) We conduct extensive experiments on three benchmark datasets to demonstrate the effectiveness of the proposed DFGH. The results show its superiority to the state-of-the-art methods especially on large-scale image datasets (e.g., MS-COCO).

2 Related work

Different from unimodal hashing, cross-modal hashing aims to realize the retrieval when queries and samples to be retrieved have different modalities. It is crucial to narrow the semantic gap between different modalities when extending unimodal hashing to cross-modal hashing. Cross-modal hashing methods can be divided into two categories in terms of whether supervised information is used during the learning process, i.e., unsupervised and supervised.

Unsupervised methods learn hash functions by capturing the correlations between different modalities without supervised information. One common way is to preserve the structure of original features. For example, Inter-Media Hashing (IMH) [43] learned hash functions by exploring both intra-media consistency and inter-media consistency. PDH [40] maintained the predictability of binary hash codes and optimized the objective function by iterative method. CMFH [14] learned hash codes by collective matrix factorization

with a latent factor model from different modalities. LSSH [57] used sparse coding to capture the salient structure of images and matrix factorization to obtain the latent concepts of texts, and then projected the learned semantic features into a common space. MSAE [47] learned hash functions by capturing the intra-media and inter-media semantic relationships between data from heterogeneous sources. LCMH [58] aimed at learning hash functions from large-scale training datasets, while considering both the intra-similarity in each modality and the inter-similarity across different modalities. Fusion Similarity Hashing (FSH) [32] modeled the fusion similarity among different modalities by constructing an undirected asymmetric graph. Unsupervised Knowledge Distillation (UKD) [18] utilized a similarity matrix as well. As an unsupervised method, it estimated the similarity matrix using the original features of image and text, i.e., it utilized the distance between the features of instances. Our similarity matrix in this paper is different from theirs, and our method focuses on the utilization of supervision information, i.e., labels of instances.

Supervised methods explore the correlations between heterogeneous data to train hash functions using supervised information. Thus, they can usually obtain better accuracy than unsupervised methods. For example, Semantic Correlation Maximization (SCM) [53] tried to train hash codes by a semantic similarity matrix that was calculated by label vectors. Cross-Modality Similarity-Sensitive Hashing (CMSSH) [4] modeled the learning of hash functions as a binary classification problem and used a boost algorithm during the learning process. CVH [24] learned hash functions by projecting similar data instances from different modalities into similar hash codes.

Many deep hashing methods have been proposed recently. Most of them are called one-stage methods, which means that they combine the learning of hash codes and hash functions in an end-to-end manner. For example, Deep Cross-Modal Hashing (DCMH) [21] integrated the learning of deep features and hash codes into an end-to-end framework, and used pairwise-based object function to train the framework. PRDH [52] integrated different types of pairwise constraints to learn the hash codes. Deep Semantic-Preserving Ordinal Hashing (DSPOH) [23] learned hash functions by using deep neural networks to explore the ranking structure of feature dimensions. Self-Supervised Adversarial Hashing (SSAH) [25] used two adversarial networks to maximize the semantic correlation and consistency of representations between different modalities. Asymmetric Supervised Consistent and Specific Hashing (ASCSH) [37] presented a multimodal mapping learning strategy and a discrete asymmetric learning framework. It extracted the pairwise similarity and semantic labels to make full use of the supervised information. Multi-Level Correlation Adversarial Hashing (MLCAH) [36] designed a multi-level correlation hashing model with a label-consistency attention mechanism to excavate the local and global correlation information. Deep Multiscale Fusion Hashing (DMFH) [38] adopted multiscale fusion models to enhance semantic relevance and improved hash code representativeness. Asymmetric Correlation Quantization Hashing (ACQH) [37] used real-valued query embeddings and stacked database quantization embeddings for improved retrieval. Fast Discriminative Discrete Hashing (FDDH) [33] presented an efficient closed-form solution and online strategy to learn discriminative hash codes. It introduced an orthogonal basis to regress hash codes and relaxed label values to minimize quantization loss, and an orthogonal transform to ensure semantic consistency. Modality-Invariant Asymmetric Networks (MIAN) [54] introduced probabilistic alignment to capture intra- and inter-modal similarities, bridging semantic and heterogeneity gaps. Multi-Label Semantic Supervised Graph Attention Hashing (MS^{2}GAH) [15] designed an end-to-end framework that integrates graph attention networks and multi-label annotations to bridge the semantic modality gap.

Different from the one-stage methods, two-stage methods divide the learning of hash codes and hash functions into two stages. The pioneering framework [28] decomposed the hash learning into two stages. Semantics-Preserving Hashing (SePH) [31] generated unified binary hash codes by minimizing the KL-divergence, and then learned hash functions by kernel logistic regression with a sampling strategy. Supervised Discrete Manifold-Embedded Cross-Modal Hashing (SDMCH) [34] generated hash codes by using semantic labels. It exploited a nonlinear manifold structure of data and constructed the correlations among multiple heterogeneous modalities. Two-Step Cross-modal Hashing (TECH) [10] learned hash codes by preserving the similarity in the original space and exploiting the label correlations in the label space at the first stage, and then learned hash functions by leveraging the similarity relationships between training instances at the second stage. Discrete Latent Factor Hashing (DLFH) [22] designed a discrete latent factor model to learn the hash codes and proposed a method called out-of-sample extension to the hash function. Fast Cross-Modal Hashing (FCMH) [49] used global and local similarity embedding and efficient discrete optimization for improved accuracy and scalability. Compared with those one-stage methods, two-stage methods are flexible enough to change the hash functions if needed. They take less time to train hash codes and hash functions, and can achieve comparable retrieval performance. However, existing two-stage methods attempt to map similar data instances to the same hash code, and assume that two data instances are similar as long as they share one label. Such coarse-grained similarity definition will produce suboptimal hash codes and impact the retrieval accuracy. We propose DFGH which defines fine-grained similarity to generate better hash codes and hash functions at

the first and second stage respectively, by considering that the more labels two data instances share, the more similar they are.

In many one-stage methods, cross-entropy [21] is adopted to preserve the pairwise similarity relationships between data instances in the Hamming space. Besides, different types of loss functions are proposed. For example, Transfer Adversarial Hashing (TAH) [7] adopted a hybrid deep architecture which incorporated a pairwise t-distribution cross-entropy loss to learn hash codes. HashGAN [6] used a cosine cross-entropy loss to learn hash functions. However, most of them treat the hard and easy examples equally and enforce the same penalization on them, which may lead to suboptimal hash codes. To address this issue, some previous methods [8, 51] attempted to assign higher weights to more important examples for prioritization. However, they just assigned the priority via the coarse-grained similarity information. In comparison, our DFGH transfers the cross-entropy loss from one-stage to two-stage framework to preserve the similarity relationships in the Hamming space, and designs a fine-grained similarity matrix to assign different weights to different pairs, making the learning of hash functions more sensitive to similar pairs.

3 Methodology

3.1 Formulation

Without loss of generality, we focus on cross-modal retrieval for bimodal data (i.e., image and text), as other modalities should be similar. Firstly, we introduce the notations and problem definition. We use bold uppercase letter $\boldsymbol{W}$ to represent a matrix and bold lowercase letter $\boldsymbol{w}$ to represent a vector. $\boldsymbol{W}^{\mathrm{T}}$ represents the transpose of the matrix $\boldsymbol{W}$; $||\cdot||_F$ represents the Frobenius norm of a matrix; $\boldsymbol{W}_{i*}$ denotes the i-th row of $\boldsymbol{W}$; $\mathbf{1}$ denotes a vector with all elements being 1; and $sign(\cdot)$ denotes a sign function, which is defined as:

$$sign(x) = \begin{cases} -1 & x < 0 \\ +1 & x \geq 0 \end{cases} \tag{1}$$

Let $O = o_i{}_{i=1}^{n}$ denote a cross-modal dataset with n instances, where $o_i = (\boldsymbol{x}_i, \boldsymbol{y}_i, \boldsymbol{l}_i)$ is the i-th instance. Specifically, $\boldsymbol{x}_i \in \mathbb{R}^{1\times d_x}$ represents the original image feature; $\boldsymbol{y}_i \in \mathbb{R}^{1\times d_y}$ represents the original text feature; and $\boldsymbol{l}_i = [\boldsymbol{l}_{i1}, \ldots, \boldsymbol{l}_{ic}]$ is the multi-label assigned to o_i, where c is the number of classes. If o_i belongs to the j-th class, $\boldsymbol{l}_{ij} = 1$, otherwise $\boldsymbol{l}_{ij} = 0$. Hence, we can use $\boldsymbol{X}$, $\boldsymbol{Y}$, $\boldsymbol{L}$ to represent the image-feature matrix, text-feature matrix, and semantic label matrix, respectively. We use $\boldsymbol{B}^{\boldsymbol{x}}$ and $\boldsymbol{B}^{\boldsymbol{y}}$ to represent the matrices composed of the binary hash codes corresponding to the image and text feature, respectively. Table 1 summarizes the main notations used in this paper.

Table 1 Main notations

Notations	Explanation
$O = o_i{}_{i=1}^{n}$	The dataset
$\boldsymbol{X} \in \mathbb{R}^{n\times d_x}$	Image feature matrix
$\boldsymbol{Y} \in \mathbb{R}^{n\times d_x}$	Text feature matrix
$\boldsymbol{L} \in \{0,1\}^{n\times c}$	Semantic label matrix
$\boldsymbol{B}^{\boldsymbol{x}} \in \{-1,1\}^{n\times k}$	Image hash code matrix
$\boldsymbol{B}^{\boldsymbol{y}} \in \{-1,1\}^{n\times k}$	Text hash code matrix
$\boldsymbol{S} \in \mathbb{R}^{n\times n}$	Fine-grained similarity matrix
$\boldsymbol{W} \in \mathbb{R}^{c\times k}$	Transformation matrix
n	Length of dataset
d_x	Dimension of image features
d_y	Dimension of text features
c	Number of classes
k	Length of hash codes

The objective of DFGH is to learn two sets of modality-specific hash functions to map the features from different modalities into a common Hamming space, i.e., $\boldsymbol{f}^{\boldsymbol{x}}(\boldsymbol{x};\theta_x) \in \{-\mathbf{1},\mathbf{1}\}^k$ for images and $\boldsymbol{f}^{\boldsymbol{y}}(\boldsymbol{y};\theta_y) \in \{-\mathbf{1},\mathbf{1}\}^k$ for texts, where k denotes the length of hash code and θ denotes the parameters of the hash function to be learned. In the Hamming space, the similarity between two binary hash codes is evaluated by the Hamming distance. For two binary hash codes $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, there is a natural relationship between the Hamming distance and their inner product, which is defined as:

$$H_{dist}(\boldsymbol{b}_i, \boldsymbol{b}_j) = \frac{1}{2}\left(k - \boldsymbol{b}_i\boldsymbol{b}_j^{\mathrm{T}}\right) \tag{2}$$

The larger the inner product is, the smaller the Hamming distance is. Thus, we can use the inner product instead of the Hamming distance to quantify the similarity between two binary hash codes in the Hamming space.

3.2 Overview

The architecture of DFGH is shown in Fig. 1, which contains two main components, i.e., Hash Code Inference with Fine-grained Similarity and Hash Code Mapping with Similarity Sensitivity. In essence, what we need to do is to generate hash codes for the training set at the first stage, and then at the second stage, utilize these generated hash codes to supervise the training of the hash functions. At the first stage, a fine-grained similarity matrix is designed to generate a set of discriminative hash codes which preserve the differentiated similarity relationships between the training data. Besides, we use an Autoencoder to encourage hash codes to preserve

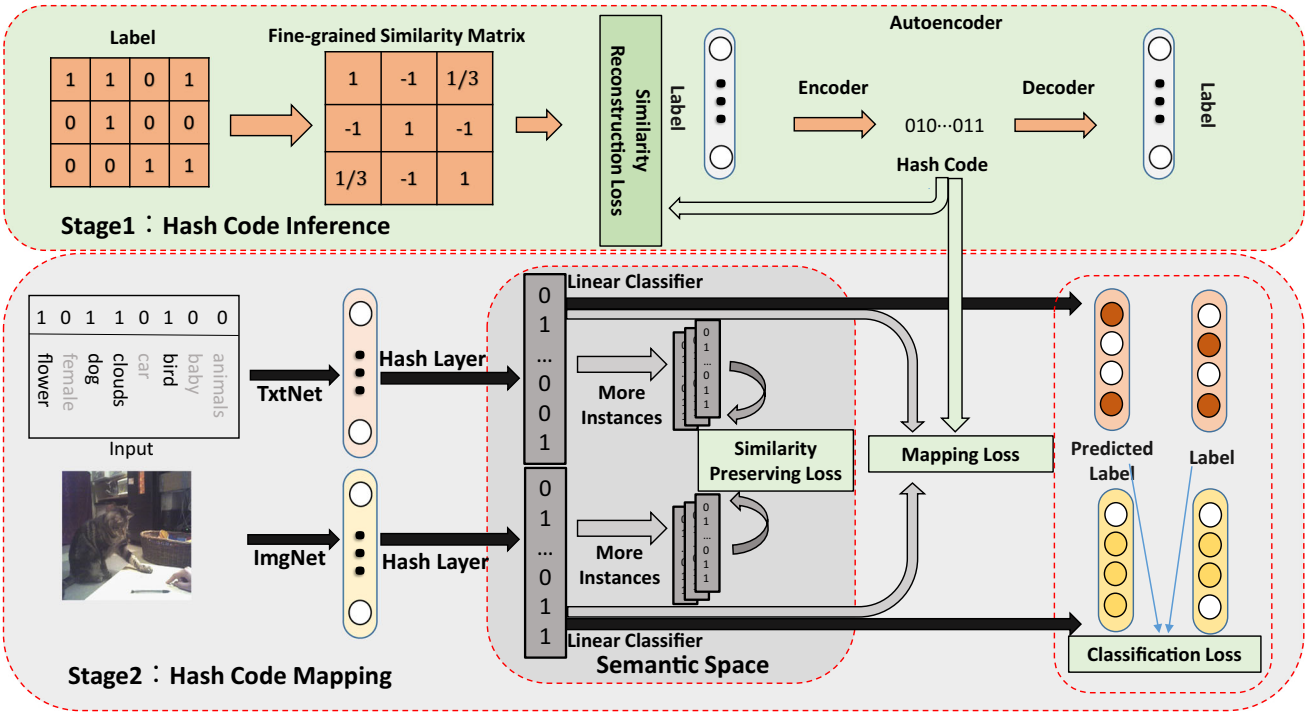


Fig. 1 The overall architecture of our proposed DFGH model. Refer to Section 3 for detail

the semantic information of labels. At the second stage, we employ two deep neural networks (ImgNet and TxtNet) to learn hash functions for the image and text modality, respectively. The output dimension of both networks is the length of hash code. We also design a similarity-preserving loss based on the similarity information (i.e., our similarity matrix), to enable the training of hash functions being more sensitive to the pairs with higher degree of similarity.

3.3 Hash code inference with fine-grained similarity

Fine-grained similarity mining At this stage, we need to infer a set of hash codes which can best preserve the differentiated similarities between the original training data. Firstly, we construct a fine-grained similarity matrix with the semantic labels to represent the differentiated similarities. As analyzed previously, most hashing methods just use a coarse-grained similarity matrix $\bar{S}$ to describe the similarities between the original data instances. Specifically, in a multi-label setting where a data instance o_i can be associated with multiple labels, it defines $\bar{S}_{ij} = 1$ if o_i and o_j share at least one label, indicating they are semantically similar, and $\bar{S}_{ij} = 0$ otherwise, where $i, j \in \{1, \ldots, n\}$. However, discrete values cannot fully describe the differentiated similarity information. As shown in Fig. 2, when data instances A and B have common labels, they will be mapped to the same point in the Hamming space, irrespective of their potential associations with different semantic label vectors. Besides, despite the fact that a data instance C shares more labels with A than with B, C and B will be mapped to the same point in the Hamming space. It reveals that the similarity between C and A is the same as the similarity between B and A in the Hamming space, though C is semantically closer to A in the original feature space.

To tackle these issues, we introduce a fine-grained similarity matrix $\hat{S}$, with its elements defined as follows:

$$\hat{S}_{ij} = \frac{\boldsymbol{l}_i \boldsymbol{l}_j^{\mathrm{T}}}{\mathbf{1}\boldsymbol{l}_i^{\mathrm{T}} + \mathbf{1}\boldsymbol{l}_j^{\mathrm{T}} - \boldsymbol{l}_i \boldsymbol{l}_j^{\mathrm{T}}} \tag{3}$$

where $\boldsymbol{l}_i$ and $\boldsymbol{l}_j$ are semantic label vectors of data instances o_i and o_j, respectively. The value of $\hat{S}_{ij}$ is larger if o_i and o_j share more labels, which indicates that o_i and o_j have a higher degree of similarity. It is obvious that the value of $\hat{S}_{ij}$ is 0 if o_i and o_j share no common labels. For the convenience of calculation, we make the hash codes belong to $\{-1, 1\}^k$, so their correct inner product for the dissimilar pairs should be closer to -1 instead of 0. To construct the relationship between the similarity matrix and the dot product of hash codes, we convert $\hat{S}_{ij}$ from $[0, 1]$ to $[-1, 1]$. We then get the final fine-grained similarity matrix $\boldsymbol{S}$ as:

$$\boldsymbol{S}_{ij} = \begin{cases} \hat{S}_{ij} & \hat{S}_{ij} > 0 \\ -1 & \hat{S}_{ij} = 0 \end{cases} \tag{4}$$

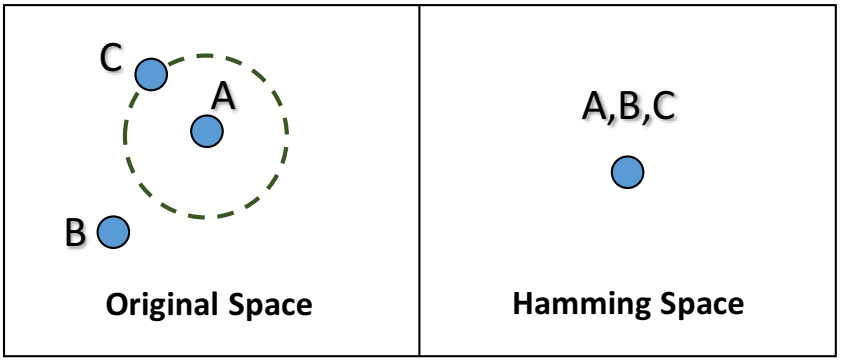


Fig. 2 An illustration of using coarse-grained similarity

When the instances o_i and o_j share at least one label, which means that they have a certain degree of similarity, we set the value of $\boldsymbol{S}_{ij}$ to be greater than 0 and expect that the Hamming distance between the two hash codes is not too large. Conversely, if no labels are shared, we aim for a maximized Hamming distance between the hash codes. We use the inner product of two hash codes to reconstruct the original relationship and define the similarity reconstruction loss as:

$$\min_{\boldsymbol{B}} \left\| \frac{1}{k}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} - \boldsymbol{S} \right\|_F^2 \quad \boldsymbol{s.t.}\ \ \boldsymbol{B} \in \{-1,\ 1\}^{n\times k} \tag{5}$$

The designed loss pushes the generated hash codes to be discriminative and to preserve the differentiated similarities between the training data.

Semantic information mining To enable the hash codes to preserve the semantic information of labels, we consider to establish connection between the hash codes and the semantic labels.

$$\boldsymbol{B} = \boldsymbol{L}\boldsymbol{W}, \quad \boldsymbol{L} = \boldsymbol{B}\boldsymbol{W}^{\mathrm{T}}, \quad \boldsymbol{W} \in \mathbb{R}^{c\times k} \tag{6}$$

This additional Autoencoder pushes the hash codes to preserve the information contained in the original semantic labels. Besides, the learning of hash codes is supervised by the fine-grained similarity reconstruction loss. Thus, there are no adverse consequences even when the number of nodes in the hidden layer (i.e., the number of bits in each hash code) exceeds the number of nodes in the input layer (i.e., the number of bits in each semantic label) within the Autoencoder. Our aim is to minimize the reconstruction error, and the objective function is formulated as:

$$\min_{\boldsymbol{W}} \left\| \boldsymbol{L} - \boldsymbol{L}\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} \right\|_F^2 \quad \boldsymbol{s.t.}\ \ \boldsymbol{L}\boldsymbol{W} = \boldsymbol{B} \tag{7}$$

This enables the generated hash codes to preserve the semantic information of labels. We use a learnable parameter α to decide the importance of reconstruction. Then, the total loss function of the first stage is defined as:

$$\begin{aligned} \min_{\boldsymbol{B},\boldsymbol{W}} \ & \left\| \frac{1}{k}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} - \boldsymbol{S} \right\|_F^2 + \alpha \left\| \boldsymbol{L} - \boldsymbol{L}\boldsymbol{W}\boldsymbol{W}^{\mathrm{T}} \right\|_F^2 \\ & \boldsymbol{s.t.}\, \boldsymbol{B} \in \{-1,\ 1\}^{n\times k}, \boldsymbol{L}\boldsymbol{W} = \boldsymbol{B} \end{aligned} \tag{8}$$

Thus we obtain optimal hash codes which preserve the differentiated similarity relationships between training data and the semantic information of labels.

Optimization Next, we proceed to derive the optimization process for the objective function. To optimize (8), we first rewrite it as:

$$\begin{aligned} \min_{\boldsymbol{B},\boldsymbol{W}} \ & \left\| \frac{1}{k}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} - \boldsymbol{S} \right\|_F^2 + \alpha \left\| \boldsymbol{L} - \boldsymbol{B}\boldsymbol{W}^{\mathrm{T}} \right\|_F^2 + \lambda \left\| \boldsymbol{B} - \boldsymbol{L}\boldsymbol{W} \right\|_F^2 \\ & \boldsymbol{s.t.}\, \boldsymbol{B} \in \{-1,\ 1\}^{n\times k} \end{aligned} \tag{9}$$

where λ is a parameter used to adjust the weight of the last term. Equation (9) is a Mixed Integer Programming (MIP) problem, which is usually NP-hard due to the binary constraints. To address this issue, we convert the binary constraints $\boldsymbol{B} \in \{-1, 1\}^{n\times k}$ into two conditions $\boldsymbol{B} \in S_b$ and $\boldsymbol{B} \in S_p$, where S_b and S_p are the sets of $[-1, 1]^{n\times k}$ and $\left\{B \middle| \|B\|_F^2 = nk\right\}$, respectively [41]. We introduce two variables $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ to absorb these constraints. Therefore, the constrains in (9) can be converted into $\boldsymbol{B} = \boldsymbol{Z}_1, \boldsymbol{Z}_1 \in S_b$ and $\boldsymbol{B} = \boldsymbol{Z}_2,\ \boldsymbol{Z}_2 \in S_p$. Building on such transformation, we design an Alternating Direction of Multipliers (ADMM) algorithm [3], and iteratively solve the augmented Lagrangian function of (9) as:

$$\begin{aligned} \min_{\boldsymbol{B},\boldsymbol{W},\boldsymbol{Z}_1,\boldsymbol{Z}_2,\boldsymbol{Y},\boldsymbol{Y}_2} \ & \left\| \frac{1}{k}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}} - \boldsymbol{S} \right\|_F^2 + \alpha \left\| \boldsymbol{L} - \boldsymbol{B}\boldsymbol{W}^{\mathrm{T}} \right\|_F^2 \\ & + \lambda \left\| \boldsymbol{B} - \boldsymbol{L}\boldsymbol{W} \right\|_F^2 \\ & + \delta_{S_b}(\boldsymbol{Z}_1) + \delta_{S_p}(\boldsymbol{Z}_2) + \frac{\rho_1}{2} \left\| \boldsymbol{B} - \boldsymbol{Z}_1 \right\|_F^2 \\ & + \frac{\rho_2}{2} \left\| \boldsymbol{B} - \boldsymbol{Z}_2 \right\|_F^2 \end{aligned}$$

$$+\mathrm{Tr}\left(\boldsymbol{Y}_1^{\mathrm{T}}(\boldsymbol{B}-\boldsymbol{Z}_1)\right)$$
$$+\mathrm{Tr}\left(\boldsymbol{Y}_2^{\mathrm{T}}(\boldsymbol{B}-\boldsymbol{Z}_2)\right) \tag{10}$$

where $\delta_S(\boldsymbol{Z})$ is an indicator function which equals to zero if $\boldsymbol{Z} \in S$ and $+\infty$ otherwise; $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ are the dual variables; ρ_1 and ρ_2 are the penalty variables. The specific process for updating these variables is shown as below.

1) $\boldsymbol{B}$-Step: Through fixing all the variables except $\boldsymbol{B}$, we can update $\boldsymbol{B}$ by minimizing the following objective functions:

$$\begin{aligned}\min_{\boldsymbol{B}} \mathcal{L} = & \left\|\frac{1}{k}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}-\boldsymbol{S}\right\|_F^2+\alpha\left\|\boldsymbol{L}-\boldsymbol{B}\boldsymbol{W}^{\mathrm{T}}\right\|_F^2+\lambda\left\|\boldsymbol{B}-\boldsymbol{L}\boldsymbol{W}\right\|_F^2 \\ & +\delta_{S_b}(\boldsymbol{Z}_1)+\delta_{S_p}(\boldsymbol{Z}_2)+\frac{\rho_1}{2}\left\|\boldsymbol{B}-\boldsymbol{Z}_1\right\|_F^2+\frac{\rho_2}{2}\left\|\boldsymbol{B}-\boldsymbol{Z}_2\right\|_F^2 \\ & +\mathrm{Tr}\left(\boldsymbol{Y}_1^{\mathrm{T}}(\boldsymbol{B}-\boldsymbol{Z}_1)\right)+\mathrm{Tr}\left(\boldsymbol{Y}_2^{\mathrm{T}}(\boldsymbol{B}-\boldsymbol{Z}_2)\right)\end{aligned} \tag{11}$$

Such a sub-problem can be iteratively solved by the LBFGS-B method, with the gradient calculated as:

$$\begin{aligned}\frac{\partial\mathcal{L}}{\partial\boldsymbol{B}} = & \frac{4}{k^2}\boldsymbol{B}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}-\frac{4}{k}\boldsymbol{S}\boldsymbol{B}+2\alpha\left(\boldsymbol{B}\boldsymbol{W}^{\mathrm{T}}-\boldsymbol{L}\right)\boldsymbol{W} \\ & +2\lambda\left(\boldsymbol{B}-\boldsymbol{L}\boldsymbol{W}\right)+\left(\rho_1+\rho_2\right)\boldsymbol{B}+\boldsymbol{Y}_1+\boldsymbol{Y}_2 \\ & -\rho_1\boldsymbol{Z}_1-\rho_2\boldsymbol{Z}_2\end{aligned} \tag{12}$$

2) $\boldsymbol{Z}$-Step: Through fixing all the variables except $\boldsymbol{Z}$, we can update $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ by the proximal minimizing methods. In particular, we eliminate $\delta_{S_b}(\boldsymbol{Z}_1)$ and $\delta_{S_p}(\boldsymbol{Z}_2)$ in (10) first and set the derivative with respect to $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ to 0. We then get the closed-form solutions of $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$, and project $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ into the sets S_b and S_p, respectively. The updating process for $\boldsymbol{Z}_1$ and $\boldsymbol{Z}_2$ is formulated as:

$$\boldsymbol{Z}_1 = \mathbf{5}_{S_b}\left(\boldsymbol{B}+\frac{1}{\rho_1}\boldsymbol{Y}_1\right) \tag{13}$$

$$\boldsymbol{Z}_2 = \mathbf{5}_{S_p}\left(\boldsymbol{B}+\frac{1}{\rho_2}\boldsymbol{Y}_2\right) = \sqrt{nk}\frac{\boldsymbol{B}+\frac{1}{\rho_2}\boldsymbol{Y}_2}{\left\|\boldsymbol{B}+\frac{1}{\rho_2}\boldsymbol{Y}_2\right\|} \tag{14}$$

where $\mathbf{5}_{S_b}$ and $\mathbf{5}_{S_p}$ are two projection operators, $\mathbf{5}_{S_b}$ projects values above 1 to 1 and values below -1 to -1., and $\mathbf{5}_{S_p}$ forces $\boldsymbol{Z}_2$ to satisfy the condition $\|\boldsymbol{Z}_2\|_F^2 = nk$.

3) $\boldsymbol{Y}$-Step: Through fixing all the variables except $\boldsymbol{Y}$, we can update $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ by performing gradient ascent on the dual problem, which is formulated as:

$$\boldsymbol{Y}_1 = \boldsymbol{Y}_1+\boldsymbol{\rho}_1\left(\boldsymbol{B}-\boldsymbol{Z}_1\right) \tag{15}$$

$$\boldsymbol{Y_2} = \boldsymbol{Y}_2+\boldsymbol{\rho}_2\left(\boldsymbol{B}-\boldsymbol{Z}_2\right) \tag{16}$$

4) $\boldsymbol{W}$-Step: Through fixing all the variables except $\boldsymbol{W}$, we can update $\boldsymbol{W}$ by taking the derivative of (10) with respect to $\boldsymbol{W}$ and setting it to 0. Thus, we obtain the following equation:

$$\boldsymbol{\alpha}\boldsymbol{W}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{B}+\boldsymbol{\lambda}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{L}\boldsymbol{W} = \boldsymbol{\alpha}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{B}+\boldsymbol{\lambda}\boldsymbol{L}^{\mathrm{T}}\boldsymbol{B} \tag{17}$$

We can solve such a Sylvester equation efficiently by the Bartels-Stewart algorithm [2].

The above iterative optimization is described in Algorithm 1. Here, the *sign* function is not utilized until the final hash code is obtained in the last step. Therefore, the derivation of *sign* function is not involved in this process.

Algorithm 1 The learning strategy of Hash Code Inference.

Input: Labels of training instances $\{\boldsymbol{l}_i\}_{i=1}^n$; Code length k; Parameters $\alpha, \gamma, \rho_1, \rho_2$
Output: Binary hash code for the training instances $\boldsymbol{B}$

```
Randomly initialize B.
Set Y_1 = B and Y_2 = B.
repeat
    Update B by solving (11) with the LBFGS-B algorithm;
    Update Z_1 and Z_2 with (13) and (14);
    Update Y_1 and Y_2 with (15) and (eq16);
    Update W by solving (17) with the Bartels-Stewart algorithm;
until convergence
return B
Get the binary hash code B = sign(B).
```

3.4 Hash code mapping with similarity sensitivity

Similarity sensitivity learning At the second stage, we aim to learn hash functions to project new query samples onto appropriate hash codes. Following the conventions within the cross-modal hashing field, we choose a Convolutional Neural Network (CNN) for images and a Multi-Layer Perception (MLP) for texts to perform the mapping. The outputs of these two networks can be denoted by $\boldsymbol{F}_{i*}^x = f^x(\boldsymbol{x}_i;\theta_x) \in \mathbb{R}^{1\times k}$ and $\boldsymbol{F}_{i*}^y = f^y(\boldsymbol{y}_i;\theta_y) \in \mathbb{R}^{1\times k}$, where θ_x and θ_y are the parameters of the image network and the text network, respectively. The loss function of the second stage can be divided into the mapping loss, classification loss and similarity preserving loss.

Considering that the above two networks (i.e., the CNN for images and the MLP for texts) should map the images and texts into a common representation space, we employ the following mapping loss to measure the error between these representations and the first-stage hash codes.

$$\mathcal{J}_1 = \left\|\boldsymbol{F}^{\boldsymbol{u}}-\boldsymbol{B}\right\|_F^2 \tag{18}$$

where u can be x or y, x for image modality and y for text modality.

Just as an Autoencoder is employed to ensure the preservation of semantic label information within hash codes during the first stage, we aspire to maintain the distinctiveness between data instances with different semantic labels in the common space. To predict the labels of the representations (i.e., the hash codes), two linear classifiers are attached to the top of the image network and the text network, respectively. The classification loss in the label space is defined as:

$$\mathcal{J}_2 = \left\| \boldsymbol{F^u V} - \boldsymbol{L} \right\|_F^2, \boldsymbol{V} \in \mathbb{R}^{k \times c} \tag{19}$$

where $\boldsymbol{V}$ is the parameter of the linear layer. This layer maps the hash code to class label for loss calculation.

The retrieval performance can be improved when the output of the network is close to the semantically similar data instances but far away from the dissimilar ones in the common representation space. Therefore, we introduce a similarity preserving loss. Firstly, we convert $\boldsymbol{S}_{ij}$ from the range of -1 to 1 back to the range of 0 to 1, which means that we convert -1 to 0 in $\boldsymbol{S}$. A coarse-grained matrix $\widetilde{\boldsymbol{S}}$ is then utilized to calculate its probability under the condition $\boldsymbol{F}^u$, which is formulated as:

$$\boldsymbol{p}(\widetilde{\boldsymbol{S}}_{ij} | \boldsymbol{F}^u_{i*}, \boldsymbol{F}^u_{j*}) = \begin{cases} \varphi(\psi_{ij}) & \widetilde{\boldsymbol{S}} = 1 \\ 1 - \varphi(\psi_{ij}) & \widetilde{\boldsymbol{S}} = 0 \end{cases} \tag{20}$$

where $\varphi(x) = \frac{1}{1+e^{-x}}$ and $\psi_{ij} = \frac{1}{2}(\boldsymbol{F}^u_{i*})(\boldsymbol{F}^u_{j*})^{\mathrm{T}}$; $\widetilde{\boldsymbol{S}}_{ij} = 1$ means that the data instances o_i and o_j form a similar pair, and $\widetilde{\boldsymbol{S}}_{ij} = 0$ means o_i and o_j form a dissimilar pair. The larger the inner product of two data instances is, the more similar they are. Thus, we formulate the negative log likelihood of the similarities as:

$$\widetilde{\mathcal{J}_3} = -\sum_{i,j=1}^{n} (\widetilde{\boldsymbol{S}}_{ij} \log(\varphi(\psi_{ij})) + (1 - \widetilde{\boldsymbol{S}}_{ij})(\log(1 - \varphi(\psi_{ij})))) \tag{21}$$

Motivated by Focal Loss (FL) [30], we assign different weights to different data pairs. For a similar pair, the more similar the pair is, the larger weight it will be assigned. Because the loss decreases as the Hamming distance decreases, we assign larger weights to significantly penalize the similar pairs with a small Hamming distance, i.e., hard similar pairs. Therefor, the loss is more sensitive to the pairs with higher degree of similarities, and then supervise the model to provide close representations for similar instances. To meet the above expectations, the weight is defined as:

$$\boldsymbol{W}_{ij} = (\gamma_1 \boldsymbol{S}_{ij})^{\widetilde{\boldsymbol{S}}_{ij}} \left(\frac{k + \mu_{ij}}{k} \gamma_2 \right)^{1 - \widetilde{\boldsymbol{S}}_{ij}} \tag{22}$$

where $\mu_{ij} = sign\left(\boldsymbol{F}^u_{i*}\right)(sign(\boldsymbol{F}^u_{j*}))^{\mathrm{T}}$; γ_1 and γ_2 are hyper-parameters that adjust the weights $\boldsymbol{W}$ and deal with the class imbalance problem. By taking the priority weight in (22), the similarity preserving loss is rewritten as a weighted cross-entropy loss:

$$\begin{aligned} \mathcal{J}_3 = -\sum_{i,j=1}^{n} & \boldsymbol{W}_{ij} (\widetilde{\boldsymbol{S}}_{ij} \log(\varphi(\psi_{ij})) \\ & + (1 - \widetilde{\boldsymbol{S}}_{ij})(\log(1 - \varphi(\psi_{ij})))) \end{aligned} \tag{23}$$

With this loss function, the mapping has the ability to preserve the differentiated similarity relationships between the original data. In addition, the weights used for the cross-entropy makes the training process sensitive to the hard similar pairs. The whole loss function of the second stage is therefore:

$$\mathcal{J} = \mathcal{J}_1 + \beta_1 \mathcal{J}_2 + \beta_2 \mathcal{J}_3 \tag{24}$$

where β_1 and β_2 are hyper-parameters to adjust the weights of $\mathcal{J}_2$ and $\mathcal{J}_3$. $\mathcal{J}_2$ is related to the length of hash code k and the number of classes c. It can be easily derived that the longer the hash code is, the easier the hash code can preserve the classification information. By contrast, the larger the number of classes is, the more difficult for the hash code to get the correct class. In conclusion, longer hash code and smaller number of the classes will lead to a smaller classification loss. Therefore, we put k in the numerator and c in the denominator. Finally, as the scale of this ratio is controlled by logarithm, β_1 is empirically set as $\ln\left(e + \left(\frac{k}{c}\right)^2\right) - 1$.

Optimization. At the second stage, we need to learn the parameters θ_x, θ_y and $\boldsymbol{V}$, which are updated by the Back-propagation (BP) algorithm with the Stochastic Gradient Descent (SGD) method. The derivatives of the loss function are computed as:

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \boldsymbol{F}^u_{i*}} = & 2\left(\boldsymbol{F}^u_{i*} - \boldsymbol{B}_{i*}\right) \\ & + \frac{1}{2}\beta_2 \boldsymbol{W}_{ij} \sum_{j:\widetilde{\boldsymbol{S}}_{ij} \in \widetilde{\boldsymbol{S}}} (\varphi(\psi_{ij}) - \widetilde{\boldsymbol{S}}_{ij}) \boldsymbol{F}^u_{j*} \\ & + \frac{1}{2}\beta_2 \boldsymbol{W}_{ji} \sum_{j:\widetilde{\boldsymbol{S}}_{ji} \in \widetilde{\boldsymbol{S}}} (\varphi(\psi_{ji}) - \widetilde{\boldsymbol{S}}_{ji}) \boldsymbol{F}^u_{j*} \end{aligned} \tag{25}$$

$$\frac{\partial \mathcal{J}}{\partial \boldsymbol{V}} = 2\beta_1 \left(\left(\boldsymbol{F}^u\right)^T \boldsymbol{F}^u \boldsymbol{V} - \left(\boldsymbol{F}^u\right)^T \boldsymbol{L} \right) \tag{26}$$

The gradients are then fed back to update the network parameters. The derivation of *sign* function is not involved in the process of back-propagation.

4 Experiments

4.1 Datasets

MIRFlickr-25K [20] consists of 25,000 images collected from Flickr. Each image is associated with a number of textual tags and labeled with at least one of the 24 unique semantic labels. We select the images with at least 20 textual tags, and obtain 20,015 image-tag pairs. For each pair, the textual instance is represented as a 1,386-dimensional Bag-of-words (BOW) vector. We randomly sample 2,000 pairs as the query set, and the remaining 18,015 pairs are used as the database where we randomly sample 10,000 pairs as the training set. It is worth mentioning that in this task, such spilt strategy is widely used and random selection does not have a significant impact on the results.

NUS-WIDE [11] is a real-world image dataset containing 269,648 images. Each image is associated with a number of textual tags labeled with at least one of the 81 unique concept labels. We only select the image-tag pairs which belong to the 21 most frequent concepts. For each pair, the textual instance is represented as a 1,000-dimensional BOW vector. 100 pairs per concept label are randomly sampled as the query set, and the remaining 500 pairs per concept label are randomly sampled as the training set. In total, there are 2,100 pairs for query set and 10,500 pairs for training set.

MS-COCO [29] contains 82,783 training images and 40,504 validation images, and is also a multi-label dataset. In our experiments, 117,218 data instances are used. Each text is represented as a 2,000-dimensional BOW vector. We randomly sample 5,000 instances as the query set, and the remaining are used as the database where we randomly sample 10,000 instances as the training set.

4.2 Evaluation metrics

In general, we experiment with two typical retrieval tasks for cross-modal hashing methods, i.e., Image-to-Text Retrieval ($\boldsymbol{I} \rightarrow \boldsymbol{T}$) and Text-to-Image Retrieval ($\boldsymbol{T} \rightarrow \boldsymbol{I}$). We adopt two commonly used protocols in cross-modal retrieval, i.e., Hamming ranking and hash lookup. Hamming ranking sorts the data instances in a database and returns the data instances with the highest similarity to a given query. The top-N-precision curve and Mean Average Precision (MAP) are widely used to measure the Hamming ranking task. Hash lookup returns all the data instances within a certain Hamming radius given the query instance. The precision-recall curve is widely used to measure the hash lookup task. To obtain the MAP score, we need to calculate the Average Precision (AP) for each query, which is defined as:

$$AP = \frac{1}{N} \sum_{r=1}^{R} p\,(r) \times rel_r \tag{27}$$

where N is the number of relevant data instances in the database; $p(r)$ represents the precision of the top-r returned data instances; $rel_r = 1$ if the r-th returned data instance is relevant to the query and $rel_r = 0$ otherwise. We can then calculate the MAP score by averaging the APs for all queries. Here, we choose MAP@R=500.

4.3 Implementation details

For image modality, following the method GCH [50], we adopt CNN-F [9] for image modality due to its excellent capability on image feature extraction. It is pre-trained on ImageNet [13]. We keep the five convolutional layers *conv1-conv5* and the next two fully-connected layers *fc6-fc7* unchanged and then replace the eighth layer *fc8* with a new fully-connected layer which has k nodes to map the deep image features into the Hamming space.

For text modality, we first use the BOW representation to transform each text into a vector, and then use a simple MLP to map the BOW vectors into the Hamming space. The MLP consists of three fully-connected layers with 512, 512 and k nodes, respectively.

For our proposed method, we set the hyper-parameters as $\alpha = \lambda = 1$, $\rho_1 = \rho_2 = 0.01$, $\beta_2 = 10$, $\gamma_1 = 1$, $\gamma_2 = 0.05$ for MIRFlickr-25K, $\gamma_1 = 0.5$, $\gamma_2 = 0.1$ for NUS-WIDE, and $\gamma_1 = 0.1$, $\gamma_2 = 0.5$ for MS-COCO. To learn the neural network parameters at the second stage, we adopt Adam solver and set the learning rate within 10^{-3} and 10^{-4}, and the batch size to 64. We implement our method with PyTorch on a single NVIDIA GTX 1080Ti GPU.

4.4 Comparison with state-of-the-art approaches

We compare our proposed method with the state-of-the-art cross-modal hashing methods, including CMSSH [4], SCM [53], STMH [46], SePH [31], DCMH [21], SSAH [25], GCH [50], DLFH [22], MLCAH [36], ASCSH [37], FCMH [49], MIAN [54] and MS^{2}GAH [15]. Among these methods, CMSSH, SCM, STMH, SePH and DLFH are shallow methods, and the rest are deep methods. The deep networks for our proposed DFGH are the same as the previous work. They all use the CNN-F network for image modality and MLP for text modality. Besides, SePH, DLFH and FCMH are also two-stage methods.

Table 2 reports the MAP scores of all ten methods and our DFGH for two cross-modal retrieval tasks with the hash codes varying from 16 bits to 64 bits. For fair comparison, we

Table 2 MAP scores of compared methods with different lengths of hash codes on three benchmark datasets

Task	Method	Year	MIRFLICKR-25K			NUS-WIDE			MS-COCO		
			16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
I→T	**CMSSH** [4]	2010	0.568	0.595	0.529	0.437	0.445	0.423	0.538	0.479	0.450
	SCM [53]	2014	0.556	0.559	0.557	0.436	0.432	0.429	0.385	0.387	0.384
	STMH [46]	2015	0.588	0.618	0.650	0.482	0.500	0.500	0.469	0.556	0.565
	SePH [31]	2015	0.752	0.772	0.784	0.653	0.659	0.688	0.561	0.601	0.648
	DCMH [21]	2017	0.724	0.731	0.731	0.568	0.561	0.596	0.505	0.536	0.557
	SSAH [25]	2018	0.782	0.790	0.800	0.642	0.636	0.639	0.550	0.558	0.557
	GCH [50]	2019	0.833	0.857	0.869	0.693	0.719	0.753	0.648	0.686	0.708
	DLFH [22]	2019	0.765	0.781	0.789	0.671	0.690	0.697	-	-	-
	MLCAH [36]	2020	0.796	0.808	0.815	0.644	0.641	0.643	0.570	0.562	0.562
	ASCSH [37]	2021	0.781	0.800	0.806	0.683	0.707	0.711	-	-	-
	FCMH [49]	2021	0.861	0.876	0.884	0.681	0.737	0.799	0.621	0.608	0.616
	MIAN [54]	2022	0.747	0.752	0.756	0.666	0.676	0.682	-	-	-
	MS^{2}GAH [15]	2022	0.851	0.871	0.879	0.704	0.727	0.765	0.662	0.701	0.732
	Our DFGH	2023	**0.876**	**0.887**	**0.893**	**0.762**	**0.779**	**0.806**	**0.722**	**0.777**	**0.808**
T→I	**CMSSH** [4]	2010	0.568	0.595	0.529	0.437	0.445	0.423	0.538	0.479	0.450
	SCM [53]	2014	0.556	0.559	0.557	0.436	0.432	0.429	0.385	0.387	0.384
	STMH [46]	2015	0.613	0.623	0.654	0.435	0.472	0.460	0.524	0.554	0.578
	SePH [31]	2015	0.689	0.697	0.709	0.578	0.575	0.568	0.572	0.620	0.650
	DCMH [21]	2017	0.764	0.749	0.780	0.558	0.591	0.616	0.549	0.572	0.605
	SSAH [25]	2018	0.791	0.795	0.803	0.669	0.662	0.666	0.537	0.538	0.529
	GCH [50]	2019	**0.892**	**0.910**	**0.907**	0.732	0.766	0.761	0.745	0.797	0.830
	DLFH [22]	2019	0.825	0.851	0.861	0.787	0.825	0.826	-	-	-
	MLCAH [36]	2020	0.794	0.805	0.805	0.662	0.673	0.687	0.544	0.547	0.594
	ASCSH [37]	2021	0.841	0.863	0.873	0.794	**0.828**	**0.843**	-	-	-
	FCMH [49]	2021	0.829	0.842	0.849	0.777	0.802	0.807	-	-	-
	MIAN [54]	2022	0.816	0.829	0.826	0.669	0.720	0.762	0.696	0.758	0.808
	MS^{2}GAH [15]	2022	0.883	0.908	0.901	0.757	0.789	0.783	0.751	0.822	0.868
	Our DFGH	2023	0.881	0.882	0.884	**0.799**	**0.828**	0.834	**0.813**	**0.893**	**0.940**

Table 2 continued

Task	Method	Year	MIRFLICKR-25K			NUS-WIDE			MS-COCO		
			16 bits	32 bits	64 bits	16 bits	32 bits	64 bits	16 bits	32 bits	64 bits
SUM	**CMSSH** [4]	2010	1.136	1.190	1.058	0.874	0.890	0.846	1.076	0.958	0.900
	SCM [53]	2014	1.112	1.118	1.114	0.872	0.864	0.858	0.770	0.774	0.768
	STMH [46]	2015	1.201	1.241	1.304	0.917	0.972	0.960	0.993	1.110	1.143
	SePH [31]	2015	1.441	1.469	1.493	1.231	1.220	1.256	1.133	1.221	1.298
	DCMH [21]	2017	1.488	1.480	1.511	1.126	1.152	1.212	1.054	1.108	1.162
	SSAH [25]	2018	1.573	1.585	1.603	1.311	1.298	1.305	1.087	1.096	1.086
	GCH [50]	2019	1.725	1.767	1.776	1.425	1.495	1.514	1.393	1.483	1.538
	DLFH [22]	2019	1.590	1.632	1.650	1.458	1.515	1.523	-	-	-
	MLCAH [36]	2020	1.590	1.613	1.620	1.306	1.314	1.330	1.114	1.109	1.156
	ASCSH [37]	2021	1.622	1.663	1.679	1.477	1.535	1.554	-	-	-
	FCMH [49]	2021	1.576	1.594	1.605	1.442	1.478	1.489	-	-	-
	MIAN [54]	2022	1.678	1.705	1.710	1.351	1.457	1.560	1.318	1.366	1.423
	MS^{2}GAH [15]	2022	1.734	**1.779**	**1.780**	1.461	1.516	1.548	1.413	1.523	1.600
	Our DFGH	2023	**1.757**	1.769	1.777	**1.561**	**1.607**	**1.640**	**1.535**	**1.670**	**1.748**

follow the setting of GCH in [50], as it reported the highest metric scores over all the comparison methods. In [50], the author reported the scores of CMSSH, SCM, STMH, SePH and DCMH. The comparisons between our method and these SOTA methods are under the same settings. We also discuss other methods, e.g., DLFH, MLCAH, ASCSH, FCMH, MIAN and MS^{2}GAH. To the best of our knowledge, these methods did not release the source code to the public, making it hard to reproduce the results under the same settings. Fortunately, according to existing works [25, 36, 50], the random sampling strategy makes little influence on the results. Under a similar scaling partition of the training and test set, the results of the same methods are almost the same. Therefore, we cite the results directly from the original papers. Furthermore, we use "SUM" to represent the sum of the MAP scores for the tasks of Image-to-Text Retrieval and Text-to-Image Retrieval. Figures 3, 4 and 5 show the top-N-precision and precision-recall curves with 64-bit hash codes on the three datasets. Because SSAH, MLCAH and MS^{2}GAH have not released their source codes and the original papers did not give the experimental results under the same settings as those listed in this work, there are no corresponding curves for these two methods in Figs. 3–5. Similarly, there are no corresponding curves for DLFH and ASCSH in Fig. 5.

As shown in Table 2 and Fig. 3–5, our DFGH outperforms most of the compared methods in terms of different datasets and different lengths of hash codes. Specifically, on MIRFlickr-25K, compared to GCH and MS^{2}GAH, which get the top three highest scores in "SUM", DFGH achieves a slight lower performance on the MAP score with 32-bit and 64-bit hash codes, while a significant improvement with 16-bit hash codes. It means that the reduction of the hash code length may have less impact on DFGH, because DFGH uses a fine-grained similarity matrix and a similarity preserving loss function to capture the differentiated similarity relationships, which is also effective for short hash codes. On both NUS-WIDE and MS-COCO, DFGH significantly outperforms the other methods. Compared with MIRFlickr-25K, NUS-WIDE and MS-COCO are more complex and contain larger amounts of data, and the tasks are thus more challenging. The excellent performance on both datasets demonstrates that DFGH performs well in complex scenarios where it is more demanding to capture the differentiated similarity relationships between the

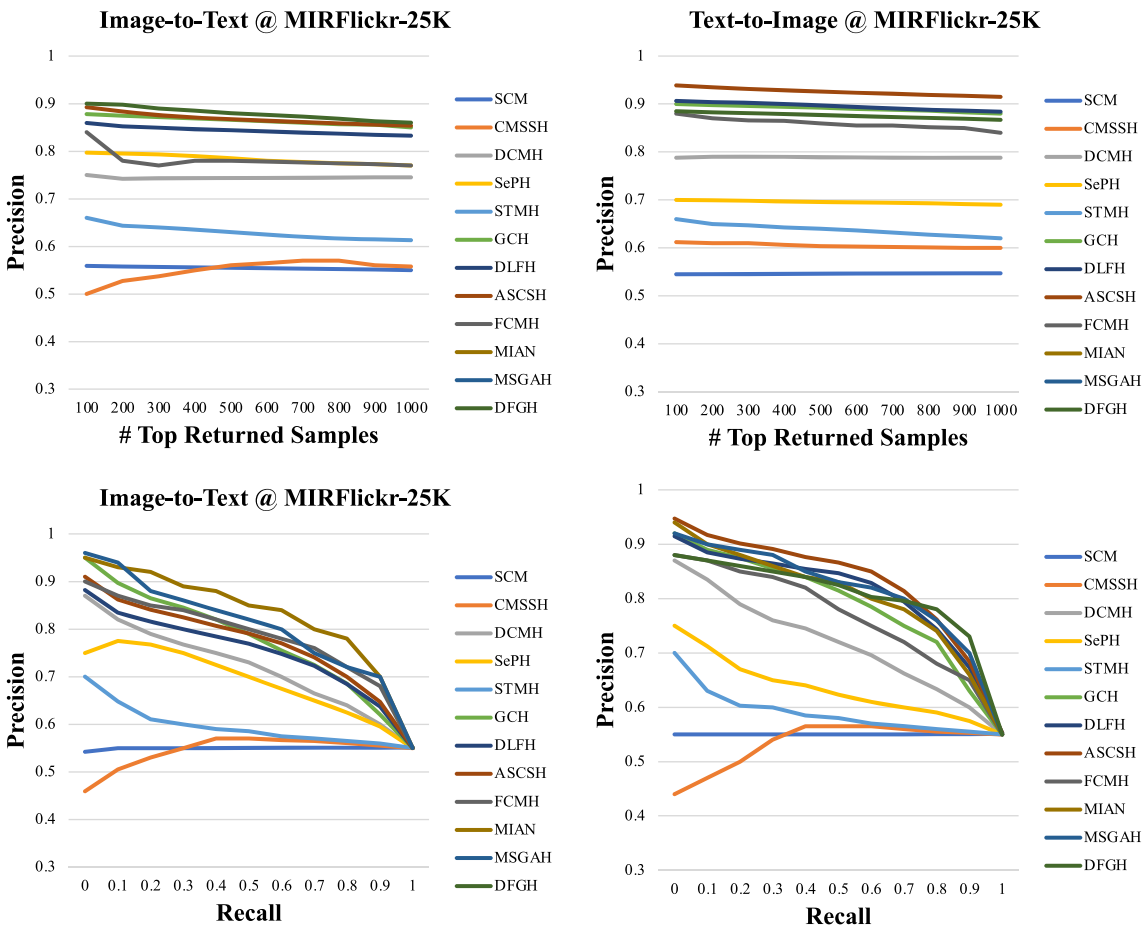


Fig. 3 The top-N-precision and precision-recall curves with 64-bit hash codes on MIRFlickr-25K

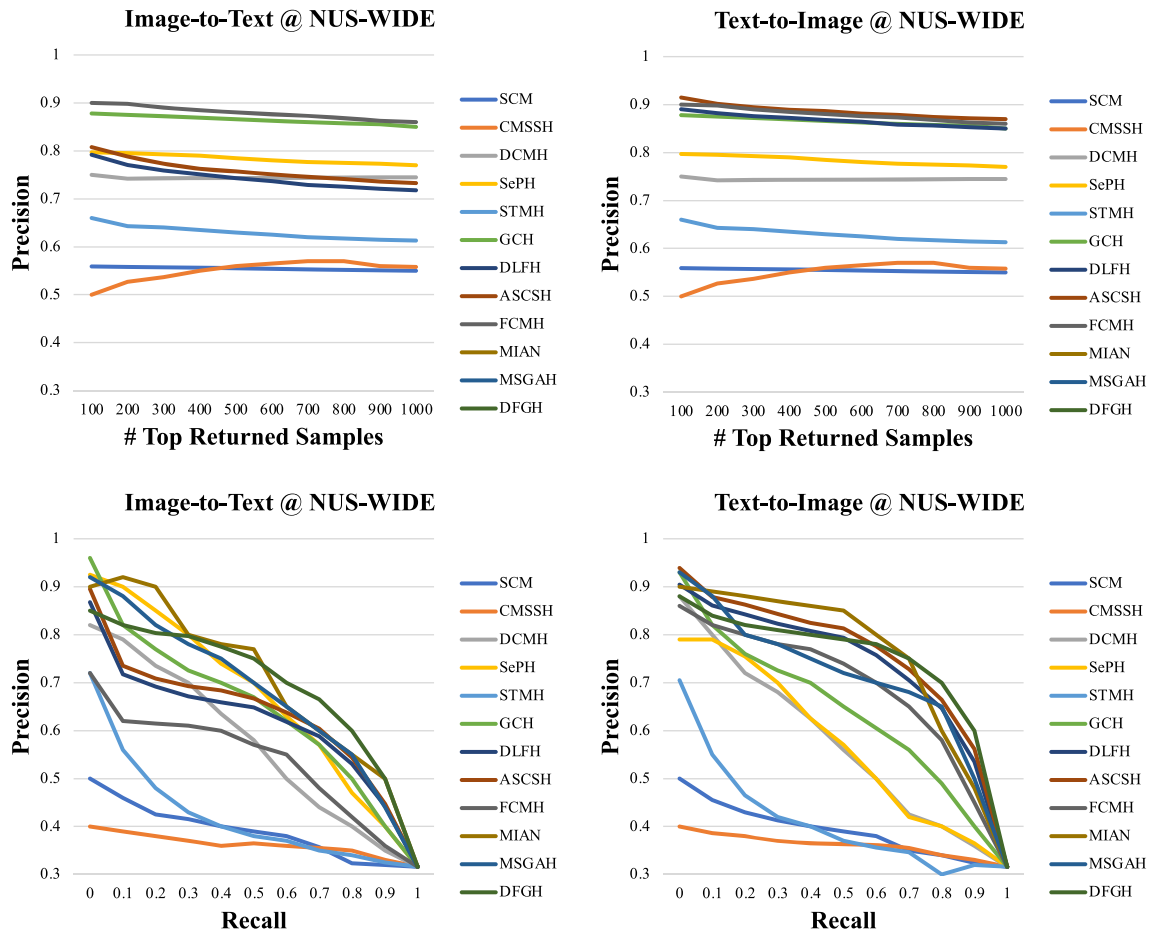


Fig. 4 The top-N-precision and precision-recall curves with 64-bit hash codes on NUS-WIDE

original data instances and preserve them in the Hamming space.

On these three benchmark datasets, for one-stage deep methods DCMH and GCH, the results of Text-to-Image Retrieval are generally better than those of Image-to-Text Retrieval, indicating that it is more difficult to capture the semantic similarity relationships between image modality and text modality via image query. For one-stage methods, the learning of hash codes is affected by both image features and text features. The generated hash codes may contain more semantic information of text modality, because it is more difficult to capture the hidden semantic information from an image than from texts. Different from the one-stage methods, the results for Image-to-Text Retrieval by our DFGH are generally slightly better than those for Text-to-Image Retrieval on MIRFlickr-25K. The reason may be that the two-stage methods only generate hash codes from the semantic labels without using the image features and text features at the first stage. Moreover, the training set is almost half the size of the entire dataset on MIRFlickr-25K, which makes the image and text networks have similar ability to map the original data to hash codes. However, on NUS-WIDE, the training set is only a small portion (less than 4%) of the entire dataset, making it more difficult to preserve the similarity relationships between different modalities for image modality. In the results of another recent two-stage method (i.e., FCMH), we can also observe a similar outcomes, while our approach further enhances performance. Additionally, with texts composed of sentences, MS-COCO has higher quality in text modality than the other two datasets. Therefore, it is much easier to capture the hidden semantic information from texts in MS-COCO. Different from SePH, our DFGH uses MLP for text modality, which has a stronger ability to extract text features.

The above experimental results have shown that the proposed DFGH outperforms the state-of-the-art cross-modal hashing methods. We now compare our DFGH with the two-stage method TECH [10] on MIRFlickr-25K. Because TECH used different settings from the other methods, we conduct another experiment by following the experimental protocols provided in TECH. Specifically, we randomly select 2,000 data instances as the query set and the remaining 18,015 instances as the training and retrieval set. Deep CNN-F features are extracted for TECH, and the MAP scores

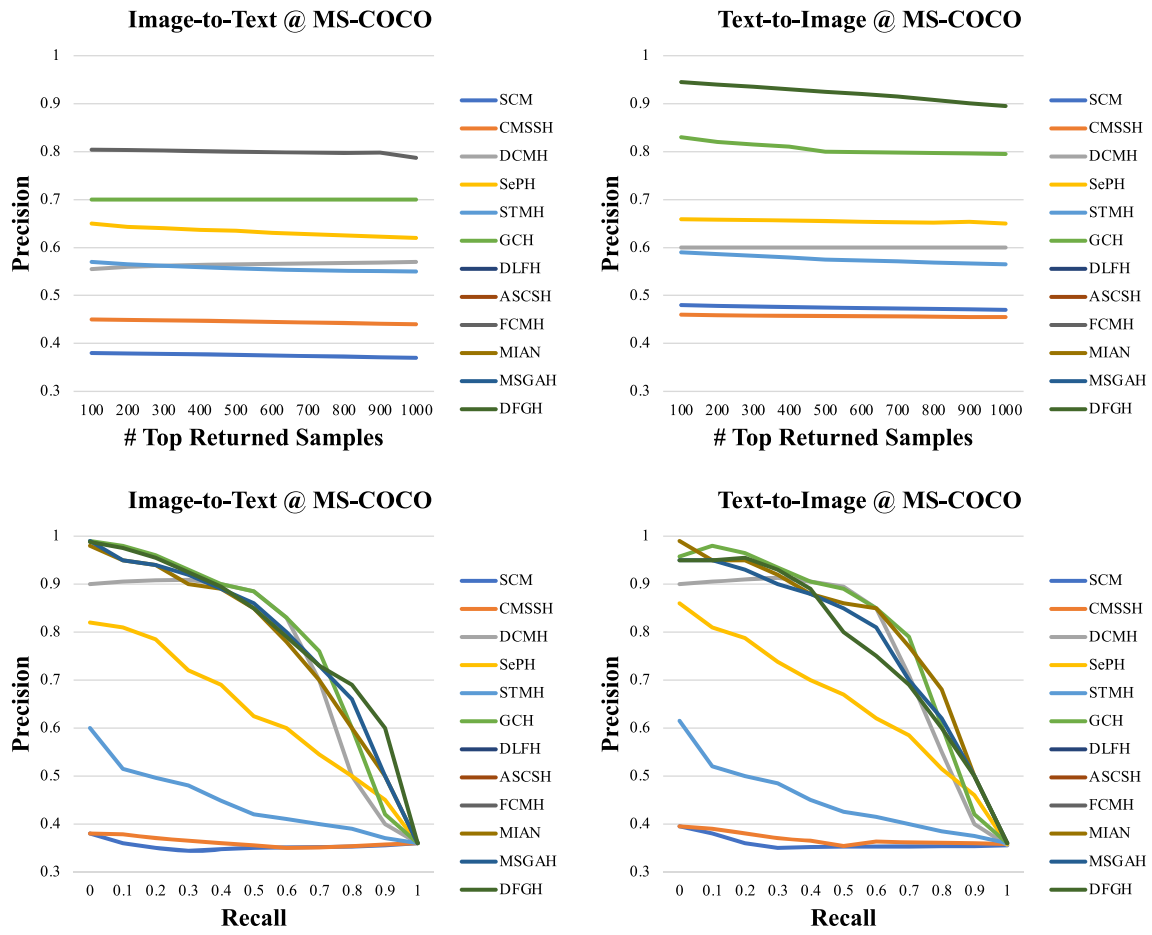


Fig. 5 The top-N-precision and precision-recall curves with 64-bit hash codes on MS-COCO

are adopted for all query instances. The related experimental results are reported in Table 3. It can be seen that DFGH outperforms TECH in most cases. From the MAP scores of DFGH, we can also find that the extracted text representations can preserve the similarity relationships between different modalities better than image representations. The greater performance boost in shorter code length proves the effectiveness of the preserved similarity relationships again. It should be noted that, in the task of text-to-image, our method outperforms TECH significantly. Our method surpasses the TECH by 0.040, 0.038, 0.026 respectively in 16 bits, 32 bits, and 64 bits. In fact, the task of text-to-image is far more widely used than the task of image-to-text in practice. For the latter task, the technique of image captioning is more common. In terms of computational complexity, the length of hash code is the determining factor. It is worthwhile to spend more time in the training process to obtain higher accuracy.

Table 3 MAP scores with different lengths of hash codes for TECH and DFGH on MIRFlickr-25K

Task	Method	MIRFLICKR-25K		
		16 bits	32 bits	64 bits
I→T	**TECH** [10]	0.806	0.819	**0.839**
	Our DFGH	**0.807**	**0.821**	0.833
T→I	**TECH** [10]	0.799	0.815	0.837
	Our DFGH	**0.839**	**0.853**	**0.863**
SUM	**TECH** [10]	1.605	1.634	1.676
	Our DFGH	**1.646**	**1.674**	**1.696**

4.5 Ablation study

Effects of main components To verify the effectiveness of the main components in our DFGH model, we conduct the ablation study experiments on three benchmark datasets, as shown in Table 4. We aim to explore four variants of DFGH: 1) DFGH-NAE is a DFGH variant without Autoencoder at the first stage; 2) DFGH-UWXE is a DFGH variant that uses unweighted cross-entropy loss at the second stage; 3) DFGH-MBC is a DFGH variant that uses multi-binary classification to train the hash functions at the second stage; and 4) DFGH-CGM is a DFGH variant that uses the coarse-

Table 4 MAP scores of DFGH and its variants on three benchmark datasets

Task	Method	MIRFLICKR-25K			NUS-WIDE			MS-COCO		
		16bits	32bits	64bits	16bits	32bits	64bits	16bits	32bits	64bits
I→T	**DFGH-NAE**	0.871	0.883	0.890	0.745	0.766	0.780	0.702	0.764	0.793
	DFGH-UWXE	0.867	0.879	0.885	0.744	0.770	0.785	0.709	0.764	0.793
	DFGH-MBC	0.861	0.875	0.886	0.743	0.769	0.780	0.703	0.757	0.796
	DFGH-CGM	0.855	0.869	0.881	0.737	0.764	0.770	0.693	0.751	0.791
	DFGH	**0.876**	**0.887**	**0.893**	**0.762**	**0.779**	**0.806**	**0.722**	**0.777**	**0.808**
T→I	**DFGH-NAE**	0.878	0.878	0.879	0.783	0.812	0.815	0.769	0.884	0.924
	DFGH-UWXE	0.873	0.874	0.881	0.786	0.811	0.816	0.783	0.869	0.924
	DFGH-MBC	0.872	0.873	0.875	0.780	0.808	0.818	0.785	0.873	0.928
	DFGH-CGM	0.860	0.868	0.875	0.784	0.799	0.813	0.755	0.862	0.904
	DFGH	**0.881**	**0.882**	**0.884**	**0.799**	**0.828**	**0.834**	**0.813**	**0.893**	**0.940**
SUM	**DFGH-NAE**	1.749	1.761	1.769	1.528	1.578	1.595	1.471	1.648	1.717
	DFGH-UWXE	1.740	1.753	1.766	1.530	1.581	1.601	1.492	1.633	1.717
	DFGH-MBC	1.733	1.749	1.761	1.523	1.577	1.598	1.488	1.630	1.724
	DFGH-CGM	1.715	1.737	1.756	1.521	1.563	1.583	1.448	1.613	1.695
	DFGH	**1.757**	**1.769**	**1.777**	**1.561**	**1.607**	**1.640**	**1.535**	**1.670**	**1.748**

DFGH-NAE: a DFGH variant without Autoencoder; DFGH-UWXE: a DFGH variant with an unweighted cross-entropy loss; DFGH-MBC: a DFGH variant using multi-binary classification; and DFGH-CGM: a DFGH variant that uses the coarse-grained similarity matrix

grained similarity matrix at the first stage and unweighted cross-entropy loss at the second stage.

As expected, DFGH outperforms DFGH-NAE, which verifies the effectiveness of the added Autoencoder at the first stage. Comparing DFGH with DFGH-UWXE, we find that the unweighted pairwise cross-entropy loss may lead to suboptimal performance. The main reason is that this loss assigns the same weight to each pair, causing the learning process to be equally sensitive to each pair. In contrast, our DFGH model uses the fine-grained similarity matrix to assign different weights to different pairs, making the training process sensitive to similar and hard pairs and further enabling the hash codes in Hamming space to preserve the differentiated similarity relationships between the original data. DFGH-UWXE significantly outperforms DFGH-CGM. The only difference between these two variants is the similarity matrix used at the first stage. This comparison shows the effectiveness of the fine-grained similarity matrix and further demonstrates the importance of fully capturing the similarity information.

Table 5 MAP scores of models without $\mathcal{J}_1$, $\mathcal{J}_2$ and $\mathcal{J}_3$ on three benchmark datasets

Task	Method	MIRFLICKR-25K			NUS-WIDE			MS-COCO		
		16bits	32bits	64bits	16bits	32bits	64bits	16bits	32bits	64bits
I → T	w/o $\mathcal{J}_1$	0.560	0.517	0.547	0.363	0.354	0.443	0.190	0.288	0.298
	w/o$\mathcal{J}_2$	0.856	0.868	0.872	0.743	0.768	0.780	0.705	0.765	0.802
	w/o$\mathcal{J}_3$	0.852	0.865	0.868	0.732	0.767	0.772	0.704	0.754	0.793
	All	**0.876**	**0.887**	**0.893**	**0.762**	**0.779**	**0.806**	**0.722**	**0.777**	**0.808**
T→I	w/o$\mathcal{J}_1$	0.545	0.433	0.491	0.268	0.313	0.374	0.188	0.301	0.247
	w/o$\mathcal{J}_2$	0.865	0.870	0.873	0.780	0.805	0.815	0.783	0.875	0.927
	w/o$\mathcal{J}_3$	0.855	0.864	0.869	0.786	0.804	0.818	0.776	0.877	0.924
	All	**0.881**	**0.882**	**0.884**	**0.799**	**0.828**	**0.834**	**0.813**	**0.893**	**0.940**
SUM	w/o$\mathcal{J}_1$	1.105	0.950	1.038	0.631	0.667	0.817	0.378	0.589	0.545
	w/o$\mathcal{J}_2$	1.721	1.738	1.745	1.523	1.573	1.595	1.488	1.640	1.729
	w/o$\mathcal{J}_3$	1.707	1.729	1.737	1.518	1.571	1.590	1.480	1.631	1.717
	All	**1.757**	**1.769**	**1.777**	**1.561**	**1.607**	**1.640**	**1.535**	**1.670**	**1.748**

"w/o" means without the corresponding loss. "All" means all the losses are used

Effects of loss functions in Stage 2 Further, to verify the effectiveness of the components of the loss function at the second stage, we conduct the ablation study experiments on three benchmark datasets, as shown in Table 5. We report the MAP scores of the model without $\mathcal{J}_1$, $\mathcal{J}_2$ and $\mathcal{J}_3$ on three benchmark datasets, respectively. On the whole, $\mathcal{J}_1$ is the most crucial loss. There is a dramatic drop in performance without $\mathcal{J}_1$. This proves the effectiveness of the pipeline of our two-stage method. The most important thing for hash function learning is to get guidance on the hash codes inferred from the first stage. For $\mathcal{J}_2$ and $\mathcal{J}_3$, there is an improvement of an average of 0.02 for each task. Note that the improvement brought by doubling the hash code length is about 0.02 here, so such improvement is significant. It indicates that the design of classification loss and weighted similarity preserving loss can well boost the retrieval performance, proving the effectiveness of preserving the semantic information of labels and similarity information in hash codes.

4.6 Parameter sensitivity analysis

We analyze parameter sensitivity on the most used datasets MIRFlickr-25K and NUS-WIDE. We design experiments with both retrieval tasks and fix the hash code length as 64 bits empirically. There are eight hyper-parameters in DFGH, i.e., α, λ, γ_1, ρ_1, ρ_2, γ_2, β_1 and β_2. As mentioned before, β_1 is related to the length of the hash code k and the number of classes c, thus we construct the value of β_1 empirically. In addition, we follow the previous work SADH in [41] and set the value of two dual variables ρ_1 and ρ_2 to 0.01. Therefore, we only present the experimental results about α, λ, and β_2 in Figs. 6, 7 and 8. Here, we use "Average" to represent the average score for Image-to-Text Retrieval and Text-to-Image Retrieval. Overall, the influence of the parameters on the results tends to be clearer and more stable. This means that in the parameter selection process, we can get the most suitable parameters without too much effort.

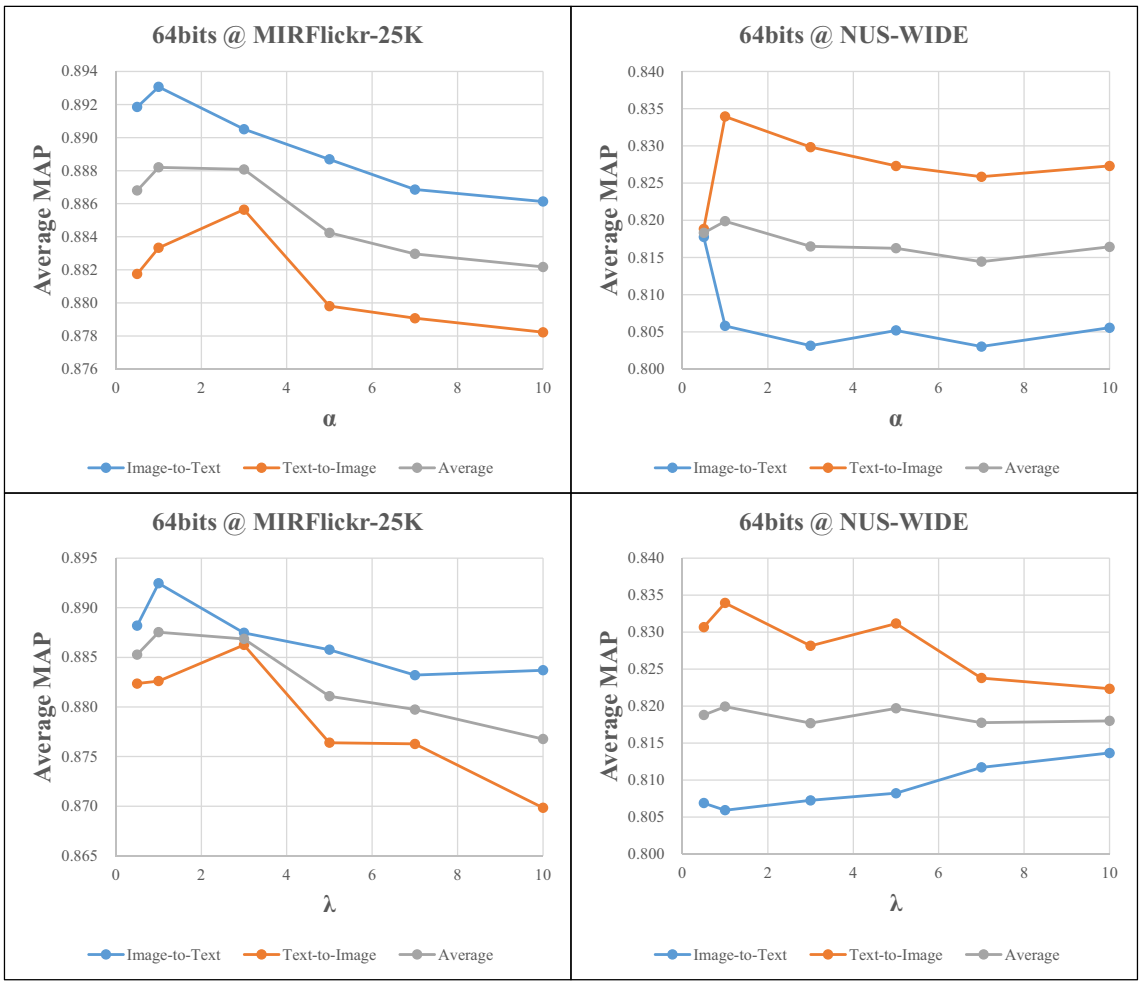


Fig. 6 Parameter analysis of α and λ on MIRFlickr-25K and NUS-WIDE

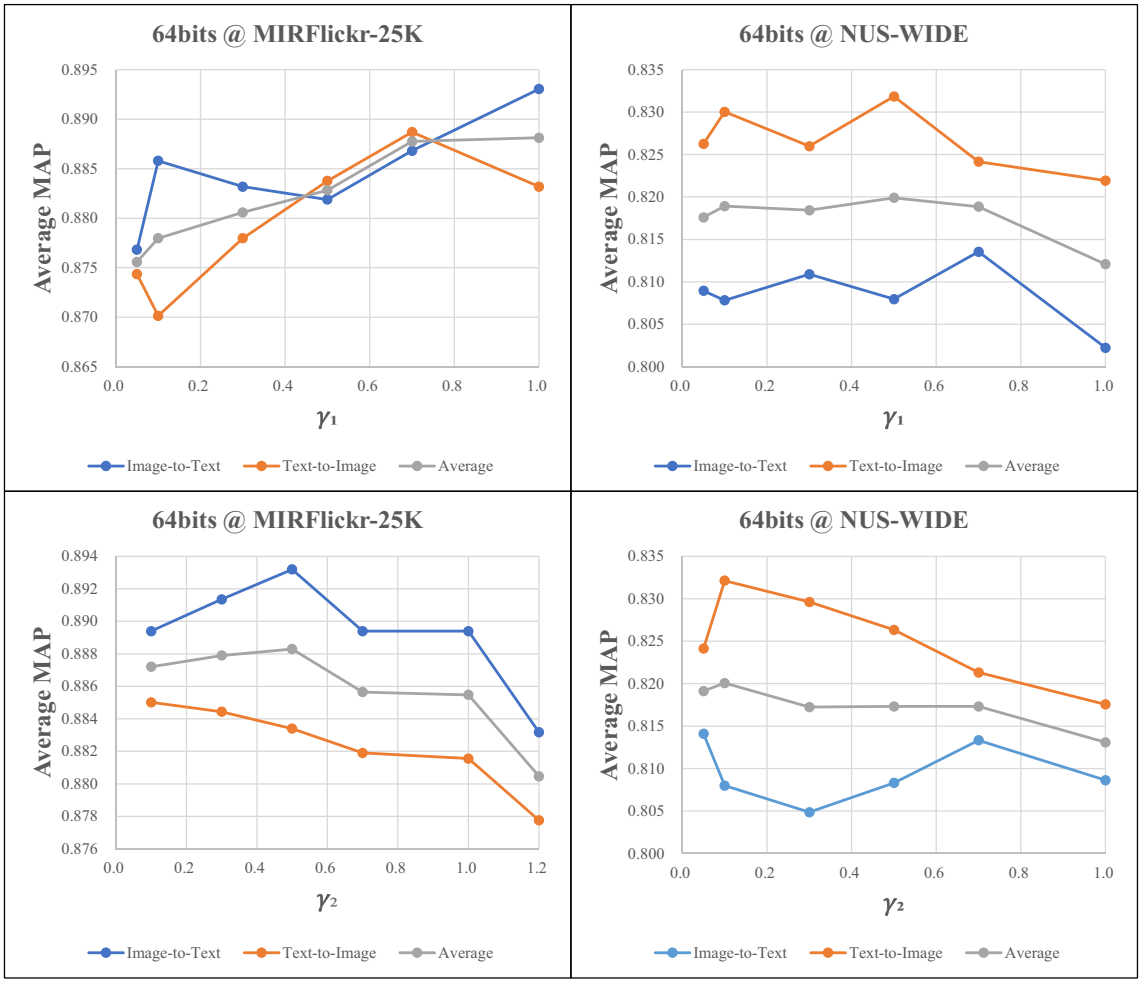


Fig. 7 Parameter analysis of γ_1 and γ_2 on MIRFlickr-25K and NUS-WIDE

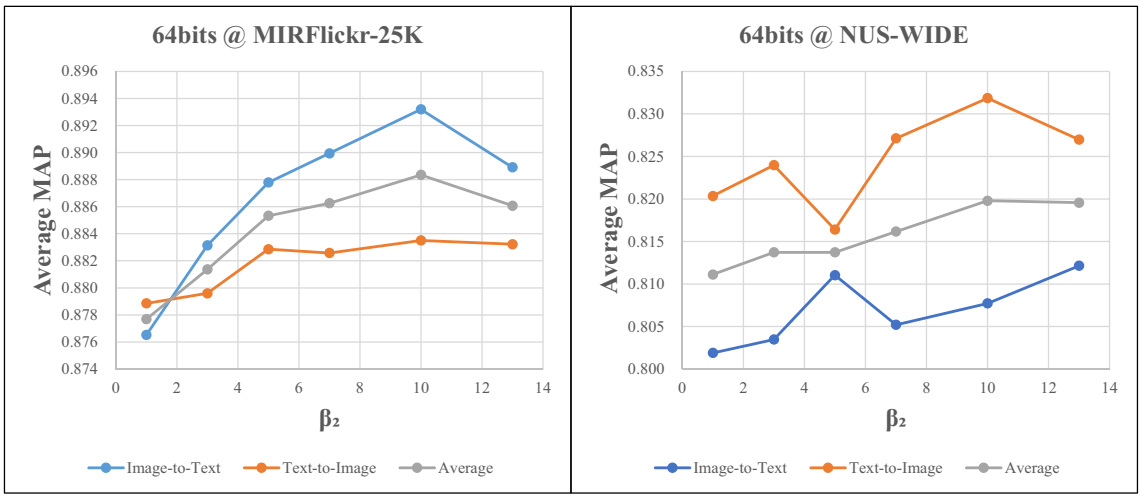


Fig. 8 Parameter analysis of β_2 on MIRFlickr-25K and NUS-WIDE

Parameters α and λ represent the importance of the Autoencoder loss. From Fig. 6, we observe that the average MAP scores of DFGH first increase and then drop with α and λ in $[0.5, 10]$ on MIRFlickr-25K, but keep relative stable on NUS-WIDE. It indicates that the semantic information of labels is helpful to improve the retrieval performance, but too much intervention of Autoencoder may make the learning of hash code worse on a small-scale dataset.

Parameters γ_1 and γ_2 adjust the coarse-grained matrix and deal with the class imbalance problem. From Fig. 7, we can observe that on MIRFlickr-25K, the average MAP scores first increase steadily with γ_1 in [0.05, 0.7], and then remain stable with γ_1 in [0.7, 1.0]. Besides, the average MAP scores first keep relatively stable with γ_2 in [0.7, 1.0], and then drop rapidly. Different from those on MIRFlickr-25K, the average MAP scores on NUS-WIDE are not sensitive to the hyper-parameters γ_1 and γ_2 in $[0.05, 0.7]$, and drop when γ_1 and γ_2 are greater than 0.7.

Parameter β_2 controls the similarity preserving loss. As shown in Fig. 8, on MIRFlickr-25K the average MAP scores first increase steadily with β_2 in [1, 10], indicating that the similarity preserving loss is necessary to learn the hash functions. However, when β_2 becomes too large, the average MAP scores also drop. Different from the average MAP scores on MIRFlickr-25K, those on NUS-WIDE first increase steadily with β_2 in [1, 10], and then keep stable over a wide range of values. Besides, the best retrieval performance can be achieved under different settings on different datasets.

5 Conclusion

In this paper, we introduce a two-stage deep hashing method (i.e., DFGH) for cross-modal retrieval. To generate better hash codes, we design a fine-grained similarity matrix to explore the differentiated similarity relationships between data instances. An autoencoder is utilized to preserve the semantic information of labels. To obtain better hash functions, we design a similarity sensitivity learning strategy and a similarity preserving loss. They enable the hash codes generated by the hash functions to preserve the differentiated similarity relationships in the Hamming space. The adoption of such fine-grained similarity matrix obviously promotes the retrieval performance, demonstrating the importance of exploring fine-grained similarity information. In the future, it is promising to utilize more information besides of labels to mine more similarity information for the construction of the similarity matrix.

Acknowledgements This work was supported by National Natural Science Foundation of China (No. 61976057; No. 62172101), and the Science and Technology Commission of Shanghai Municipality (No. 21511101000; No. 22DZ1100101), and the Fundamental Research Funds for the Central Universities of China (No. 2023110139).

Author Contributions Conceptualization and formal analysis: All authors; Methodology, software and investigation: Yangdong Chen, Jiaqi Quan; Writing - original draft preparation: Yangdong Chen, Jiaqi Quan; Writing - review and editing: Yuejie Zhang, Rui Feng, Tao Zhang; Funding acquisition: Yuejie Zhang, Rui Feng; Resources: Yuejie Zhang, Rui Feng, Tao Zhang; Supervision: Yuejie Zhang, Rui Feng, Tao Zhang; Project administration: Yuejie Zhang;

Funding This work was supported by National Natural Science Foundation of China (No. 61976057; No. 62172101), and the Science and Technology Commission of Shanghai Municipality (No. 21511101000; No.22DZ1100101).

Data availability and access The data of MIRFlickr that support the findings of this study are openly available at: https://press.liacs.nl/mirflickr/, reference number [20]. The data of NUS-WIDE that support the findings of this study are available at: https://lms.comp.nus.edu.sg/wp-content/uploads/2019/research/nuswide/NUS-WIDE.html, reference number [11]. The data of that support the findings of this study are available at: https://cocodataset.org/, reference number [29].

Declarations

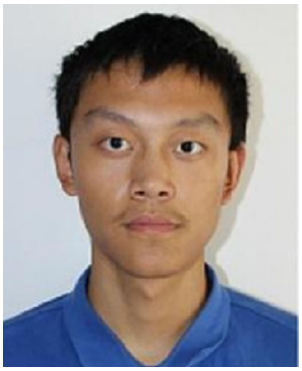
Conflicts of interest The authors have no competing interests to declare that are relevant to the content of this article.

Ethical standard The authors declare that they have no potential conflicts of interest with the content of this paper, and there is no research involving human participants or animals in this paper.

References

1. Ayadi H, Torjmen-Khemakhem M, Daoud M et al (2018) Mf-re-rank: A modality feature-based re-ranking model for medical image retrieval. J Assoc Inf Sci Technol 69(9):1095–1108
2. Bartels RH, Stewart GW (1972) Solution of the matrix equation ax+ xb= c [f4]. Commun ACM 15(9):820–826
3. Boyd S, Parikh N, Chu E (2011) Distributed optimization and statistical learning via the alternating direction method of multipliers. Now Publishers Inc
4. Bronstein MM, Bronstein AM, Michel F, et al (2010) Data fusion through cross-modality metric learning using similarity-sensitive hashing. In: CVPR, pp 3594–3601
5. Cao Y, Long M, Wang J, et al (2016) Deep visual-semantic hashing for cross-modal retrieval. In: KDD, pp 1445–1454
6. Cao Y, Liu B, Long M, et al (2018) Hashgan: Deep learning to hash with pair conditional wasserstein gan. In: CVPR, pp 1287–1296
7. Cao Z, Long M, Huang C, et al (2018) Transfer adversarial hashing for hamming space retrieval. In: AAAI
8. Cao Z, Sun Z, Long M, et al (2018) Deep priority hashing. In: MM, pp 1653–1661
9. Chatfield K, Simonyan K, Vedaldi A, et al (2014) Return of the devil in the details: Delving deep into convolutional nets. In: BMVC
10. Chen ZD, Wang Y, Li HQ, et al (2019) A two-step cross-modal hashing by exploiting label correlations and preserving similarity in both steps. In: MM, pp 1694–1702

11. Chua TS, Tang J, Hong R, et al (2009) Nus-wide: a real-world web image database from national university of singapore. In: CIVR, pp 1–9
12. Deng C, Chen Z, Liu X et al (2018) Triplet-based deep hashing network for cross-modal retrieval. IEEE Trans Image Process 27(8):3893–3903
13. Deng J, Dong W, Socher R, et al (2009) Imagenet: A large-scale hierarchical image database. In: CVPR, pp 248–255
14. Ding G, Guo Y, Zhou J (2014) Collective matrix factorization hashing for multimodal data. In: CVPR, pp 2075–2082
15. Duan Y, Chen N, Zhang P et al (2022) Ms2gah: Multi-label semantic supervised graph attention hashing for robust cross-modal retrieval. Pattern Recognit 128:108676
16. Erin Liong V, Lu J, Tan YP, et al (2017) Cross-modal deep variational hashing. In: ICCV, pp 4077–4085
17. Hendricks LA, Venugopalan S, Rohrbach M, et al (2016) Deep compositional captioning: Describing novel object categories without paired training data. In: CVPR, pp 1–10
18. Hu H, Xie L, Hong R, et al (2020) Creating something from nothing: Unsupervised knowledge distillation for cross-modal hashing. In: CVPR, pp 3123–3132
19. Hu Y, Jin Z, Ren H, et al (2014) Iterative multi-view hashing for cross media indexing. In: MM, pp 527–536
20. Huiskes MJ, Lew MS (2008) The mir flickr retrieval evaluation. In: MIR, pp 39–43
21. Jiang QY, Li WJ (2017) Deep cross-modal hashing. In: CVPR, pp 3232–3240
22. Jiang QY, Li WJ (2019) Discrete latent factor model for cross-modal hashing. IEEE Trans Image Process 28(7):3490–3501
23. Jin L, Li K, Li Z et al (2018) Deep semantic-preserving ordinal hashing for cross-modal similarity search. TNNLS 30(5):1429–1440
24. Kumar S, Udupa R (2011) Learning hash functions for cross-view similarity search. In: IJCAI, pp 1360–1365
25. Li C, Deng C, Li N, et al (2018) Self-supervised adversarial hashing networks for cross-modal retrieval. In: CVPR, pp 4242–4251
26. Li H, Zhang C, Jia X, et al (2021) Adaptive label correlation based asymmetric discrete hashing for cross-modal retrieval. IEEE Transactions on Knowledge and Data Engineering
27. Li W, Zheng Y, Zhang Y et al (2021) Cross-modal retrieval with dual multi-angle self-attention. J Assoc Inf Sci Technol 72(1):46–65
28. Lin G, Shen C, Suter D, et al (2013) A general two-step approach to learning-based hashing. In: ICCV, pp 2552–2559
29. Lin TY, Maire M, Belongie S, et al (2014) Microsoft coco: Common objects in context. In: ECCV, pp 740–755
30. Lin TY, Goyal P, Girshick R, et al (2017) Focal loss for dense object detection. In: ICCV, pp 2980–2988
31. Lin Z, Ding G, Hu M, et al (2015) Semantics-preserving hashing for cross-view retrieval. In: CVPR, pp 3864–3872
32. Liu H, Ji R, Wu Y, et al (2017) Cross-modality binary code learning via fusion similarity hashing. In: CVPR, pp 7380–7388
33. Liu X, Wang X, Ym Cheung (2021) Fddh: Fast discriminative discrete hashing for large-scale cross-modal retrieval. IEEE Trans Neural Netw Learn Syst 33(11):6306–6320
34. Luo X, Yin XY, Nie L, et al (2018) Sdmch: Supervised discrete manifold-embedded cross-modal hashing. In: IJCAI, pp 2518–2524
35. Luo X, Zhang PF, Huang Z et al (2019) Discrete hashing with multiple supervision. IEEE Trans Image Process 28(6):2962–2975
36. Ma X, Zhang T, Xu C (2020) Multi-level correlation adversarial hashing for cross-modal retrieval. IEEE Trans Multimed 22(12):3101–3114
37. Meng M, Wang H, Yu J et al (2020) Asymmetric supervised consistent and specific hashing for cross-modal retrieval. IEEE Trans Image Process 30:986–1000
38. Nie X, Wang B, Li J, et al (2020) Deep multiscale fusion hashing for cross-modal retrieval. TCSVT
39. Noh H, Seo PH, Han B (2016) Image question answering using convolutional nal network with dynamic parameter prediction. In: CVPR, pp 30–38
40. Rastegari M, Choi J, Fakhraei S, et al (2013) Predictable dual-view hashing. In: ICML, pp 1328–1336
41. Shen F, Xu Y, Liu L et al (2018) Unsupervised deep hashing with similarity-adaptive and discrete optimization. IEEE Trans Pattern Anal Mach Intell 40(12):3034–3044
42. Shen Y, Liu L, Shao L, et al (2017) Deep binaries: Encoding semantic-rich cues for efficient textual-visual cross retrieval. In: ICCV, pp 4097–4106
43. Song J, Yang Y, Yang Y, et al (2013) Inter-media hashing for large-scale retrieval from heterogeneous data sources. In: SIGMOD, pp 785–796
44. Tu RC, Mao XL, Ma B, et al (2020) Deep cross-modal hashing with hashing functions and unified hash codes jointly learning. IEEE Transactions on Knowledge and Data Engineering
45. Wang D, Cui P, Ou M, et al (2015) Deep multimodal hashing with orthogonal regularization. In: IJCAI
46. Wang D, Gao X, Wang X, et al (2015) Semantic topic multimodal hashing for cross-media retrieval. In: IJCAI
47. Wang W, Ooi BC, Yang X, et al (2014) Effective multi-modal retrieval based on stacked auto-encoders. In: VLDB, pp 649–660
48. Wang Y, Luo X, Nie L, et al (2020) Batch: A scalable asymmetric discrete cross-modal hashing. IEEE Transactions on Knowledge and Data Engineering
49. Wang Y, Chen ZD, Luo X et al (2021) Fast cross-modal hashing with global and local similarity embedding. IEEE Trans Cybern 52(10):10064–10077
50. Xu R, Li C, Yan J, et al (2019) Graph convolutional network hashing for cross-modal retrieval. In: IJCAI, pp 982–988
51. Yan C, Pang G, Bai X, et al (2019) Deep hashing by discriminating hard examples. In: MM, pp 1535–1542
52. Yang E, Deng C, Liu W, et al (2017) Pairwise relationship guided deep hashing for cross-modal retrieval. In: AAAI
53. Zhang D, Li WJ (2014) Large-scale supervised multimodal hashing with semantic correlation maximization. In: AAAI
54. Zhang Z, Luo H, Zhu L et al (2022) Modality-invariant asymmetric networks for cross-modal hashing. IEEE Trans Knowl Data Eng 35(5):5091–5104
55. Zhen Y, Yeung DY (2012) Co-regularized hashing for multimodal data. In: NIPS, p 1376
56. Zheng C, Zhu L, Lu X et al (2019) Fast discrete collaborative multimodal hashing for large-scale multimedia retrieval. IEEE Trans Knowl Data Eng 32(11):2171–2184
57. Zhou J, Ding G, Guo Y (2014) Latent semantic sparse hashing for cross-modal similarity search. In: SIGIR, pp 415–424
58. Zhu X, Huang Z, Shen HT, et al (2013) Linear cross-modal hashing for efficient multimedia search. In: MM, pp 143–152



Yangdong Chen received the B.S. degree in Computer Science from Fudan University, Shanghai, China, in 2019. He is currently a Ph.D student in School of Computer Science, Fudan University, Shanghai, China. He is a member of Institution of Media Computing in School of Computer Science. His research interest is multimedia/cross-media information analysis, including sketch-based image retrieval, sketch synthesis and cross modal retrieval.

Jiaqi Quan received the B.E. degree in Electronic and Information Engineering from Sichuan University, Sichuan, China, in 2017 and the M.E. degree in Computer Science from Fudan University, Shanghai, China, in 2021. His research interest is multimedia/cross-media information analysis, including cross modal retrieval, face recognition and face quality assessment.

Yuejie Zhang received the B.S. degree in Computer Software, the M.S. degree in Computer Application, and the Ph.D. degree in Computer Software and Theory from Northeastern University, Shenyang, China, in 1994, 1997 and 1999, respectively. She was a Postdoctoral Researcher at Fudan University, Shanghai, China, from 1999 to 2001. In 2001, she joined School of Computer Science, Fudan University as an Assistant Professor, and then become Associate Professor and Full Professor. Her research interests include multimedia/cross-media information analysis, processing, and retrieval, and machine learning.

Rui Feng received the B.S. degree in Industrial Automatic from Harbin Engineering University, Haerbin, China, in 1994, the M.S. degree in Industrial Automatic from Northeastern University, Shenyang, China, in 1997, and the Ph.D. degree in Control Theory and Engineering from Shanghai Jiaotong University, Shanghai, China, in 2003. In 2003, He joined Department of Computer Science and Engineering (now School of Computer Science), Fudan University as an Assistant Professor, and then become Associate Professor and Full Professor. His research interests include multimedia information analysis and processing, and machine learning.

Tao Zhang received the B.S. and M.S. degree in Automation Control, and the Ph.D. degree in System Engineering from Northeastern University, Shenyang, China, in 1992, 1997 and 2000, respectively. He was a Postdoctoral Researcher at Fudan University, Shanghai, China, from 2001 to 2003. In 2003, he joined School of Information Management and Engineering, Shanghai University of Finance and Economics as an Associate Professor and then become Full Professor. His research interests include big data analysis, system modeling and optimization.