Information Processing and Management 59 (2022) 102886

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/ipm

Cross-modal image–text search via Efficient Discrete Class Alignment Hashing

Song Wang [a,b], Huan Zhao [a,b,*], Yunbo Wang [c], Jing Huang [a,b], Keqin Li [a,b,d]

[a] College of Computer Science and Electronic Engineering of Hunan University, Changsha 410082, China
[b] Key Laboratory for Embedded and Network Computing of Hunan Province, Changsha 410082, China
[c] Wangxuan Institute of Computer Technology, Peking University, Beijing, 100080, China
[d] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Keywords:
Class alignment
Cross-modal image–text search
Hash code
Supervised hashing

ABSTRACT

Hashing has produced enormous potentials in cross-modal image–text search, which learns compact binary codes by exploring the correlations between distinct modalities. However, there still exist some limitations. First, most existing methods neglect the relation between the data characteristics and supervised information. Second, a relaxation strategy results in large quantization errors. Third, constructing large $n \times n$ (a.k.a. training size) similarity graphs increases computational load. To address these issues, we propose a novel discrete supervised hashing method, termed Efficient Discrete Class Alignment Hashing (EDCAH), which integrates class alignment and matrix factorization for hashing learning. Specifically, it exploits the semantic consistency of data instances and informative labels to simultaneously learn the hash codes and hash functions. Meanwhile, a discrete optimization strategy is developed to solve the EDCAH, which is beneficial to generate high-quality hash codes. Furthermore, to improve the learning efficiency of EDCAH, we propose a fast and efficient variant dubbed EDCAH-t that utilizes a two-step hashing strategy. Extensive experiments demonstrate the superiority of EDCAH and EDCAH-t in both search accuracy and learning efficiency.

1. Introduction

1.1. Background and motivation

Cross-modal image–text search, a fundamental but popular research topic (Lu, Zhu, Cheng, Song, & Zhang, 2019; Wang, Ou, Liang, & Sun, 2021; Zhang, Li, Jiang, Yuan, & Zhang, 2018) that aims to realize accurate retrieval across different data modalities, has attracted considerable attention in machine learning (Li, 2017; Wang, Shen, Zhang, & Liu, 2020), information retrieval (Han, Zhang, Ren, & Schuller, 2019; He & Zhao, 2017; Zhao, Cao, Xu, & Lu, 2020), and multimedia modeling (Liang, He, Sun, & Tan, 2019; Zeng, Zhang, & Zhu, 2020). Due to the tremendous growth of multimedia data (e.g., images, texts, and videos) in social networks, performing accurate and fast searches has become an intractable challenge on the search engines due to its limited search ability and storage resource (Peng, Huang, & Zhao, 2018; Zhang & Peng, 2018). Thus, Cross-modal Image–Text Hashing (CMITH) (Li, Yang, Wang, Song, & Li, 2021; Wang, Zhang, Song, Sebe, & Shen, 2018; Yao et al., 2020; Zhao, Wang, She, & Su, 2020), has been proposed to transform multimedia data into compact hash codes while preserving the semantic similarity of the data instances. Such

* Corresponding author at: College of Computer Science and Electronic Engineering of Hunan University, Changsha 410082, China.
E-mail addresses: swang17@hnu.edu.cn (S. Wang), hzhao@hnu.edu.cn (H. Zhao), wangyunbo09@gmail.com (Y. Wang), jingh@hnu.edu.cn (J. Huang), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.ipm.2022.102886
Received 28 June 2021; Received in revised form 4 January 2022; Accepted 22 January 2022
0306-4573/© 2022 Elsevier Ltd. All rights reserved.

a method greatly promotes learning efficiency and reduces storage costs based on the generated hash codes (Ding, Wong, Lai, & Zhang, 2020; Huang et al., 2020). Due to the efficiency in both search speed and storage cost, CMITH has been widely applied in various large-scale cross-modal image–text search scenarios.

Existing CMITH methods can be classified into two categories according to the dependence on supervised information: unsupervised hashing (Ding, Guo, & Zhou, 2014; Gui, Liu, Sun, Tao, & Tan, 2017; Li et al., 2017; Shen et al., 2018; Wang, Sun, Zhao, & Su, 2017; Zhang & Peng, 2020; Zhou, Ding, & Guo, 2014) and supervised hashing (Chen et al., 2020; Chen, Shen, Yang, Xu, & Song, 2017; Jiang & Li, 2019; Li, Yan, Luo, Nie, & Xu, 2019; Lin, Ding, Han, & Wang, 2016; Liu, Ji, Wu, & Hua, 2016; Tang, Wang, & Shao, 2016; Wang, Gao, Wang, & He, 2019). The former achieves the search tasks by exploring the data distribution and characteristics of the original instances to construct the hash functions, which encodes the instances into the hash codes. The latter supervised hashing, performs the search tasks by employing the supervision information (i.e., semantic labels or pairwise data constraints) to obtain the hash functions. Generally speaking, the supervised methods obtain better performance than the unsupervised ones in the tasks of cross-modal image–text search.

A variety of supervised hashing methods (Shen et al., 2016; Wang et al., 2019; Wang, Liang, Cao, & Sun, 2019; Ye & Peng, 2018; Zheng et al., 2020) has been designed for cross-modal image–text search. These methods learn unified hash codes or modality-specific hash functions by taking advantage of the similarity graphs or matrix factorization techniques. For example, several typical works include Supervised Matrix Factorization Hashing (SMFH) (Liu et al., 2016), Semantic Correlation Maximization (SCM) (Zhang & Li, 2014), Semantics-preserving Hashing (SePH) (Lin et al., 2016), and Label Consistent Matrix Factorization Hashing (LCMFH) (Wang et al., 2019). Recently, deep learning-based supervised hashing (Jiang & Li, 2017; Li et al., 2018; Su, Zhong, & Zhang, 2019; Wu et al., 2019; Yang et al., 2017) has been proposed to adopt the supervised learning strategy. Although achieving state-of-the-art performance, such deep hashing methods only come at the expense of heavy computational costs and substantial hyperparameters adjustments. These challenges make the corresponding deep hashing methods impractical in large-scale data search applications. Therefore, in this paper, we primarily focus on the shallow supervised hashing methods with the applications to cross-modal image–text search.

Despite obtaining promising performance, existing supervised hashing methods are still limited in several respects. (1) They perform the hash code learning procedure without fully considering the relation between the data instances and label supervision. For example, the methods in Wang et al. (2019), Wang, Gao, Wang, He, and Yuan (2016) and Xu, Shen, Yang, Shen, and Li (2017) only encode the semantic label information to directly generate unified hash codes for search tasks, leading to information loss of the data instances. As a result, it cannot well preserve the semantic correlations of the original data and shared class labels to deliver better performance. (2) The discriminability of the hash codes would be weakened when using a continuous relaxation strategy to solve the objective function. In Ding et al. (2014), Liu, Ji, Wu, Huang, and Zhang (2017), Zhang and Li (2014) and Zhou et al. (2014), adopting a relaxation strategy (Li et al., 2017) for the hash optimization, such methods directly convert the original continuous features into unified hash codes by discarding the discrete constraints of their models. Consequently, they inevitably cause the quantization loss of final hash codes and subsequent suboptimal search accuracy. (3) Most approaches are with limited learning efficiency due to the heavy computational load during training. Specifically, Lin et al. (2016), Liu et al. (2016) and Tang et al. (2016) design the training models by constructing large $n \times n$ similarity graphs based on the semantic labels or pairwise data constraints, where n denotes the number of training instances. Generally, $n \times n$ similarity graphs with a large value of n usually produce high computational complexity, which causes the training to be time-consuming and eventually compromises the learning efficiency of hashing methods.

1.2. Research objectives and contributions

Generally, it is convenient to retrieve in the same modality by unimodal search (a.k.a. text-based or image-based retrieval). How to use one modality to search the other modality (i.e., cross-modal search) is more complicated, which is the focus of this paper. As a result, we mainly focus on achieving accurate and fast matching with applications to cross-modal image–text search. Besides, this research has received extensive concentrations for revealing the semantic consistency between vision and language. However, due to the modality gap (a.k.a. the characteristics of heterogeneous data), the most critical challenge is how to transform the image–text data into binary space and also preserve the correspondence between modalities. Therefore, we conclude the main research objectives of this paper and propose a novel supervised hashing method, termed Efficient Discrete Class Alignment Hashing (EDCAH). It not only integrates class alignment and matrix factorization to simultaneously learn unified hash codes and hash functions, but it also utilizes a discrete optimization strategy to solve the proposed algorithm without adopting the relaxation strategy. Moreover, to enhance the learning efficiency of the proposed model, a fast and efficient variant of EDCAH, i.e., EDCAH-t, is further presented by exploiting a two-step hashing scheme to accelerate the EDCAH model. Concretely, the main contributions of this work are as follows:

- We propose a novel supervised hashing method EDCAH. By integrating class alignment, matrix factorization, and optimization strategy, our method simultaneously learns discrete unified hash codes and modality-specific hash functions. And thus it can jointly exploit the correlations of the training instances and semantic labels to improve the discriminative capability of the hash codes.
- A fast and efficient variant of EDCAH, i.e., EDCAH-t, is developed to reduce the training complexity of the proposed model. Specifically, it presents a two-step hashing scheme to learn compact hash codes and effective hash functions. Due to the advantage in learning efficiency, EDCAH-t can be applied to rapid yet practical large-scale cross-modal image–text search applications.

- We develop a novel discrete optimization strategy to efficiently solve the EDCAH model. This strategy optimizes the proposed objective function by a fast and simple operation, which contributes to enhancing the learning efficiency during training. Extensive experimental results on popular datasets validate the superiority of the proposed EDCAH and its variant against state-of-the-art methods.

The remainder of this paper is organized as follows: Section 2 introduces the related works on hashing methods. Section 3 presents the details of the proposed method. Section 4 provides experimental results and analyses. Section 5 concludes this paper.

2. Related work

Research work on cross-modal image–text hashing search methods can be broadly classified as follows: (1) unsupervised hashing methods; (2) supervised hashing methods.

2.1. Unsupervised hashing methods

Unsupervised hashing methods learn the hash codes or hash functions by capturing the training data characteristics for cross-modal image–text search tasks. For instance, Kumar and Udupa (2011) and Song, Yang, Yang, Huang, and Shen (2013) produced Cross-view hashing (CVH) and Inter-media hashing (IMH) methods, respectively. They learn the hash functions by extending spectral hashing (Weiss, Torralba, & Fergus, 2009) from unimodal to multimodal retrieval scenarios. Zhu, Huang, Shen, and Zhao (2013) constructed Linear Cross-modal Hashing (LCMH) which utilizes the anchor maps to maintain the similarity between modalities for hashing learning. Zhou et al. (2014) presented Latent Semantic Sparse Hashing (LSSH), to first obtain the image and text features by sparse coding and matrix factorization and then project these features into a common space to learn unified binary codes. Ding et al. (2014) proposed Collective Matrix Factorization Hashing (CMFH), which learns unified hash codes by adopting matrix factorization to discover the correlations between modalities. Long, Cao, Wang, and Yu (2016) formulated Composite Correlation Quantization (CCQ) to project different modalities into a shared space and express composite quantizers that transform common representations into compact binary codes. Other typical of methods can be referred to Liu et al. (2017) and Weiss et al. (2009). However, these above methods cannot preserve well-learned semantic similarities of the original data without supervised information, obtaining suboptimal search performance.

2.2. Supervised hashing methods

In contrast, supervised hashing methods study the hash codes or hash functions by exploring the semantic correlations from supervised information. For instance, Liu et al. (2016) offered Supervised Matrix Factorization Hashing (SMFH), which generates a graph regularized matrix factorization model to obtain the hash codes. Tang et al. (2016) outlined a homologous supervised matrix factorization hashing (SMFH) method to maintain the local geometric consistency regarding each modality and the semantic similarity from the labels for generating unified hash codes. Zhang and Li (2014) reported Semantic Correlation Maximization (SCM), which explores the semantic similarities between modalities by joining the class matrices of the instances. Lin et al. (2016) illustrated Semantics-preserving Hashing (SePH), which achieves the binary codes via the Kullback–Leibler divergence of the probability distributions, and then learns the hash functions by leveraging the predictive models. However, these above methods possess heavy computation costs by constructing large $n \times n$ (a.k.a. training size) similarity graphs, making them unsuitable for large-scale datasets. To tackle the optimization problem in learning efficiency, Xu et al. (2017) created Discriminative Cross-modal Hashing (DCH), to obtain the unified hash codes by a discrete coordinate descent strategy and then to produce the different hash functions via learned codes. Wu, Luo, Xu, Guo, and Shi (2018) devised Dictionary Learning based Supervised Discrete Hashing (DLSDH), which learns the sparse representations for distinct data modalities, and then final hash codes are obtained by using a bit-by-bit operation. Wang et al. (2019) designed Label Consistent Matrix Factorization Hashing (LCMFH), which yields the discriminative unified hash codes by integrating the shared label matrix and collective matrix factorization. Shen et al. (2021) implemented a novel Subspace Relation Learning for Cross-modal Hashing (SRLCH) method by leveraging the transformed labels and subspace relation information to deliver the unified hash codes. However, such methods fail to consider the joint relationship between distinct modalities and thus formulate less effective hash codes for cross-modal image–text search applications. Meanwhile, several methods learn the unified hash codes by a bit-by-bit optimization strategy, making the training procedure rather time-consuming.

Different from the aforementioned studies, the proposed EDCAH and its variant EDCAH-t accomplish the search tasks by integrating the hash code learning procedure, a two-step hashing scheme, and discrete optimization strategy into a unified framework, which contributes to improving the search performance and learning efficiency. Despite obtaining promising results, existing supervised hashing methods still have much room to enhance the field of cross-modal image–text search.

3. The proposed approach

In this section, we elaborate on the details of the proposed EDCAH method and its variant EDCAH-t, and their pipeline is presented in Fig. 1. This study mainly focuses on the image and text modalities for cross-modal image–text search. For convenience, we present the main notations used in the study in Table 1.

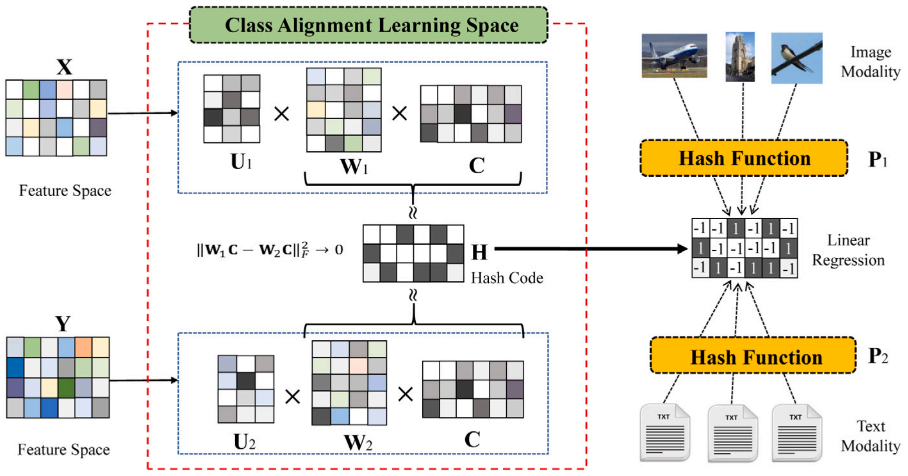

Fig. 1. The pipeline of the proposed EDCAH. The training procedure consists of two subsections: hash code learning (red dotted box) and hash function learning (black dotted arrow). Specifically, EDCAH simultaneously learns the hash codes and hash functions by integrating class alignment and matrix factorization to generate unified hash codes while preserving the semantic consistency of data instances and label supervision. The proposed variant EDCAH-t employs the two-step hashing scheme by the linear regression to learn the modality-specific hash functions. The two ways of learning hash functions are the main difference between EDCAH and EDCAH-t. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Notations used in this study for the proposed method.

Notation	Definition
$\mathcal{O}$	Training instances of image–text data $\mathcal{O} = \{X, Y\}$
$\mathbf{X}, \mathbf{Y}$	Image feature matrix $\mathbf{X} \in \mathbb{R}^{d_1\times n}$, text feature matrix $\mathbf{Y} \in \mathbb{R}^{d_2\times n}$
$\mathbf{C}$	Semantic label matrix based on pairwise image–text data $\mathbf{C} \in \mathbb{R}^{c\times n}$
$\mathbf{H}$	Unified hash code matrix of the training instances $\mathbf{H} \in \mathbb{R}^{l\times n}$
$\mathbf{U}_1, \mathbf{U}_2$	Basic matrices for the matrix factorization $\mathbf{U}_1 \in \mathbb{R}^{d_1\times l}$, $\mathbf{U}_2 \in \mathbb{R}^{d_2\times l}$
$\mathbf{W}_1, \mathbf{W}_2$	Projection matrices for the class alignment $\mathbf{W}_1 \in \mathbb{R}^{l\times c}$, $\mathbf{W}_2 \in \mathbb{R}^{l\times c}$
$\mathbf{P}_1, \mathbf{P}_2$	Projection matrices for the hash functions $\mathbf{P}_1 \in \mathbb{R}^{l\times d_1}$, $\mathbf{P}_2 \in \mathbb{R}^{l\times d_2}$
$\phi(\bullet)$	Kernel function for the input of feature matrices
d_1, d_2	Dimensions of the image, text feature matrices
n	Number of the training instances
c	Number of the label classes
l	Length of unified hash codes

3.1. Model formulation for EDCAH

Given the feature matrices $\mathbf{X}$, $\mathbf{Y}$ and the corresponding class label matrix $\mathbf{C}$, we first have the m-dimensional feature vectors by $\phi(\mathbf{K}^{(t)}) = \left\{\phi\left(\mathbf{k}_i^t\right)\right\}_{i=1}^n$ for the model input, where $\mathbf{K}^{(t)} = \{\mathbf{X}, Y\}$, and such vectors are computed by Gaussian kernel function

$$\phi(k^{(t)}) = \left[\exp\left(-\left\|k^{(t)} - p_1^{(t)}\right\|^2/\varepsilon\right), \ldots, \exp\left(-\left\|k^{(t)} - p_{\mathrm{m}}^{(t)}\right\|^2/\varepsilon\right)\right]^T, \tag{1}$$

where $\left\{k_j^{(t)}\right\}_{j=1}^m$ includes m randomly selected anchor points, ε is the kernel width. The main idea of EDCAH is to adopt class alignment and matrix factorization to learn unified hash codes $\mathbf{H}$ and hash functions $\mathbf{P}_1$, $\mathbf{P}_2$ in the same optimization manner, where $\mathbf{H} \in \{-1, 1\}^{l\times n}$ in the following. The overall objective function of EDCAH is as follows:

$$\min_{\mathbf{U}_1,\mathbf{U}_2,\mathbf{W}_1,\mathbf{W}_2,\mathbf{P}_1,\mathbf{P}_2} \{J\left(\mathbf{U}_1, \mathbf{U}_2, \mathbf{W}_1, \mathbf{W}_2, \mathbf{P}_1, \mathbf{P}_2\right)\}. \tag{2}$$

The detailed formulation of Eq. (2) is represented as follows:

$$\begin{aligned} J =&\lambda_1 \|\phi(\mathbf{X}) - \mathbf{U}_1\mathbf{W}_1\mathbf{C}\|_F^2 + (1-\lambda_1)\|\phi(\mathbf{Y}) - \mathbf{U}_2\mathbf{W}_2\mathbf{C}\|_F^2 \\ &+ \lambda_2\left(\|\mathbf{W}_1\mathbf{C} - \mathbf{H}\|_F^2 + \|\mathbf{W}_2\mathbf{C} - \mathbf{H}\|_F^2\right) + \lambda_3\|\mathbf{W}_1\mathbf{C} - \mathbf{W}_2\mathbf{C}\|_F^2 \\ &+ \lambda_4\left(\|\mathbf{H}-\mathbf{P}_1\phi(\mathbf{X})\|_F^2 + \|\mathbf{H}-\mathbf{P}_2\phi(\mathbf{Y})\|_F^2\right) \\ &+ \lambda_5 Re\left(\mathbf{U}_1, \mathbf{U}_2, \mathbf{W}_1\mathbf{C},\ \mathbf{W}_2\mathbf{C},\ \mathbf{P}_1,\ \mathbf{P}_2\right), \end{aligned} \tag{3}$$

where the parameters λ_1, λ_2, λ_3, λ_4 and λ_5 are trade-off coefficients, and $Re(\cdot) = \|\cdot\|_F^2$ denotes the regularization term to avoid the over-fitting of our model. The first four terms definite the hash code learning procedure. The five term represents the task of learning the hash functions, while the last term is the regularization formulation. In the following, we explain each part of the above Eq. (3) in detail.

Hash Code Learning

The proposed EDCAH learns unified hash codes by the first four terms of Eq. (3). As previously mentioned, matrix factorization is essentially an unsupervised algorithm that is inapplicable for supervised hashing methods. Therefore, we combine class label alignment with matrix factorization to learn the semantic relations of image–text data. Given the feature matrices $\mathbf{X}$, $\mathbf{Y}$ and the class label matrix $\mathbf{C}$, our method uses class alignment matrix factorization to find the basic matrices $\mathbf{U}_1 \in \mathbb{R}^{d_1\times l}$ and $\mathbf{U}_2 \in \mathbb{R}^{d_2\times l}$ and the different latent modality-specific representation matrices $\mathbf{W}_1\mathbf{C} \in \mathbb{R}^{l\times n}$ and $\mathbf{W}_2\mathbf{C} \in \mathbb{R}^{l\times n}$. Specifically, we define the following formula:

$$\begin{aligned}\min_{\mathbf{U}_1,\mathbf{U}_2,\mathbf{W}_1,\mathbf{W}_2} &\left\{\lambda_1 \|\phi(\mathbf{X}) - \mathbf{U}_1\mathbf{W}_1\mathbf{C}\|_F^2 + (1-\lambda_1)\|\phi(\mathbf{Y}) - \mathbf{U}_2\mathbf{W}_2\mathbf{C}\|_F^2\right.\\ &\left.+\lambda_2\left(\|\mathbf{W}_1\mathbf{C}-\mathbf{H}\|_F^2 + \|\mathbf{W}_2\mathbf{C}-\mathbf{H}\|_F^2\right) + \lambda_3\|\mathbf{W}_1\mathbf{C}-\mathbf{W}_2\mathbf{C}\|_F^2\right\}.\end{aligned} \tag{4}$$

The first two items denote class alignment matrix factorization for correlating distinct modalities. The last two items directly produce unified hash codes by embedding the semantic label matrix into the binary code learning procedure, which circumvents the use of large similarity graphs.

Hash Function Learning

In the hash function learning, the two hash functions are obtained by linear regression and transferred to the querying stage for out-of-sample instances. The mathematical definition is as follows:

$$\min_{\mathbf{P}_1,\mathbf{P}_2} \lambda_4\left\{\|\mathbf{H}-\mathbf{P}_1\phi(\mathbf{X})\|_F^2 + \|\mathbf{H}-\mathbf{P}_2\phi(\mathbf{Y})\|_F^2\right\}. \tag{5}$$

3.2. Model formulation for EDCAH-t

To improve the learning efficiency of the EDCAH, we further propose its variant EDCAH-t, which adopts a two-step hashing scheme to reduce the model complexity. The overall objective function of EDCAH-t is formulated as:

$$\min_{\mathbf{U}_1,\mathbf{U}_2,\mathbf{W}_1,\mathbf{W}_2}\{F(\mathbf{U}_1,\mathbf{U}_2,\mathbf{W}_1,\mathbf{W}_2)\}. \tag{6}$$

The detailed formulation of Eq. (6) is as follows:

$$\begin{aligned}J =&\mu_1\|\phi(\mathbf{X}) - \mathbf{U}_1\mathbf{W}_1\mathbf{C}\|_F^2 + (1-\mu_1)\|\phi(\mathbf{Y}) - \mathbf{U}_2\mathbf{W}_2\mathbf{C}\|_F^2\\ &+\mu_2\left(\|\mathbf{W}_1\mathbf{C}-\mathbf{H}\|_F^2 + \|\mathbf{W}_2\mathbf{C}-\mathbf{H}\|_F^2\right) + \mu_3\|\mathbf{W}_1\mathbf{C}-\mathbf{W}_2\mathbf{C}\|_F^2\\ &+\mu_5 Re\left(\mathbf{U}_1,\mathbf{U}_2,\mathbf{W}_1\mathbf{C},\ \mathbf{W}_2\mathbf{C}\right),\end{aligned} \tag{7}$$

where μ_1, μ_2, μ_3 and μ_5 are trade-off coefficients. Analogous to the model formulation of EDCAH, the objective function of EDCAH-t is composed of two terms. Its optimization target is to first generate the hash codes by minimizing Eq. (7) and then to learn the hash functions by minimizing Eq. (16).

3.3. Optimization strategy

The optimization problems of Eqs. (2) and (6) are non-convex for all matrix variables but convex for any variable when the others are fixed (Ding et al., 2014). In the following, we solve these optimization problems through the proposed optimization strategy.

Optimization Strategy for EDCAH: The optimization problem of Eq. (2) can be solved by the proposed optimization strategy. The detailed description consists of four iterated main steps.

1. Updating $\mathbf{U}_1$ and $\mathbf{U}_2$. By fixing the other variables and letting $\frac{\partial J}{\partial \mathbf{U}_1}=0$, $\frac{\partial J}{\partial \mathbf{U}_2}=0$, we have

$$\begin{aligned}\mathbf{U}_1 &= \lambda_1\phi(\mathbf{X})\mathbf{C}^{\mathrm{T}}\mathbf{W}_1{}^{\mathrm{T}}\left(\lambda_1\mathbf{W}_1\mathbf{C}\mathbf{C}^{\mathrm{T}}\mathbf{W}_1{}^{\mathrm{T}}+\lambda_5\mathbf{I}\right)^{-1},\\ \mathbf{U}_2 &= \lambda_1\phi(\mathbf{Y})\mathbf{C}^{\mathrm{T}}\mathbf{W}_2{}^{\mathrm{T}}\left(\lambda_1\mathbf{W}_2\mathbf{C}\mathbf{C}^{\mathrm{T}}\mathbf{W}_2{}^{\mathrm{T}}+\lambda_5\mathbf{I}\right)^{-1}.\end{aligned} \tag{8}$$

2. Updating $\mathbf{W}_1$ and $\mathbf{W}_2$. By fixing the other variables and letting $\frac{\partial J}{\partial \mathbf{U}_1}=0$, $\frac{\partial J}{\partial \mathbf{U}_2}=0$, we obtain

$$\begin{aligned}\mathbf{W}_1 =&\left[\lambda_1\mathbf{U}_1{}^{\mathrm{T}}\mathbf{U}_1 + (\lambda_2+\lambda_3+\lambda_5)\,\mathbf{I}\right]^{-1}\\ &\times\left[\lambda_1\mathbf{U}_1{}^{\mathrm{T}}\phi(\mathbf{X})\mathbf{C}^{\mathrm{T}}+\lambda_2\mathbf{H}\mathbf{C}^{\mathrm{T}}+\lambda_3\mathbf{W}_2\mathbf{C}\mathbf{C}^{\mathrm{T}}\right]\left[\mathbf{C}\mathbf{C}^{\mathrm{T}}\right]^{-1},\\ \mathbf{W}_2 =&\left[\lambda_1\mathbf{U}_2{}^{\mathrm{T}}\mathbf{U}_2 + (\lambda_2+\lambda_3+\lambda_5)\,\mathbf{I}\right]^{-1}\\ &\times\left[\lambda_1\mathbf{U}_2{}^{\mathrm{T}}\phi(\mathbf{Y})\mathbf{C}^{\mathrm{T}}+\lambda_2\mathbf{H}\mathbf{C}^{\mathrm{T}}+\lambda_3\mathbf{W}_1\mathbf{C}\mathbf{C}^{\mathrm{T}}\right]\left[\mathbf{C}\mathbf{C}^{\mathrm{T}}\right]^{-1}.\end{aligned} \tag{9}$$

3. Updating $\mathbf{P}_1$ and $\mathbf{P}_2$. By fixing the other variables and letting $\frac{\partial J}{\partial \mathbf{P}_1}=0$, $\frac{\partial J}{\partial \mathbf{P}_2}=0$, the solutions are

$$\begin{aligned}\mathbf{P}_1 =&\lambda_4\mathbf{H}\phi\left(\mathbf{X}^{\mathrm{T}}\right)\left(\lambda_4\phi(\mathbf{X})\phi\left(\mathbf{X}^{\mathrm{T}}\right)+\lambda_5\mathbf{I}\right)^{-1},\\ \mathbf{P}_2 =&\lambda_4\mathbf{H}\phi\left(\mathbf{Y}^{\mathrm{T}}\right)\left(\lambda_4\phi(\mathbf{Y})\phi\left(\mathbf{Y}^{\mathrm{T}}\right)+\lambda_5\mathbf{I}\right)^{-1}.\end{aligned} \tag{10}$$

4. Updating H. By fixing the other variables, Eq. (2) can be rewritten as follows:

$$\begin{aligned}\min_{\mathbf{H}} \Big\{ &\lambda_2 \left(\|\mathbf{W}_1\mathbf{C}-\mathbf{H}\|_F^2 + \|\mathbf{W}_2\mathbf{C}-\mathbf{H}\|_F^2 \right) \\ &+\lambda_4 \left(\|\mathbf{H}-\mathbf{P}_1\phi(\mathbf{X})\|_F^2 + \|\mathbf{H}-\mathbf{P}_2\phi(\mathbf{Y})\|_F^2 \right) \Big\}.\end{aligned} \tag{11}$$

The two terms in Eq. (11) are equivalent to the following Eq. (12):

$$\begin{aligned}\|\mathbf{W}_1\mathbf{C}-\mathbf{H}\|_F^2 &= tr\left[(\mathbf{W}_1\mathbf{C}-\mathbf{H})^{\mathrm{T}} (\mathbf{W}_1\mathbf{C}-\mathbf{H}) \right] \\ &= tr\left(\mathbf{C}^{\mathrm{T}}\mathbf{W}_1{}^{\mathrm{T}}\mathbf{W}_1\mathbf{C}\right) - 2tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{W}_1\mathbf{C}\right) + tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right), \\ \|\mathbf{H}-\mathbf{P}_1\mathbf{X}\|_F^2 &= tr\left[(\mathbf{H}-\mathbf{P}_1\mathbf{X})^{\mathrm{T}} (\mathbf{H}-\mathbf{P}_1\mathbf{X}) \right] \\ &= tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right) - 2tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{P}_1\mathbf{X}\right) + tr\left(\mathbf{X}^{\mathrm{T}}\mathbf{P}_1{}^{\mathrm{T}}\mathbf{P}_1\mathbf{X}\right).\end{aligned} \tag{12}$$

Similarly, the other two terms can be easily obtained, analogous to Eq. (12). Notably, $tr(\mathbf{H}^{\mathrm{T}}\mathbf{H})$ is constant under the condition $\mathbf{H} \in \{-1,1\}^{l\times n}$ in Section 3.1. Therefore, these four trace matrices $tr\left(\mathbf{C}^{\mathrm{T}}\mathbf{W}_1{}^{\mathrm{T}}\mathbf{W}_1\mathbf{C}\right)$, $tr\left(\phi\left(\mathbf{X}^{\mathrm{T}}\right)\mathbf{P}_1{}^{\mathrm{T}}\mathbf{P}_1\phi(\mathbf{X})\right)$, $tr\left(\phi\left(\mathbf{Y}^{\mathrm{T}}\right)\mathbf{P}_2{}^{\mathrm{T}}\mathbf{P}_2\phi(\mathbf{Y})\right)$ and $tr\left(\mathbf{C}^{\mathrm{T}}\mathbf{W}_2{}^{\mathrm{T}}\mathbf{W}_2\mathbf{C}\right)$ are constant when updating $\mathbf{H}$ by fixing the other variables. Consequently, Eq. (11) can be rewritten as:

$$\begin{aligned}-\min_{\mathbf{H}} \Big\{ &\lambda_2 tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{W}_1\mathbf{C}\right) + \lambda_2 tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{W}_2\mathbf{C}\right) \\ &+\lambda_4 tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{P}_1\phi(\mathbf{X})\right) + \lambda_4 tr\left(\mathbf{H}^{\mathrm{T}}\mathbf{P}_2\phi(\mathbf{Y})\right) \Big\}.\end{aligned} \tag{13}$$

Finally, we obtain the closed-form solution of $\mathbf{H}$:

$$\mathbf{H} = \mathrm{sgn}\left(\lambda_2\mathbf{W}_1\mathbf{C} + \lambda_2\mathbf{W}_2\mathbf{C} + \lambda_4\mathbf{P}_1\phi(\mathbf{X}) + \lambda_4\mathbf{P}_2\phi(\mathbf{Y})\right), \tag{14}$$

where $\mathrm{sgn}(\cdot)$ is a sign function that converts a continuous value into binary code.

According to Eq. (14), the unified hash codes can be learned discretely during the optimization procedure. Thus, rather than adopting the continuous relaxation strategy, **H** avoids the large quantization errors of the hash codes by implementing the proposed discrete optimization strategy. The training procedure of EDCAH is summarized in Algorithm 1. The time complexity of Algorithm 1 is determined mainly by the number of iterations (lines 2–7).

Algorithm 1 EDCAH Training Procedure

```
Input:
   Feature matrices $\mathbf{X}$, $\mathbf{Y}$, label matrix $\mathbf{C}$, parameters $\lambda_1$, $\lambda_2$, $\lambda_3$, $\lambda_4$, $\lambda_5$, and number of iterations $w$.
Output:
   Hash codes $\mathbf{H}$, projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$.
1: Randomly initialize $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{P}_1$, $\mathbf{P}_2$.
2: for $j = 1$ to $w$ do
3:   Calculate $\mathbf{U}_1$ and $\mathbf{U}_2$ by using Eq. (8);
4:   Calculate $\mathbf{W}_1$ and $\mathbf{W}_2$ by using Eq. (9);
5:   Calculate $\mathbf{P}_1$ and $\mathbf{P}_2$ by using Eq. (10);
6:   Calculate $\mathbf{H}$ by using Eq. (14).
7: end for
8: return $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{P}_1$, $\mathbf{P}_2$, and $\mathbf{H}$.
```

Optimization Strategy for EDCAH-t: By observing the form of Eqs. (2) and (6), we find that the process for generating the hash codes of EDCAH is similar to that of EDCAH-t. The main difference is the learning procedure of the hash functions in Eq. (16). Analogous to Algorithm 1, the optimization procedure of Eq. (6) includes three main steps. Specifically, the solutions of $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{W}_1$ and $\mathbf{W}_2$ are the same as those in Eqs. (8) and (9), respectively. Analogous to Eq. (14), we can obtain the closed-form solution of the hash codes $\mathbf{H}$ for EDCAH-t as follows:

$$\mathbf{H} = \mathrm{sgn}\left(\mu_2\mathbf{W}_1\mathbf{C} + \mu_2\mathbf{W}_2\mathbf{C}\right). \tag{15}$$

Moreover, we learn the hash functions based on Eq. (15) by employing linear regression in the two-step hashing scheme:

$$\|\mathbf{H}-\mathbf{P}_t\phi(\mathbf{A})\|_F^2 + \mu_4\|\mathbf{P}_t\|_F^2, \tag{16}$$

where $\mathbf{P}_t = \left\{\mathbf{P}_1, \mathbf{P}_2\right\}$, $\mathbf{A} = \{\mathbf{X}, \mathbf{Y}\}$. Thus, the final closed-form solution of the hash function is:

$$\mathbf{P}_t = \mathbf{H}\phi\left(\mathbf{A}^{\mathrm{T}}\right)\left(\phi(\mathbf{A})\phi\left(\mathbf{A}^{\mathrm{T}}\right) + \mu_4\mathbf{I}\right)^{-1}. \tag{17}$$

The optimization process for EDCAH-t is shown in Algorithm 2, and the time complexity depends on the number of iterations in lines 2–6. Algorithm 2 has two main advantages against Algorithm 1: (1) it obtains more compact hash codes; (2) it converges faster that contributes to shortening the training time of EDCAH in hashing learning efficiency.

Algorithm 2 EDCAH-t Training Procedure

```
Input:
    Feature matrices $\mathbf{X}$, $\mathbf{Y}$, label matrix $\mathbf{C}$, parameters $\mu_1$, $\mu_2$, $\mu_3$, $\mu_4$, $\mu_5$, and number of iterations $w$.
Output:
    Hash codes $\mathbf{H}$.
1: Randomly initialize $\mathbf{U}_1$, $\mathbf{U}_2$, $\mathbf{W}_1$, and $\mathbf{W}_2$.
2: for $j = 1$ to $w$ do
3:     Calculate $\mathbf{U}_1$ and $\mathbf{U}_2$ by using Eq. (8);
4:     Calculate $\mathbf{W}_1$ and $\mathbf{W}_2$ by using Eq. (9);
5:     Calculate $\mathbf{H}$ by using Eq. (15).
6: end for
7: return H.
```

3.4. Out-of-sample extension for querying

For any a new query data x or y, the hash codes h are formulated by the learned projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$. For EDCAH, we implement the hash functions $h(x)$ and $h(y)$ under out-of-sample extension via Eq. (18):

$$\begin{aligned} h(x) &= \operatorname{sgn}\left(\mathbf{P}_1 x\right), \\ h(y) &= \operatorname{sgn}\left(\mathbf{P}_2 y\right). \end{aligned} \tag{18}$$

For EDCAH-t, according to Eq. (17), the hash functions are obtained as follows:

$$h(a) = \operatorname{sgn}\left(\mathbf{P}_t a\right), \text{ here } a \in \{x, y\}. \tag{19}$$

4. Experiments and analysis

To validate the superiority of EDCAH and its variant EDCAH-t, we conduct extensive experiments on three public datasets and make comparisons with several state-of-the-art hashing methods regarding both search accuracy and learning efficiency.

4.1. Experimental setups

In this subsection, we introduce popular datasets, evaluation metrics, baselines and implementation details for a series of cross-modal image–text search experiments.

Datasets: We evaluate the search performance of the proposed EDCAH and its variant on Wiki (Pereira et al., 2013), MIRFLICKR-25K (Huiskes & Lew, 2008), and NUS-WIDE (Chua et al., 2009). The first is a single-label dataset and the last two are multi-label datasets. The detailed descriptions of these public datasets are reported in Table 2.

Wiki is composed of 2866 image–text pairs collected from Wikipedia. Each instance belongs to one of ten frequently used topics, such as warfare, art, or sky. For an instance, each image is represented as a 128-dimensional scale-invariant feature transform feature vector, and each text is denoted by a 10-dimensional Latent Dirichlet Allocation topic vector. We follow Wang et al. (2019) to randomly pick up 2173 instances as the training and retrieval set, and the remaining instances as the query set.

MIRFILCKR-25K includes approximately 25,000 instances from Flickr. Each image is annotated by several user-assigned tags with some of the 24 provided labels. For each instance, the image is represented by a 512-dimensional generalized search trees feature vector, and the text is expressed as a 1386-dimensional bag of words vector. Following Ji et al. (2017), we randomly select 2000 instances as the query set and use the remaining instances as the training and retrieval set.

NUS-WIDE contains approximately 270,000 images with annotated tags from 81 provided semantic concepts. Following the setting in Jiang and Li (2019), we choose the 10 most widely-used semantic concepts (including 186,577 images) as the research data. Each image is represented as a 500-dimensional bag of visual words vector, and the corresponding text is represented as a 1000-dimensional bag of words vector. Based on the protocol in Jiang and Li (2019), we randomly select 2000 instances as the query set, and rest instances as the retrieval set. Besides, we sample 10,000 instances from the remaining 184,577 instances as the training set.

Evaluation Metrics: We adopt three standard evaluation metrics (Gong, Lazebnik, Gordo, & Perronnin, 2013), namely, mean Average Precision (**mAP**), Precision–Recall curve (**PR**) and topN-precision curve (**topN**), to evaluate the performance of the proposed method. The mAP value is the mean of the average precision (AP) of each instance:

$$AP = \frac{1}{L} \sum_{r=1}^{R} P(r)\xi(r), \tag{20}$$

where L is the number of relevant instances in the retrieved results, $P(r)$ denotes the precision of the top r training instances, and R is the size of the query set in the experimental setting. $\xi(r)$ is an indicator function. If the rth instance is similar to q, $\xi(r) = 1$; otherwise, $\xi(r) = 0$.

Table 2
The details of the three evaluated datasets.

Dataset	Wiki	MIRFILICKR-25K	NUS-WIDE
Total	2866	20,015	186,577
Query set	693	2000	2000
Training set	2173	18,015	10,000
Retrieval set	2173	18,015	184,577
Image feature	128-d SIFT	512-d GIST	500-d BOVW
Text feature	10-d LDA	1386-d BOW	1000-d BOW

The PR curve and the topN curve are widely used as metrics in the existing hashing-based cross-modal image–text search studies. The former reflects the precision at different recall levels, and the latter denotes the variation in precision among the top-ranked N image or text instances. For these evaluation metrics, a larger score indicates better performance by the hashing method (Chen et al., 2020).

Baselines and Implementation Details: We select several state-of-the-art hashing methods as baselines for comparison with our method, including unsupervised hashing (LSSH (Zhou et al., 2014), CMFH (Ding et al., 2014)) and supervised hashing (SCM-seq (Zhang & Li, 2014), SePH (Lin et al., 2016), SMFH (Liu et al., 2016), LCMFH (Wang et al., 2019), SRLCH (Shen et al., 2021)). The corresponding source codes of all the baselines were kindly provided by the authors, while we complete the code implementation for LCMFH based on the CMFH source. We implement such baselines with suggested by their papers and report their best results. The parameter settings of the proposed EDCAH and its variant are obtained by the grid search method (Wang et al., 2021; Zheng et al., 2020). Specifically, the best performance of EDCAH is obtained when $\{\lambda_1 = 0.5, \lambda_2 = 100, \lambda_3 = 10, \lambda_4 = 1, \lambda_5 = 0.001\}$, $\{\lambda_1 = 0.5, \lambda_2 = 0.1, \lambda_3 = 10, \lambda_4 = 0.1, \lambda_5 = 0.001\}$ and $\{\lambda_1 = 0.5, \lambda_2 = 0.1, \lambda_3 = 10, \lambda_4 = 0.1, \lambda_5 = 0.001\}$ on Wiki, MIRFLICKR-25K and NUS-WIDE, respectively. And the best performance of EDCAH-t is achieved when $\{\mu_1 = 0.5, \mu_2 = 10, \mu_3 = 0.1, \mu_4 = 0.1, \mu_5 = 0.001\}$, $\{\mu_1 = 0.5, \mu_2 = 1, \mu_3 = 0.01, \mu_4 = 0.1, \mu_5 = 0.001\}$ and $\{\mu_1 = 0.5, \mu_2 = 1, \mu_3 = 0.01, \mu_4 = 0.1, \mu_5 = 0.001\}$ on Wiki, MIRFLICKR-25K and NUS-WIDE, respectively. Due to space limitations, we present the experimental results for only two hash bit sizes (32 bits and 64 bits) in the evaluation metrics.

In the experiments, we conduct two cross-modal image–text search tasks: I→T and T→I, where I→T denotes that the images are used as queries to retrieve relevant texts, and T→I indicates that the texts are used as queries to search for relevant images. All comparison methods are implemented on MATLAB 2016a, whose server configurations are a workstation equipped with an Intel(R) Core(TM) CPU i9-9820X @ 3.3 GHz, 128 GB memory.

4.2. Search accuracy comparison

Accuracy on Wiki: To demonstrate the feasibility of EDCAH, we compare it with all the baselines on Wiki. Table 3 shows the mAP values of all comparison methods. We can observe that: (1) EDCAH achieves the best scores compared to all the baselines on both search tasks. For example, EDCAH improves the mAP value from 21.86% (LSSH), 25.16% (SMFH), 24.91% (SePH), 32.88% (LCMFH), 34.55% (SRLCH) to 37.10% for 32 bits in the I→T task. The main reason for this improvement is that the joint learning of class alignment and matrix factorization produces high-quality hash codes. (2) Moreover, the mAP scores of all the compared methods gradually improve as the hash code length increases from 16 to 128. Specifically, the mAP scores of EDCAH-t are increased in a progression of 35.52% (16 bits), 37.90% (32 bits), 39.85% (64 bits) and 40.04% (128 bits). This result shows that longer hash codes contain more useful information. (3) For all the baselines, the mAP scores of the T→I tasks work better than that of I→T tasks. The main explanation is that the text features of the image–text pairs can better express the semantic information (e.g., structural characteristics, attribute characteristics) against the image features.

Figs. 2 and 3 show the PR and topN results of all comparison methods @ 32 bits and 64 bits hash codes on Wiki. The following can be observed: (1) The proposed EDCAH and its variant achieve the best and second-best precision on both search tasks, similar to the mAP results. (2) As the recall improves, the precision scores of all the baselines show a sharp decreasing tendency. However, EDCAH and EDCAH-t still outperform other methods. (3) EDCAH and its variant also outperform the comparison methods in terms of the topN metric, which is consistent with that of the PR and mAP. To summarize, EDCAH yields the best results of all the tested hashing methods concerning the mAP, PR, and topN metrics, demonstrating the effectiveness of the proposed method.

Accuracy on MIRFLICKR-25K: Table 4 shows the mAP values of all comparison methods at four code lengths. We observe that (1) EDCAH and its variant outperform all baselines on both search tasks. For example, EDCAH improves the mAP values from 58.85% (LSSH), 69.93% (SePH), 76.03% (LCMFH), 71.70% (SRLCH) to 78.89% @ 32 bits on the T→I task. (2) Specifically, the mAP values of EDCAH can be up to 14.66% for I→T and 19.99% for T→I compared to unsupervised hashing (LSSH), and 2.97% for I→T, 2.69% for T→I against supervised hashing (LCMFH). One potential explanation is that EDCAH utilizes the class alignment matrix factorization and optimization strategy to generate unified discrete hash codes. (3) The difference between the mAP results of two cross-modal image–text search tasks on the MIRFLICKR-25K dataset is smaller than that of the Wiki dataset. One possible reason is that the image quality of Wiki is poor, and the relevance between image and semantic tag is not high. Another possible reason may be that the images of MIRFLICKR-25K have corresponding tags and attached annotations while the size of this dataset is 25,000, which greatly reduces the difference between different modalities to improve search performance.

Table 3
The mAP scores of all comparison methods on Wiki.

Methods	I→T				T→I			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
LSSH (Zhou et al., 2014)	0.242	0.219	0.230	0.249	0.620	0.636	0.632	0.633
CMFH (Ding et al., 2014)	0.245	0.261	0.259	0.267	0.605	0.638	0.632	0.649
SMFH (Liu et al., 2016)	0.232	0.252	0.228	0.247	0.579	0.630	0.649	0.670
SCM-seq (Zhang & Li, 2014)	0.247	0.236	0.239	0.259	0.382	0.448	0.442	0.441
SePH (Xu et al., 2017)	0.242	0.249	0.260	0.261	0.682	0.694	0.698	0.690
LCMFH (Wang et al., 2019)	0.327	0.329	0.352	0.364	0.698	0.688	0.703	0.735
SRLCH (Shen et al., 2021)	0.332	0.346	0.365	0.372	0.715	0.728	0.730	0.755
EDCAH	**0.359**	0.371	0.387	0.390	**0.739**	0.753	0.759	0.769
EDCAH-t	0.355	**0.379**	**0.399**	**0.400**	0.732	**0.759**	**0.760**	**0.772**

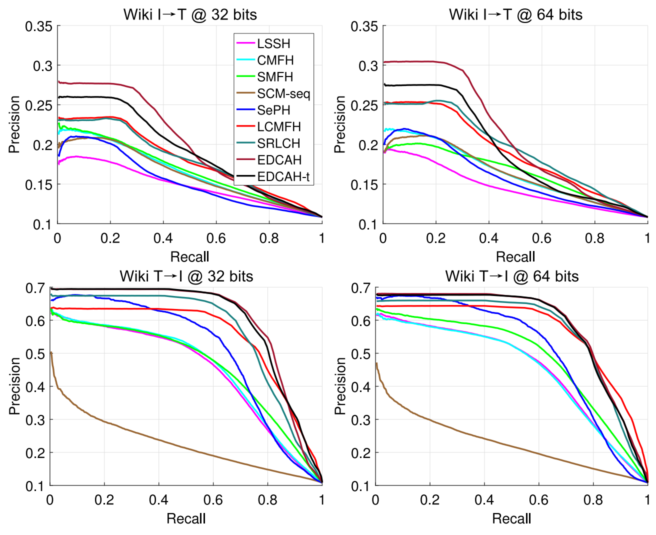

Fig. 2. PR curves of different methods @ 32 bits and 64 bits on Wiki.

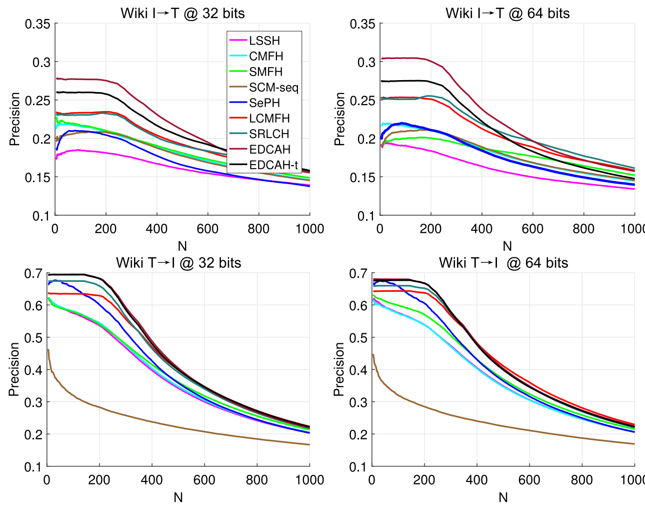

Fig. 3. topN curves of different methods @ 32 bits and 64 bits on Wiki.

Figs. 4 and 5 show the PR and topN curves of all comparison methods @ 32 bits and 64 bits on MIRFLICKR-25K, respectively. EDCAH and EDCAH-t achieve better precision but slightly underperform SRLCH on both search tasks respectively. The possible explanation may be that SRLCH employs the transformed label information to obtain more discriminative hash codes for the search tasks. Although EDCAH and EDCAH-t are slightly inferior to SRLCH on topN metric, they still perform better on mAP and PR, and

Table 4
The mAP scores of all comparison methods on MIRFLICKR-25K.

Methods	I→T				T→I			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
LSSH (Zhou et al., 2014)	0.577	0.575	0.574	0.576	0.586	0.589	0.589	0.588
CMFH (Ding et al., 2014)	0.583	0.582	0.580	0.578	0.588	0.586	0.586	0.583
SMFH (Liu et al., 2016)	0.612	0.620	0.642	0.664	0.623	0.621	0.666	0.698
SCM-seq (Zhang & Li, 2014)	0.625	0.634	0.643	0.649	0.639	0.650	0.659	0.665
SePH (Xu et al., 2017)	0.652	0.652	0.655	0.660	0.692	0.699	0.705	0.704
LCMFH (Wang et al., 2019)	0.679	0.693	0.697	0.701	0.735	0.760	0.774	0.775
SRLCH (Shen et al., 2021)	0.654	0.668	0.693	0.705	0.691	0.717	0.745	0.758
EDCAH	**0.698**	**0.724**	**0.730**	**0.737**	0.758	**0.789**	0.797	**0.807**
EDCAH-t	0.688	0.721	0.729	0.735	**0.759**	0.787	**0.799**	0.807

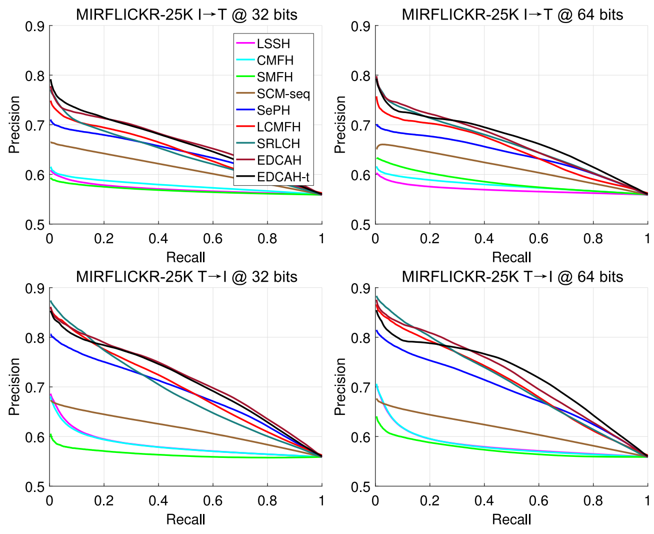

Fig. 4. PR curves of different methods @ 32 bits and 64 bits on MIRFLICKR-25K.

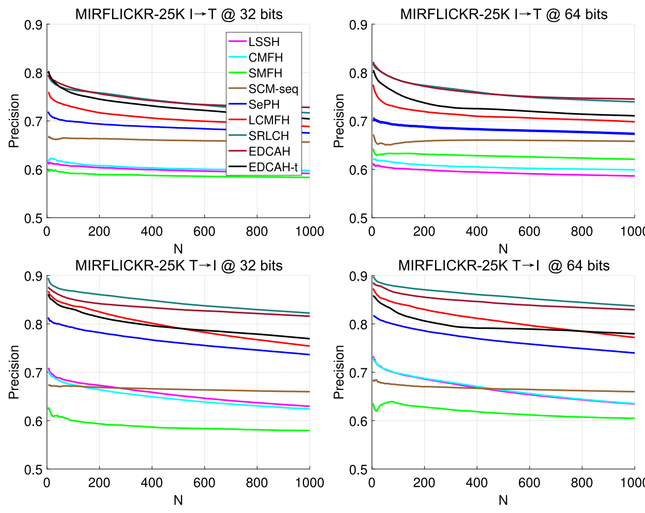

Fig. 5. topN curves of different methods @ 32 bits and 64 bits on MIRFLICKR-25K.

consume less training time reflected by Section 4.4. Furthermore, all the methods obtain a higher precision on MIRFLICKR-25K than they do on Wiki. The main reason is that the sample instances on MIRFLICKR-25K possess smaller semantic differences than do those of Wiki, which allows the models to obtain greater semantic correlation from the label supervision. Overall, these results

Table 5
The mAP scores of all comparison methods on NUS-WIDE.

Methods	I→T				T→I			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
LSSH (Zhou et al., 2014)	0.397	0.391	0.393	0.396	0.422	0.422	0.422	0.418
CMFH (Ding et al., 2014)	0.378	0.381	0.376	0.379	0.385	0.387	0.384	0.392
SMFH (Liu et al., 2016)	0.430	0.418	0.418	0.423	0.464	0.465	0.449	0.459
SCM-seq (Zhang & Li, 2014)	0.574	0.589	0.571	0.600	0.562	0.591	0.604	0.624
SePH (Xu et al., 2017)	0.548	0.550	0.563	0.563	0.637	0.648	0.667	0.670
LCMFH (Wang et al., 2019)	0.610	0.613	0.621	0.638	0.692	0.710	0.719	0.735
SRLCH (Shen et al., 2021)	0.610	0.625	**0.643**	0.649	0.721	**0.748**	0.762	0.777
EDCAH	**0.625**	**0.634**	0.640	**0.651**	**0.722**	0.743	**0.773**	0.776
EDCAH-t	0.624	0.631	0.638	0.649	0.722	0.743	0.772	**0.777**

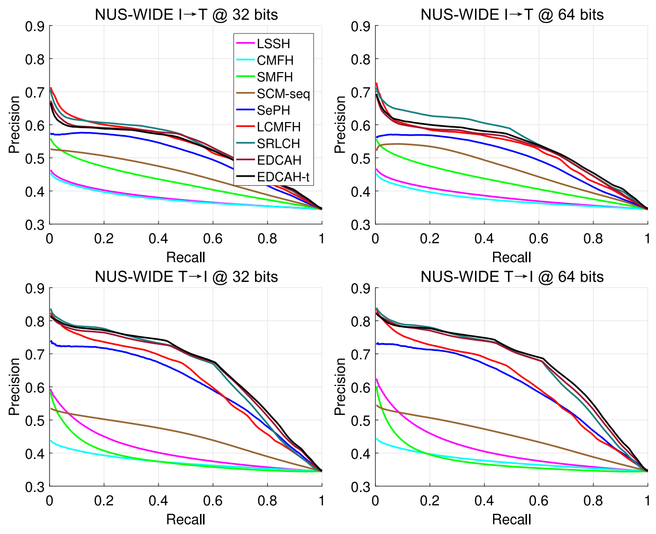

Fig. 6. PR curves of different methods @ 32 bits and 64 bits on NUS-WIDE.

show that the proposed methods outperform most baselines in terms of search accuracy except that they are slightly slower than SRLCH on topN metric.

Accuracy on NUS-WIDE: The mAP values of all comparison methods on NUS-WIDE are shown in Table 5. Figs. 6 and 7 plot the PR and topN curves, respectively. According to these experimental results, EDCAH obtains the best mAP and PR scores and they are similar to those of EDCAH-t on both search tasks, which are consistent with the results on MRIFLICKR-25K. The main reason may be that EDCAH produces effective hash codes or hash functions by utilizing the class alignment matrix factorization and discrete optimization. In addition, we observe that EDCAH and EDCAH-t yield comparable performance over SRLCH on mAP but are slightly slower topN scores than SRLCH. However, SRLCH requires a heavy training time and as the number of hash bits increases, the time increases exponentially. To summarize, we can conclude that EDCAH and its variant have superior search accuracy and are suitable for practical large-scale cross-modal image–text search applications.

Bootstrap estimates in search performance: To validate the performance reliability of the proposed EDCAH and EDCAH-t, we perform the bootstrap estimates experiment by following the settings in Bisani and Ney (2004) and Li (2019). Concretely, we adopt the bootstrap method with 99% confidence intervals (1000 sampling) accompanied by 100, 200 repeated experiment results, respectively. Table 6 reports the average mAP values of selected competitive LCMFH, SRLCH and our method with 32 and 64 code lengths fixed on Wiki and NUS-WIDE. The maximum 99% confidence interval (a.k.a. C.I.) of all map scores is $\pm 0.49212\%$. As shown in Table 6, we note that all the average mAP scores on two datasets agree on the level of 99%. And the obtained $\pm 0.49212\%$ possesses narrow confidence interval and considerable standard error, which demonstrate that all the data of Table 6 is trustworthy. The above observations further show the reliability of Tables 3–5.

4.3. Learning efficiency comparison

To verify the efficiency of EDCAH and EDCAH-t, we make comparisons with the baselines in terms of the training time and training size.

Training Time Comparison: Table 7 lists the training time of all comparison methods on MIRFLICKR-25K. It is clear that EDCAH-t requires the least amount of time compared to all the baselines. For example, EDCAH-t is approximately 6-times faster than LCMFH,

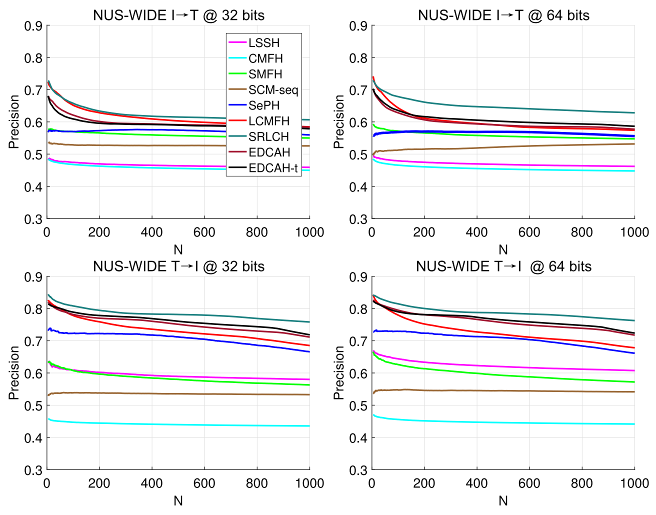

Fig. 7. topN curves of different methods @ 32 bits and 64 bits on NUS-WIDE.

Table 6
The mAP scores of selected methods with the bootstrap estimates on Wiki and NUS-WIDE. (99% C.I. = ±0.49212%).

Methods	Times	Wiki (I→T)		NUS-WIDE (I→T)		Wiki (T→I)		NUS-WIDE (T→I)	
		32 bits	64 bits	32 bits	64 bits	32 bits	64 bits	32 bits	64 bits
LCMFH (Wang et al., 2019)	100	0.330	0.350	0.615	0.622	0.689	0.703	0.710	0.717
	200	0.326	0.354	0.613	0.624	0.686	0.701	0.709	0.720
SRLCH (Shen et al., 2021)	100	0.342	0.362	0.627	0.640	0.729	0.732	0.743	0.765
	200	0.348	0.366	0.624	**0.644**	0.725	0.729	**0.748**	0.763
EDCAH	100	0.369	0.389	0.636	0.643	0.756	0.760	0.745	0.769
	200	0.372	0.385	**0.638**	0.637	0.753	0.758	0.743	**0.774**
EDCAH-t	100	**0.380**	**0.400**	0.632	0.641	0.755	**0.763**	0.747	0.770
	200	0.375	0.394	0.631	0.639	**0.759**	0.761	0.743	0.773

Table 7
Training time (seconds) comparison of all comparison methods on MIRFLICKR-25K.

Methods	MIRFLICKR-25K			
	16 bits	32 bits	64 bits	128 bits
LSSH (Zhou et al., 2014)	42.08	44.27	45.37	51.25
CMFH (Ding et al., 2014)	22.22	24.97	24.71	26.33
SMFH (Liu et al., 2016)	18.93	21.28	20.99	22.98
SCM-seq (Zhang & Li, 2014)	10.16	20.05	33.83	69.23
SePH (Xu et al., 2017)	106.66	198.03	372.42	708.91
LCMFH (Wang et al., 2019)	9.71	10.12	11.18	14.85
SRLCH (Shen et al., 2021)	114.60	115.34	119.09	125.43
EDCAH	5.93	6.64	8.78	11.96
EDCAH-t	**1.48**	**1.57**	**1.99**	**3.07**

100-times faster than SRLCH, and 4-times faster than EDCAH under four hash bits. The possible reason is that the two-step hashing scheme employs fewer matrix variables to reduce the computational complexity of the EDCAH model and lower the equipment load during training. Thus, EDCAH-t can further improve the learning efficiency of our method.

Table 8 shows the comparison results of the training time when the training size is varied from 2000 to 50,000 for 64 bits on NUS-WIDE. Thereinto, the training time of SRLCH under 2000 samples is not given because such a method adopts 5000 kernel anchors to train the model. Specifically, EDCAH-t achieves the shortest training time than all comparison methods. For example, it reduces the training time from 442.59 (CMFH), 27.89 (LCMFH), 234.87 (SRLCH), and 21.59 (EDCAH) to 9.47 at a training size of 50,000. Based on the above observations, EDCAH-t has the fastest training speed among all the baselines on the MIRFLICKR-25K and NUS-WIDE datasets, further demonstrating its superiority about learning efficiency.

Effect of Training Size: Table 9 shows the mAP scores for EDCAH and EDCAH-t under different training sizes (from 2000 to 180,000) at 64-bit hash codes. It reveals from Table 9 that EDCAH and EDCAH-t tend to converge rapidly when the training size is approximately 10,000. Thus, the use of 10,000 samples (that is, approximately 5% of the training set) on NUS-WIDE yields desirable

Table 8
The variation of training time (seconds) under different training set sizes @ 64 bits on NUS-WIDE.

Methods	Training size (K = 1000)				
	2K	5K	10K	20K	50K
LSSH (Zhou et al., 2014)	5.59	13.55	24.35	47.48	73.41
CMFH (Ding et al., 2014)	33.42	66.60	187.88	228.68	442.59
SMFH (Liu et al., 2016)	2.09	8.63	34.68	149.43	1122.47
SCM-seq (Zhang & Li, 2014)	17.92	16.79	21.06	24.33	25.20
SePH (Xu et al., 2017)	148.62	157.66	167.86	167.86	186.90
LCMFH (Wang et al., 2019)	1.10	1.94	4.32	7.33	27.89
SRLCH (Shen et al., 2021)	–	43.49	68.27	110.73	234.87
EDCAH	0.76	1.62	3.23	6.40	21.59
EDCAH-t	**0.69**	**1.23**	**1.25**	**3.99**	**9.47**

Table 9
The mAP scores of EDCAH and its variant @ 64 bits with varying training sizes on NUS-WIDE.

Task	Methods	Training size (K = 1000)								
		0.2K	0.5K	7.5K	10K	20K	50K	100K	150K	180K
I→T	EDCAH	0.600	0.614	0.635	0.640	0.645	0.649	0.650	0.652	0.656
	EDCAH-t	0.595	0.627	0.636	0.638	0.640	0.649	0.648	0.654	0.654
T→I	EDCAH	0.742	0.758	0.763	0.773	0.780	0.782	0.783	0.7850	0.787
	EDCAH-t	0.746	0.762	0.769	0.772	0.781	0.782	0.784	0.782	0.785

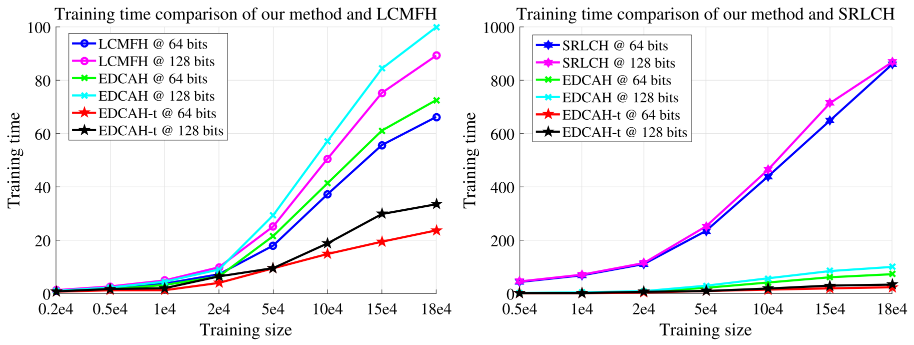

Fig. 8. The variation in training time with different training sizes @ 64 bits and 128 bits on NUS-WIDE.

results, and further increasing training set sizes does not substantially affect the search performance. However, as the training set size increases, the training time increases gradually as well.

To further validate the effect of training size on training time, Fig. 8 shows the training time comparison of EDCAH, EDCAH-t, and the two competitive methods LCMFH, SRLCH under different training sizes for both 64-bit and 128-bit on NUS-WIDE. We find that our method achieves similar results with LCMFH when the training size is less than 10,000; however, when the training size exceeds 20,000, the training time of EDCAH and LCMFH increases exponentially, while that of EDCAH-t increases linearly. Specifically, EDCAH-t is about 3.1-times faster than the other two methods with 180,000 training sizes regarding 64-bit and 128-bit. As for SRLCH, EDCAH-t still keeps a linear growth while SRLCH increases exponentially from 5000 to 180,000 samples. Fig. 8 and Tables 8 and 9 indicate that EDCAH-t has the fastest training speed among all the baselines on NUS-WIDE, highlighting its critical advantage in terms of learning efficiency, making it suitable for rapid large-scale cross-modal image–text search applications.

4.4. Empirical analysis

Ablation Experiment: To verify the efficacy of the EDCAH and EDCAH-t, we design two variants for comparison: EDCAH-C (EDCAH-t-C) and EDCAH-R (EDCAH-t-R). Specifically, EDCAH-C does not utilize the class alignment matrix factorization term (i.e., $\lambda_1 = 0$), and EDCAH-R adopts a continuous relaxation strategy to optimize the proposed algorithm (i.e., relaxing the discrete constraints for real-value variables when generating the hash codes). The mAP results of these variants are presented in Tables 10 and 11.

Compared to EDCAH-C, EDCAH achieves better mAP values. For example, the mAP scores of EDCAH are increased by 1.96% (Wiki), 3.45% (MIRFLICKR-25K), and 2.46% (NUS-WIDE) for the I→T task, and 2.27% (Wiki), 3.06% (MIRFLICKR-25K) and 2.39% (NUS-WIDE) for the T→I task. The main reason for these results is that the EDCAH combines matrix factorization and class label alignment to generate discriminative hash codes that contribute to preserving better correlations between modalities. Moreover, the mAP values of EDCAH-t are similar to those of EDCAH.

Table 10
The mAP scores of EDCAH, EDCAH-t and their variants on Wiki.

Methods	I→T				T→I			
	16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
EDCAH-C	0.337	0.350	0.366	0.376	0.715	0.724	0.738	0.751
EDCAH-t-C	0.333	0.352	0.364	0.375	0.709	0.728	0.730	0.752
EDCAH-R	0.335	0.346	0.366	0.371	0.713	0.725	0.740	0.753
EDCAH-t-R	0.330	0.344	0.367	0.384	0.713	0.727	0.737	0.757
EDCAH	**0.359**	0.371	0.387	0.390	**0.739**	0.753	0.759	0.770
EDCAH-t	0.355	**0.379**	**0.399**	**0.400**	0.732	**0.759**	**0.760**	**0.772**

Table 11
The mAP scores of EDCAH, EDCAH-t and their variants on MIRFLICKR-25K and NUS-WIDE.

Task	Methods	MIRFILCKR-25K				NUS-WIDE			
		16 bits	32 bits	64 bits	128 bits	16 bits	32 bits	64 bits	128 bits
I→T	EDCAH-C	0.663	0.681	0.693	0.713	0.605	0.604	0.616	0.627
	EDCAH-t-C	0.664	0.689	0.692	0.713	0.610	0.601	0.614	0.625
	EDCAH-R	0.675	0.683	0.701	0.714	0.602	0.614	0.623	0.632
	EDCAH-t-R	0.666	0.673	0.703	0.714	0.593	0.616	0.624	0.629
	EDCAH	**0.698**	**0.724**	**0.730**	**0.737**	**0.625**	**0.634**	**0.640**	**0.651**
	EDCAH-t	0.688	0.721	0.729	0.735	0.624	0.631	0.638	0.649
T→I	EDCAH-C	0.720	0.759	0.773	0.777	0.695	0.720	0.748	0.756
	EDCAH-t-C	0.717	0.753	0.772	0.775	0.700	0.723	0.749	0.756
	EDCAH-R	0.713	0.735	0.772	0.780	0.6983	0.726	0.747	0.759
	EDCAH-t-R	0.705	0.728	0.773	0.7844	0.691	0.725	0.749	0.757
	EDCAH	0.758	**0.789**	0.797	**0.807**	**0.722**	**0.743**	**0.773**	0.776
	EDCAH-t	**0.759**	0.787	**0.799**	0.807	0.722	0.743	0.772	**0.777**

Compared to EDCAH-R, EDCAH obtains higher mAP scores. Specifically, the mAP values of EDCAH can be up to 2.27% (Wiki), 2.88% (MIRFLICKR-25K), and 1.99% (NUS-WIDE) for the I→T task, and 2.20% (Wiki), 3.80% (MIRFLICKR-25K) and 2.11% (NUS-WIDE) for the T→I task. And the experimental observations for EDCAH-t are similar to those of EDCAH. Overall, these results demonstrate the efficacy of the proposed method and its variant.

Parameter Analysis: To analyze the impact of the parameters in search performance, we conduct experiments on three datasets w.r.t. 64 bits. The results with 64 hash bits are similar to those at other hash bits (a.k.a. 16-bit, 32-bit, and 128-bit). Due to space limitations, we only show the EDCAH results at 64 bits, as plotted in Fig. 9. Concretely, λ_1 controls the influence of class alignment matrix factorization, λ_2 impacts the hash code learning procedure, λ_3 influences the contribution of association matching, λ_4 controls the hash function learning process, and λ_5 is a regularization term that controls model convergence. Here, we set $\lambda_1 = 0.5$ because the impact of input image data is equivalent to that of text data for our method (Chen et al., 2020).

Fig. 9 indicates that for Wiki, the parameter settings of $\lambda_2 \in [1, 100]$, $\lambda_3 \in [1, 10]$, $\lambda_4 \in [0.01, 1]$ and $\lambda_5 \in \left[10^{-6}, 1\right]$ apparently do not affect the search performance; for MIRFLICKR-25K and NUS-WIDE, the insensitive regions for the parameter settings are $\lambda_2 \in [0.1, 100]$, $\lambda_3 \in [1, 10]$, $\lambda_4 \in \left[10^{-3}, 10^{-1}\right]$, and $\lambda_5 \in \left[10^{-6}, 10^{-1}\right]$. The parameter setting results for EDCAH-t are consistent with those of EDCAH. These results show that the proposed EDCAH and its variant EDCAH-t maintain stable search performance when these parameter values in these ranges are employed.

Convergence Analysis: To validate that the designed optimization strategy can converge, we further carry out a convergence study of our method on the three datasets w.r.t. 64 bits, as plotted in Fig. 10. Following the setting in Luo et al. (2018), the objective values are normalized by dividing the maximum on each dataset. It is observed that: (1) The proposed method and its variant show a good tendency to converge; EDCAH converges in approximately 25 iterations and EDCAH-t about 20 iterations. Thus, these results verify the convergence abilities of the EDCAH and EDCAH-t models. (2) Besides, EDCAH-t converges faster than EDCAH on all three datasets, indicating the superiority of the two-step hashing scheme in improving the learning efficiency of our method.

4.5. Theoretical and practical implications

In summary, the implications of this work are mainly manifested in two aspects.

Theoretically, this work is a fundamental but hot research top for exploring the correlation between distinct modalities. And enabling the bi-directional search of images and texts is rather significant to understand the correspondences between vision and language in the multimedia and computer vision communities. Meanwhile, for our method, the extension of multiple modalities is simple and easy by constructing the proposed objective function, which is a very interesting topic worthy of further exploration and research.

Practically, performing fast and accurate searches is a challenging problem while faced with limited storage memory and search capability. For this reason, our study fully captures the similar semantics of data instances and supervised information, which further enhances the search performance and promises the accuracy of hashing methods. Besides, our approach is not only effective, but

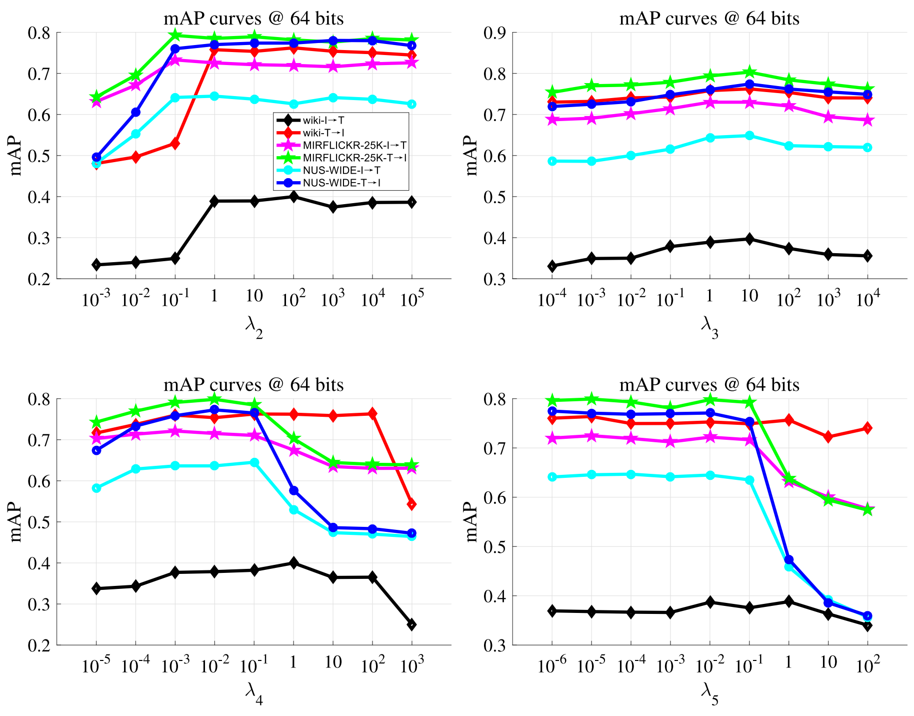

Fig. 9. Parameter analysis curves of EDCAH on three benchmark datasets.

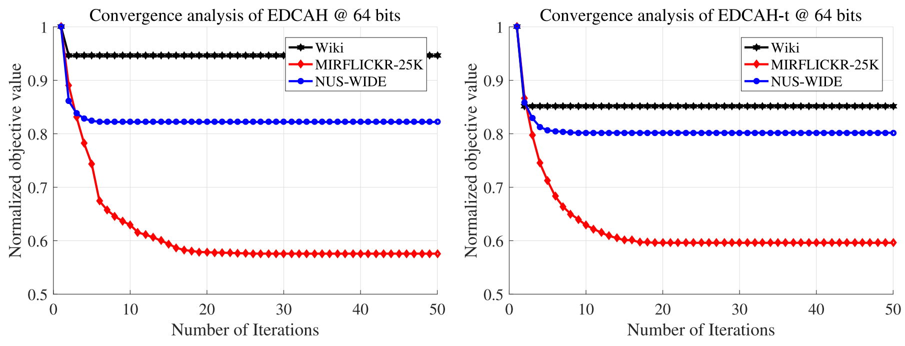

Fig. 10. Convergence analysis of EDCAH and its variant @ 64 bits on all datasets.

also has made progress in learning efficiency. Specifically, the study is a scalable hashing method and reduces the training time, making it feasible and practical for large-scale cross-modal image–text datasets.

When training hashing model, we adopt the hand-crafted shallow features rather than the deep features for search tasks. The main explanation is that deep hashing methods benefit model performance at the expense of heavy computational cost and substantial hyperparameters adjustments. Therefore, it is worth noting that our approach displays a tradeoff between search accuracy and training speed.

5. Conclusion

This paper presents a novel discrete supervised hashing method (EDCAH). Our method can generate discrete discriminative hash codes for cross-modal image–text search because of two main characteristics: (1) it designs the hash code learning by integrating matrix factorization and class alignment; (2) it adopts discrete optimization strategies to obtain closed-form solutions of the hash codes during training. (3) to reduce the computational complexity of EDCAH, a fast and efficient variant EDCAH-t is further proposed to boost and optimize the learning efficiency of the EDCAH model that adopts a two-step hashing scheme. Extensive experimental results on three widely used benchmark datasets demonstrate that EDCAH and EDCAH-t outperform the state-of-the-art hashing methods in both search accuracy and learning efficiency. Besides, EDCAH-t possesses better learning efficiency of hashing model than all the baselines on large-scale datasets, making it suitable for rapid yet practical cross-modal image–text search applications in precision-first and high-speed situations. In the future, we consider to extend our method by integrating embedded learning to generate more compact, precise hash codes to improve the search performance.

CRediT authorship contribution statement

Song Wang: Methodology, Software, Data curation, Formal analysis, Validation, Writing – original draft, Writing – review & editing. **Huan Zhao:** Funding acquisition, Writing – original draft, Validation, Supervision, Writing – review & editing. **Yunbo Wang:** Conceptualization, Resources, Data curation, Investigation, Writing – review & editing. **Jing Huang:** Resources, Data curation, Investigation, Writing – review & editing. **Keqin Li:** Resources, Investigation, Writing – review & editing.

Acknowledgments

This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1713400, and in part by the National Natural Science Foundation of China under Grant 61772188, Special Project of Foshan Science and Technology Innovation Team, China under Grant FS0AA-KJ919-4402-0069.

References

Bisani, M., & Ney, H. (2004). Bootstrap estimates for confidence intervals in ASR performance evaluation. In *Proc. icassp* (pp. 409–412).

Chen, Z., Li, C., Luo, X., Nie, L., Zhang, W., & Xu, X. (2020). SCRATCH: A scalable discrete matrix factorization hashing framework for cross-modal retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, *30*(7), 2262–2275.

Chen, S., Shen, F., Yang, Y., Xu, X., & Song, J. (2017). Supervised hashing with adaptive discrete optimization for multimedia retrieval. *Neurocomputing*, *253*, 97–103.

Chua, T.-S., Tang, J., Hong, R., Li, H., Luo, Z., & Zheng, Y. (2009). NUS-WIDE: a real-world web image database from National University of Singapore. In *Proc. civr* (pp. 48–56).

Ding, G., Guo, Y., & Zhou, J. (2014). Collective matrix factorization hashing for multimodal data. In *Proc. cvpr* (pp. 2083–2090).

Ding, Y., Wong, W. K., Lai, Z., & Zhang, Z. (2020). Discriminative dual-stream deep hashing for large-scale image retrieval. *Information Processing Management*, *57*(6), Article 102288.

Gong, Y., Lazebnik, S., Gordo, A., & Perronnin, F. (2013). Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(12), 2916–2929.

Gui, J., Liu, T., Sun, Z., Tao, D., & Tan, T. (2017). Fast supervised discrete hashing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(2), 490–496.

Han, J., Zhang, Z., Ren, Z., & Schuller, B. (2019). Implicit fusion by joint audiovisual training for emotion recognition in mono modality. In *Proc. icassp.* (pp. 5861–5865).

He, S., & Zhao, H. (2017). Automatic syllable segmentation algorithm of Chinese speech based on MF-DFA. *Speech Communications*, *92*(9), 42–51.

Huang, C., Luo, X., Zhang, J., Liao, Q., Wang, X., Jiang, Z., et al. (2020). Explore instance similarity: An instance correlation based hashing method for multi-label cross-model retrieval. *Information Processing Management*, *57*(2), Article 102165.

Huiskes, M. J., & Lew, M. S. (2008). The MIR flickr retrieval evaluation. In *Proc. mir* (pp. 39–43).

Ji, R., Liu, H., Cao, L., Liu, D., Wu, Y., & Huang, F. (2017). Toward optimal manifold hashing via discrete locally linear embedding. *IEEE Transactions on Image Processing*, *26*(11), 5411–5420.

Jiang, Q. Y., & Li, W. J. (2017). Deep cross-modal hashing. In *Proc. CVPR.* (pp. 3270–3278).

Jiang, Q.-Y., & Li, W.-J. (2019). Discrete latent factor model for cross-modal hashing. *IEEE Transactions on Image Processing*, *28*(7), 3490–3501.

Kumar, S., & Udupa, R. (2011). Learning hash functions for cross-view similarity search. In *Proc. ijcai* (pp. 1360–1365).

Li, K. (2017). Optimal task dispatching on multiple heterogeneous multiserver systems with dynamic speed and power management. *IEEE Trans. Sustain. Comput.*, *2*(2), 167–182.

Li, K. (2019). Optimal task execution speed setting and lower bound for delay and energy minimization. *Journal of Parallel and Distributed Computing*, *123*, 13–25.

Li, C., Deng, C., Li, N., Liu, W., Gao, X., & Tao, D. (2018). Self-supervised adversarial hashing networks for cross-modal retrieval. In *Proc. CVPR.* (pp. 4242–4251).

Li, X., Gao, L., Xu, X., Shao, J., Shen, F., & Song, J. (2017). Kernel based latent semantic sparse hashing for large-scale retrieval from heterogeneous data sources. *Neurocomputing*, *253*, 89–96.

Li, C., Yan, T., Luo, X., Nie, L., & Xu, X. (2019). Supervised robust discrete multimodal hashing for cross-media retrieval. *IEEE Transactions on Multimedia*, *21*(11), 2863–2877.

Li, W.-H., Yang, S., Wang, Y., Song, D., & Li, X.-Y. (2021). Multi-level similarity learning for image-text retrieval. *Information Processing Management*, *58*(1), Article 102432.

Liang, J., He, R., Sun, Z., & Tan, T. (2019). Aggregating randomized clustering-promoting invariant projections for domain adaptation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(5), 1027–1042.

Lin, Z., Ding, G., Han, J., & Wang, J. (2016). Cross-view retrieval via probability-based semantics-preserving hashing. *IEEE Transactions on Cybernetics*, *47*(12), 4342–4355.

Liu, H., Ji, R., Wu, Y., & Hua, G. (2016). Supervised matrix factorization for cross-modality hashing. In *Proc. ijcai* (pp. 1767–1773).

Liu, H., Ji, R., Wu, Y., Huang, F., & Zhang, B. (2017). Cross-modality binary code learning via fusion similarity hashing. In *Proc. cvpr* (pp. 7380–7388).

Long, M., Cao, Y., Wang, J., & Yu, P. S. (2016). Composite correlation quantization for efficient multimodal retrieval. In *Proc. sigir* (pp. 579–588).

Lu, X., Zhu, L., Cheng, Z., Song, X., & Zhang, H. (2019). Efficient discrete latent semantic hashing for scalable cross-modal retrieval. *Signal Processing*, *154*, 217–231.

Luo, X., Nie, L., He, X., Wu, Y., Chen, Z.-D., et al. (2018). Fast scalable supervised hashing. In *Proc. sigir* (pp. 735–744).

Peng, Y., Huang, X., & Zhao, Y. (2018). An overview of cross-media retrieval: Concepts, methodologies, benchmarks, and challenges. *IEEE Transactions on Circuits and Systems for Video Technology*, *28*(9), 2372–2385.

Pereira, J. C., Coviello, E., Doyle, G., Rasiwasia, N., Lanckriet, G. R., Levy, R., et al. (2013). On the role of correlation and abstraction in cross-modal multimedia retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *36*(3), 521–535.

Shen, H. T., Liu, L., Yang, Y., Xu, X., Huang, Z., Shen, F., et al. (2021). Exploiting subspace relation in semantic labels for cross-modal hashing. *IEEE Transactions on Knowledge and Data Engineering*, *33*(10), 3351–3365.

Shen, F., Xu, Y., Liu, L., Yang, Y., Huang, Z., & Shen, H. T. (2018). Unsupervised deep hashing with similarity-adaptive and discrete optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(12), 3034–3044.

Shen, F., Zhou, X., Yang, Y., Song, J., Shen, H. T., & Tao, D. (2016). A fast optimization method for general binary code learning. *IEEE Transactions on Image Processing*, *25*(12), 5610–5621.

Song, J., Yang, Y., Yang, Y., Huang, Z., & Shen, H. T. (2013). Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *Proc. sigmod* (pp. 785–796).

Su, S., Zhong, Z., & Zhang, C. (2019). Deep joint-semantics reconstructing hashing for large-scale unsupervised cross-modal retrieval. In *Proc. ICCV.* (pp. 3027–3035).

Tang, J., Wang, K., & Shao, L. (2016). Supervised matrix factorization hashing for cross-modal retrieval. *IEEE Transactions on Image Processing*, *25*(7), 3157–3166.

Wang, D., Gao, X.-B., Wang, X., & He, L. (2019). Label consistent matrix factorization hashing for large-scale cross-modal similarity search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(10), 2466–2479.

Wang, D., Gao, X., Wang, X., He, L., & Yuan, B. (2016). Multimodal discriminative binary embedding for large-scale cross-modal retrieval. *IEEE Transactions on Image Processing*, *25*(10), 4540–4554.

Wang, Y., Liang, J., Cao, D., & Sun, Z. (2019). Local semantic-aware deep hashing with hamming-isometric quantization. *IEEE Transactions on Image Processing*, *28*(6), 2665–2679.

Wang, Y., Luo, X., Nie, L., Song, J., Zhang, W., & Xu, X. (2021). BATCH: A scalable asymmetric discrete cross-modal hashing. *IEEE Transactions on Knowledge and Data Engineering*, *33*(11), 3507–3519.

Wang, Y., Ou, X., Liang, J., & Sun, Z. (2021). Deep semantic reconstruction hashing for similarity retrieval. *IEEE Transactions on Circuits and Systems for Video Technology*, *31*(1), 387–400.

Wang, W., Shen, Y., Zhang, H., & Liu, L. (2020). Semantic-rebased cross-modal hashing for scalable unsupervised text-visual retrieval. *Information Processing & Management*, *57*(6), Article 102374.

Wang, L., Sun, W., Zhao, Z., & Su, F. (2017). Modeling intra- and inter-pair correlation via heterogeneous high-order preserving for cross-modal retrieval. *Signal Processing*, *131*, 249–260.

Wang, J., Zhang, T., Song, J., Sebe, N., & Shen, H. T. (2018). A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *40*(4), 769–790.

Weiss, Y., Torralba, A., & Fergus, R. (2009). Spectral hashing. In *Proc. neurips* (pp. 1753–1760).

Wu, G., Han, J., Lin, Z., Ding, G., Zhang, B., & Ni, Q. (2019). Joint image-text hashing for fast large-scale cross-media retrieval using self-supervised deep learning. *IEEE Trans. Ind. Electron.*, *66*(12), 9868–9877.

Wu, Y., Luo, X., Xu, X.-S., Guo, S., & Shi, Y. (2018). Dictionary learning based supervised discrete hashing for cross-media retrieval. In *Proc. icmr* (pp. 222–230).

Xu, X., Shen, F., Yang, Y., Shen, H. T., & Li, X. (2017). Learning discriminative binary codes for large-scale cross-modal retrieval. *IEEE Transactions on Image Processing*, *26*(5), 2494–2507.

Yang, E., Deng, C., Liu, W., Liu, X., Tao, D., & Gao, X. (2017). Pairwise relationship guided deep hashing for cross-modal retrieval. In *Proc. AAAI.* (pp. 1618–1625).

Yao, T., Han, Y., Wang, R., Kong, X., Yan, L., Fu, H., et al. (2020). Efficient discrete supervised hashing for large-scale cross-modal retrieval. *Neurocomputing*, *385*, 358–367.

Ye, Z., & Peng, Y. (2018). Multi-scale correlation for sequential cross-modal hashing learning. In *Proc. mm* (pp. 852–860).

Zeng, H., Zhang, H., & Zhu, L. (2020). Label consistent locally linear embedding based cross-modal hashing. *Information Processing Management*, *57*(6), Article 102136.

Zhang, D., & Li, W.-J. (2014). Large-scale supervised multimodal hashing with semantic correlation maximization. In *Proc. AAAI.* (pp. 2177–2183).

Zhang, S., Li, J., Jiang, M., Yuan, P., & Zhang, B. (2018). Scalable discrete supervised multimedia hash learning with clustering. *IEEE Transactions on Circuits and Systems for Video Technology*, *28*(10), 2716–2729.

Zhang, J., & Peng, Y. (2018). Query-adaptive image retrieval by deep-weighted hashing. *IEEE Transactions on Multimedia*, *20*(9), 2400–2414.

Zhang, J., & Peng, Y. (2020). Multi-pathway generative adversarial hashing for unsupervised cross-modal retrieval. *IEEE Transactions on Multimedia*, *22*(1), 174–187.

Zhao, H., Cao, J., Xu, M., & Lu, J. (2020). Variational neural decoder for abstractive text summarization. *Computer Science and Information Systems*, *17*(2), 537–552.

Zhao, H., Wang, S., She, X., & Su, C. (2020). Supervised matrix factorization hashing with quantitative loss for image-text search. *IEEE Access*, *8*, 102051–102064.

Zheng, C., Zhu, L., Lu, X., Li, J., Cheng, Z., & Zhang, H. (2020). Fast discrete collaborative multi-modal hashing for large-scale multimedia retrieval. *IEEE Transactions on Knowledge and Data Engineering*, *32*(11), 2171–2184.

Zheng, C., Zhu, L., Lu, X., Li, J., Cheng, Z., & Zhang, H. (2020). Fast discrete collaborative multi-modal hashing for large-scale multimedia retrieval. *IEEE Transactions on Knowledge and Data Engineering*, *32*(11), 2171–2184.

Zhou, J., Ding, G., & Guo, Y. (2014). Latent semantic sparse hashing for cross-modal similarity search. In *Proc. sigir.* (pp. 415–424).

Zhu, X., Huang, Z., Shen, H. T., & Zhao, X. (2013). Linear cross-modal hashing for efficient multimedia search. In *Proc. mm*, (pp. 143–152).