Information Sciences 581 (2021) 136–154

Contents lists available at ScienceDirect

Information Sciences

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

Noise-robust Deep Cross-Modal Hashing

Runmin Wang [a,b], Guoxian Yu [a,b,*], Hong Zhang [a], Maozu Guo [d], Lizhen Cui [b], Xiangliang Zhang [c]

[a] College of Computer and Information Sciences, Southwest University, Chongqing, China
[b] School of Software, Shandong University, Jinan, China
[c] CEMSE, King Abdullah University of Science and Technology, Thuwal, Saudi Arabia
[d] College of Electrical and Information Engineering, Beijing University of Civil Engineering and Architecture, Beijing, China

ARTICLE INFO

Article history:
Received 21 July 2020
Received in revised form 9 August 2021
Accepted 11 September 2021
Available online 14 September 2021

Keywords:
Cross-modal hashing
Noise labels
Deep learning
Feature similarity
Label similarity

ABSTRACT

Cross-modal hashing has been intensively studied to efficiently retrieve multi-modal data across modalities. Supervised cross-modal hashing methods leverage the labels of training data to improve the retrieval performance. However, most of these methods still assume that the semantic labels of training data are ideally complete and noise-free. This assumption is too optimistic for real multi-modal data, whose label annotations are, in essence, error-prone. To achieve effective cross-modal hashing on multi-modal data with noisy labels, we introduce an end-to-end solution called Noise-robust Deep Cross-modal Hashing (NrDCMH). NrDCMH contains two main components: a noise instance detection module and a hash code learning module. In the noise detection module, NrDCMH firstly detects noisy training instance pairs based on the margin between the label similarity and feature similarity, and specifies weights to pairs using the margin. In the hash learning module, NrDCMH incorporates the weights into a likelihood loss function to reduce the impact of instances with noisy labels and to learn compatible deep features by applying different neural networks on multi-modality data in a unified end-to-end framework. Experimental results on multi-modal benchmark datasets demonstrate that NrDCMH performs significantly better than competitive methods with noisy label annotations. NrDCMH also achieves competitive results in 'noise-free' scenarios.

1. Introduction

With the rapid expansion of the Internet in modern life and an ever-increasing range of social networks available, tremendous multimedia data are generated every day. Thus, approximate nearest neighbor (ANN) search becomes increasingly important for many applications. Hashing maps high-dimensional data into low-dimensional binary codes, which can then be efficiently used for an ANN search. By converting all kinds of multimedia data into fixed-length binary codes, hashing methods can significantly reduce the storage cost and sharply increase the search speed. Based on these merits, hashing methods are attracting increased attention [2,47,23,6,38,32].

In recent years, the amount and variety of multimedia data (i.e., image, video, text, and audio) are increasing rapidly. In many scenarios, we need to search for data across different modalities. For example, we may want to find images that have a

* Corresponding author at: College of Computer and Information Sciences, Southwest University, Chongqing, China.
E-mail addresses: rmwang@email.swu.edu.cn (R. Wang), gxyu@sdu.edu.cn (G. Yu), zhangh@swu.edu.cn (H. Zhang), guomaozu@bucea.edu.cn (M. Guo), clz@sdu.edu.cn (L. Cui), xiangliang.zhang@kaust.edu.sa (

https://doi.org/10.1016/j.ins.2021.09.030
0020-0255/

similar semantic meaning using the query text and vice versa. However, traditional methods for single-modal retrieval cannot work under these cross-modal scenarios. To address this problem, single-modal hashing methods have been extended to multi-modal scenarios and cross-modal hashing methods have been proposed. In terms of using supervision information (i.e., class labels) or not, hashing methods can be divided into two categories. Unsupervised cross-modal hashing methods mainly aim to map heterogeneous data into a common Hamming space to maximize the correlation [8,50]. In contrast, supervised methods leverage the semantic label information of data and generally achieve a better performance than unsupervised ones [22,25,26].

Most of these cross-modal hashing (CMH) methods aim to find a common Hamming space and project data from different modalities toward this shared space to generate binary codes for search. Almost all these shallow methods take hand-crafted features (*e.g.*, SIFT [27] and bag-of-words vector) as the input. An underlying issue of these CMH methods with shallow architectures is that the feature extraction procedure and learning procedure are isolated. Furthermore, the performance is severely restricted by the hand-crafted feature quality. To overcome this problem, several studies have investigated CMH methods based on deep neural networks (DNN) [3,19]. These CMH methods with deep architecture implement feature learning and hash code learning simultaneously and realize the whole framework in an end-to-end architecture. Due to the consistency and compatibility between the feature learning procedure and hash code learning procedure, these end-to-end methods usually achieve a better performance than shallow counterparts. However, supervised methods based on DNN typically require adequate precisely-labeled training data to guarantee the performance, which is difficult to acquire in practice. Although there are some wholly labeled small datasets, manually label large-scale cross-modal data is still time-consuming and expensive. Hence, semi-supervised CMH methods that can take advantage of both labeled and unlabeled data have also been invented [47,23,25,44].

To the authors' knowledge, almost all existing CMH methods divide multi-modal instances into two categories: instances without labels and instances with entirely credible labels. However, in the real world, there are many instances with non-credible labels. For example, users on Facebook and Instagram often submit photos with short texts and some tags. These photos and texts are a natural bed of multi-modal instances. However, users often mislabel or ignore some labels when adding tags, and, as a result, these multi-modal data contain label noise. Thus, there is abundant data with label noise. Nevertheless, existing CMH methods neglect this practical issue.

Here we propose an end-to-end deep noise-robust cross-modal hashing (NrDCMH) method for multi-modal data with noisy labels. NrDCMH firstly evaluates the gap between feature similarity induced from different data modalities and label similarity induced from the shared label space. The larger the gap, the larger the probability that noisy labels contaminate at least one instance of the pair, and the smaller the weight we will assign to this pair. Then, NrDCMH incorporates the weight into a likelihood loss function to reduce the impact of instances with noisy labels and to extract deep features from multi-modality data using different deep networks. In addition, the hashing quantification loss is also integrated with the likelihood loss function to learn compatible features and hash functions in a coherent fashion. The main contributions of this paper are as follows:

- We focus on cross-modal hashing with label noise, which is a practical and important but largely overlooked topic in supervised single/multi-modal hashing.
- NrDCMH computes the margin between the feature similarity and label similarity of instance pairs and prioritizes the training instance pairs with a large gap as the label corrupted instances. NrDCMH integrates the gap with a likelihood loss function to reduce the impact of noisy labels, and learn compatible deep features from multi-modality and hashing codes in a unified objective.
- Our experiments on benchmark datasets show that NrDCMH outperforms other related and recent competitive cross-modal hashing methods [46,8,25,19] in cross-modal data retrieval with corrupted labels, and performs comparatively well with 'noise-free' data.

The rest of this paper is organized as follows. We briefly introduce the related and representative works in Section 2. Section 3 elaborates on the proposed method. Experimental results and analysis are provided in Section 4. The conclusions and future work are given in Section 5.

2. Related works

Generally, our work is closely related with cross-modal hashing and deep learning with label noise.

2.1. Cross-modal hashing

Cross-modal hashing methods have been widely studied in recent years for their efficacy on multi-modal data. In this paper, we divide CMH methods into three categories: unsupervised, supervised, and semi-supervised.

Unsupervised CMH methods adopt a similar approach to canonical correlation analysis (CCA) [14], which finds a common Hamming space to maximize the correlation among different modalities. Cross view hashing (CVH) extends single-modal spectral hashing (SH) [39] to a cross-modal scenario. Collective matrix factorization hashing (CMFH) [8] and latent semantic sparse hashing (LSSH) [50] share a similar technique that finds a common latent semantic subspace for different modalities

via matrix factorization and then converts features in the latent subspace into hash codes. Semantic topic multi-modal hashing (STMH) [37] models text with semantic topics and images with semantic concepts, and then projects these two types of semantic features into a common subspace. Finally, STMH obtains the hash code by determining whether a topic or concept is contained in a text or an image. In recent years, deep neural networks have emerged as a dominant machine learning tool for a wide variety of application domains. Several unsupervised deep CMH methods have been proposed recently. Unsupervised deep cross-modal hashing (UDCMH) [40] combines deep feature learning networks and matrix factorization to directly learn the discrete hash codes in a binary latent representation space. Deep joint-semantic reconstructing hashing (DJSRH) [33] constructs a novel joint-semantic affinity matrix and then reconstructs the affinity matrix to generate the binary code. Generally, unsupervised CMH methods focus only on the correlation among different modalities, they typically ignore the rich semantic information embedded in class labels.

Supervised CMH methods can leverage semantic information and often achieve a better performance than their unsupervised counterparts. To name a few, cross-modality similarity sensitive hashing (CMSSH) [2] regards every bit of hash code as a classification task and learns the whole hash code bits one by one. Multi-modal latent binary embedding (MLBE) [49] establishes a probabilistic graphical model and treats the binary latent factors as hash codes. Semantic correlation maximization (SCM) [46] optimizes the hashing functions by maximizing the correlation between two modalities with respect to the semantic similarity obtained from the labels. Interestingly, SCM degenerates to CCA if the semantic similarity matrix is equal to an identity matrix $\mathbf{I}_n$, just like CVH under the unsupervised setting. Semantics preserving hashing (SePH) [22] transforms the similarity obtained from semantic labels into a probability distribution and approximates it with to-be-learned hash codes in the Hamming space, and then learns the hash codes via kernel logistic regression. Fast discrete cross-modal hashing (FDCH) [24] learns two modality-specific projections with drifts; it projects features of different modalities to the common hash codes and learns a projection from the common hash codes to class labels to preserve the similarity in the labels. Deep supervised CMH methods have also spurred much research interest in recent years. Deep cross-modal hashing (DCMH) [19] uses features learned by convolutional neural networks rather than hand-crafted features. DCMH also integrates both a feature learning procedure and hash code learning procedure into the same deep learning framework. Pairwise relationship guided deep hashing (PRDH) [42] uses both a pairwise intra-modal and inter-modal similarity obtained from semantic labels to learn the hash code. Semantic deep cross-modal hashing (SDCH) [41] integrates deep features learned from different modalities and adds a fully connected layer to learn the unified hash code from these integrated deep features.

Semi-supervised CMH methods can leverage a small set of labeled data and abundant unlabeled data to bypass the cost of annotating a huge amount of multi-modal training data. Weakly-supervised cross-modal hashing (WCHash) [26] first learns a latent central modality and then uses a weak-label learning method to enrich the labels and complete the similarity matrix. WCHash obtains unified hash codes for each instance from the latent central modality under the principle of CCA. Semi-supervised cross-modal hashing by generative adversarial network (SCH-GAN) [47] focuses on the scenario when the corresponding instances in other modalities are missing or imperfect. SCH-GAN utilizes the generation ability of Generative Adversarial Network (GAN) [12] to find the most similar instances in other modalities (maybe not the perfect counterparts) and then uses the instance pairs to train the model. Ranking-based deep cross-modal hashing (RDCMH) [25] first derives a similarity ranking list for every instance, and then optimizes the feature learning model and hash learning model simultaneously with the ranking loss. Attribute-guided network (AgNet) [17] extends zero-shot hashing to the multi-modal setting, which can encode instances of unseen categories. Cross-modal zero-shot hashing (CZHash) [23] adopts a similar idea with AgNet, whereby it can not only handle data modalities with different labels, but also use both labeled and unlabeled data.

Generally speaking, unsupervised CMH methods neglect class labels that contain much semantic information. In contrast, most supervised CMH methods completely trust the collected class labels, and semi-supervised methods only consider the incomplete or insufficient labels. They all ignore the fact that the obtained labels maybe corrupted by noise. To enable CMH in a more practical setting, we propose NrDCMH, which explicitly models the label noises of training data for effective CMH.

2.2. Learning with noisy labels

The potential performance of deep neural networks is well known to be dependent on a large amount of precisely-labeled training data. However, more often, we have a large amount of data that may not be precisely labeled. In other words, the data have noisy labels. Existing deep solutions for learning from data with noisy labels can be roughly divided into four directions, one of which aims to estimate the noise distribution (a transition matrix $\mathbf{T}$ in most cases). Sukhbaatar et al. [34] proposed several simple approaches to learn the transition matrix, where the main idea is to add an extra linear layer as a transition layer to learn the noise distribution. Goldberger et al. [11] treated the true labels as latent variables and added an extra linear layer at the end of the backbone network. They then adopted the EM algorithm to learn parameters of backbone network and noise distribution alternately. Patrini et al. [29] proposed two procedures for loss correction with a transition matrix $\mathbf{T}$, which is obtained by prior knowledge or estimated based on noisy class probability. In contrast, Hendrycks et al. [15] estimated the transition matrix $\mathbf{T}$ via stacking the mean predictions.

The second direction aims to estimate the true labels of corrupted instances and learn from the estimated labels. Veit et al. [36] integrated a label cleaning network and a classifier network. The label cleaning network supervised by the verified labels (true labels that are known from the beginning) is used to predict the true labels of corrupted instances. The cleaned labels (labels estimated by label cleaning network) and verified labels are augmented to train the classifier network. Probabilistic noise correction for learning with noisy labels (PENCIL)[43] proposes an end-to-end framework within which the

corrected label (estimated true label) is updated via back-propagation at every iteration, similarly to the other parameters of DNN. Li et al. [21] designed a distillation network in which an auxiliary model trained with a small clean dataset is used to hedge the risk introduced by noisy labels.

Noise-robust loss function design is another paradigm that aims to find a category of loss function inherently robust to label noise. Ghosh et al. [10] proved a sufficient condition on a loss function under which the model is tolerant to uniform label noise, and also proved that the 0–1 loss, Sigmoid loss, ramp loss, and probit loss satisfy this condition. In addition, Ghosh et al. [9] proved that the loss function based on the mean absolute value of error is inherently robust to label noise in DNN. Zhang et al. [48] presented a set of noise-robust loss functions that are generalizations of mean absolute error (MAE) and categorical cross-entropy (CCE). These loss functions perform well in a wide range of noisy label scenarios.

The fourth approach to deal with label noise selects reliable and clean instances to train the primary model, which can also be implemented by assigning a weight to the instance according to its credibility that it is clean. Co-teaching [13] trains two networks separately, selects samples with small losses, and communicates with each other. During this procedure, instances with high credibility are emphasized and the risk introduced by instances with noisy labels is hedged. MentorNet [18] introduces an extra network to learn the weights of samples and select samples for training according to the weights. Ren et al. [31] proposed a meta-learning algorithm to assign weights to training instances based on their gradient directions.

Our method presented in this work belongs to the fourth category, where we try to detect noisy instances by comparing the label similarity and feature similarity of instance pairs based on the proposition that instances with similar labels should have high feature similarity and vice versa. Under this proposition, we estimate the label corruption probability for each instance pair based on the gap between label similarity and feature similarity, and then prioritize instances with a low corrupted probability for training the deep CMH model.

3. Proposed method

3.1. Problem definition

In this paper, we use boldface uppercase letters (like $\mathbf{W}$) to represent matrices, and boldface lowercase letters (like $\mathbf{v}, \mathbf{w}$) to represent vectors. The same letters represent the same matrix without special instructions. For example, s_{ij} represents the element in the i-th row and j-th column of matrix $\mathbf{S}$, $\mathbf{W}_{*j}$ represents the j-th column of $\mathbf{W}$, and $\mathbf{W}_{i*}$ represents the i-th row of $\mathbf{W}$. $\mathbf{W}^T$ is the transpose of $\mathbf{W}$. $\mathbf{1}$ is a vector with all elements being 1. We use $tr(\cdot)$ and $\|\cdot\|_F$ to denote the trace and Frobenius norm of a matrix, and $sign(\cdot)$ is the sign function. The main notations used in this paper are listed in Table 1.

Without loss of generality, we present our solution with two modalities (text and image) in this paper. Note that NrDCMH can be easily adapted to cases with more than two modalities. Assuming that $\mathcal{O} = \{o_i\}_{i=1}^n$ is the training set with n instances. Each instance $o_i = \{\mathbf{x}_i, \mathbf{y}_i\}$ contains two modalities (X and Y). For each o_i, we have L labels $\mathbf{l}_i \in \{0,1\}^L$ with a certain probability of being corrupted, and each one of these 1's in vector $\mathbf{l}_i$ means o_i belongs to this category, and each 0 means it doesn't. Label noise makes some zeros in the ground-truth label vector shift to ones. $\mathbf{A} \in \{0,1\}^{n\times n}$ is the indicator matrix obtained from the (noisy) label matrix and is specified as follows:

$$a_{ij} = \begin{cases} 1, & o_i \text{ and } o_j \text{ have a common label} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

here $a_{ij} = 1$ means instance i and j share at least one common label, in other words, they are semantically similar to each other. Thus we should keep them close in the hash coding space.

The goal of cross-modal hashing is to learn a hash function for each modality and hash code for each instance. Here we denote hash functions of two modalities as $H^x(\mathbf{x}_i) \in \{-1,+1\}^c$ and $H^y(\mathbf{y}_i) \in \{-1,+1\}^c$, where c is the hash code length. We

Table 1
Mainly used notations.

Notation	Description
c	the hash code length
$\mathcal{O} = \{o_i\}_{i=1}^n$	training set, and o_i is a training instance
$\mathbf{l}_i \in \{0,1\}^L$	label vector of instance i
$H^x(\mathbf{x}_i)$, $H^y(\mathbf{y}_i)$	hash functions for modality X and Y
$\mathbf{h}_i^x$, $\mathbf{h}_i^y$	hash codes of instance i
$\mathbf{F}^x \in \mathbb{R}^{c\times n}$, $\mathbf{F}^y \in \mathbb{R}^{c\times n}$	deep feature matrices of X and Y
$\mathbf{B} \in \mathbb{R}^{c\times n}$	hash code matrix
$\mathbf{A} \in \{0,1\}^{n\times n}$	indicator matrix showing whether the instance pair (i,j) share a label
s_{ij}^{11}, s_{ij}^{22}	feature similarity in the modality X and Y for the instance pair
s_{ij}^l	label similarity between instances
Δs_{ij}	gap between feature similarity and label similarity of instance pair
ω_{ij}	weight for the instance pair

denote hash code of instance i from modality X as $\mathbf{h}_i^x = H^x(\mathbf{x}_i)$. For approximate nearest neighbour search purpose, the hashing function H^x and H^y should keep the proximity of the instances, i.e. if $a_{ij} = 0$, then the Hamming distance of $\mathbf{h}_i^x$ and $\mathbf{h}_j^y$ should be large, and should be small if $a_{ij} = 1$.

The whole hashing framework is shown in Fig. 1. As illustrated, our model contains two main modules: 1) a noise detection module and 2) a cross-modal hash learning module. We detect instance pairs with noisy labels in the noise detection module and give these pairs smaller weights. Then the weights are used to guide the training of the hash learning module. The specific details of the two modules will be presented in the following sections.

3.2. Cross-modal hashing

3.2.1. Feature learning

Due to the powerful representation learning ability of deep neural networks, we introduce two deep neural networks to learn the features of different modalities respectively. To improve the effectiveness, we use fine-tuning techniques and adopt two pre-trained neural networks. To learn the feature from image modality, we use a convolutional neural network adapted from CNN-F [5]. CNN-F is a network with eight layers, an input image size of 224 × 224 × 3, and output as a 1000-dimensional vector. CNN-F was originally used to classify and predict which category from 1000 classes the input image belongs to. CNN-F contains five convolutional layers (conv1-conv5) and three fully connected layers (fc6-fc8). The first seven layers of our image-net are the same with CNN-F, and we change the last fully connected layer to c nodes, where c is the preset hash code length. We use ReLU as the activation function for the first seven layers. Architecture of the image-net is listed in Table 2. Here "filter:*num* × *size* × *size*" denotes the number of convolution filters and their receptive field size. "st." denotes the convolution stride. "pad n" denotes that n pixels are added to each edge of the input image. "LRN" means Local Response Normalisation is applied to this layer. "×2 pool" means max-pooling downsampling is applied and the pooling window size is 2 × 2. We emphasize that we do not focus on the design of DNNs; we just use a convolutional neural network for its good performance in feature learning. Therefore, other deep feature learning models, such as AlexNet [20], can also be used here as an alternative.

For the text modality, we first represent the texts into bag-of-words (BOW) vectors. Then the BOW vectors are used as input for a pre-trained DNN whose outputs are the learned features of the text modality. The text-net contains two fully connected layers denoted as "fc1" and "fc2". "fc1" contains 8,192 nodes and "fc2" contains the same number of nodes as the hash code length. The activation function is ReLU, just the same as the image-net. The detailed configuration of this text feature learning network is also given in Table 2. We want to remark that the focus of our work is the design of hashing learning and noise detection module, not feature engineering. The used feature learning model can be replaced with other models, i.e., Transformer [35] for text modality.

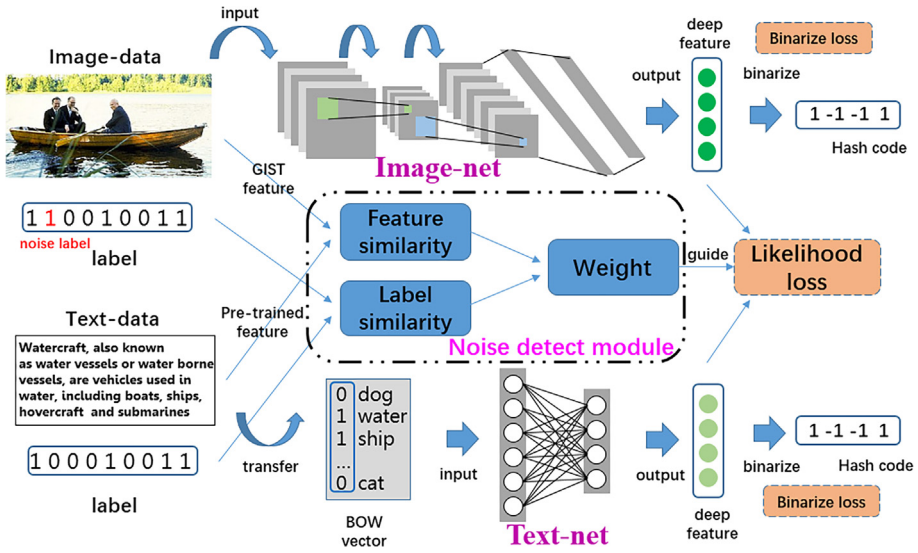


Fig. 1. *Module Overview*. NrDCMH consists of two parts: noise detection module and hash learning module. NrDCMH first detects noisy instance pairs using the gap between feature similarity and label similarity of training data. It then incorporates the gap to define a weighted likelihood loss to reduce the loss caused by noisy training pairs and to learn compatible deep features by CNNs and DNNs on image and text modalities, and also the hashing functions in a coherent fashion.

Table 2
Configuration of networks.

Network	Layers	Configuration
Image-net	conv1	filters: 64 × 11 × 11, st. 4, pad 0, LRN, × 2 pool
	conv2	filters: 256 × 5 × 5, st. 1, pad 2, LRN, × 2 pool
	conv3	filters: 256 × 3 × 3, st. 1, pad 1
	conv4	filters: 256 × 3 × 3, st. 1, pad 1
	conv5	filters: 256 × 3 × 3, st. 1, pad 1, × 2 pool
	fc6	nodes: 4096
	fc7	nodes: 4096
	fc8	nodes: code length c
Text-net	fc1	nodes: 8192
	fc2	nodes: code length c

3.2.2. Hash code learning

The principle of hashing is to preserve the proximity of data. As outlined in [4], given the indicator matrix $\mathbf{A} = \{a_{ij}\}$, we want to get hash codes with the highest posterior probability. Thus we use a logarithm maximum a posteriori estimation and Bayes formula as follows:

$$\begin{aligned} \log p(\mathbf{H}^x, \mathbf{H}^y|\mathbf{A}) &\propto \log p(\mathbf{A}|\mathbf{H}^x, \mathbf{H}^y)p(\mathbf{H}^x)p(\mathbf{H}^y) \\ &= \sum_{i,j} \log p(a_{ij}|\mathbf{h}_i^x, \mathbf{h}_j^y)p(\mathbf{h}_i^x)p(\mathbf{h}_j^y) \end{aligned} \tag{2}$$

where $p(a_{ij}|\mathbf{h}_i^x, \mathbf{h}_j^y)$ is the likelihood function, $p(\mathbf{h}^x)$ and $p(\mathbf{h}^y)$ are prior probabilities, $\mathbf{H}^x = [\mathbf{h}_1^x, \cdots, \mathbf{h}_m^x]$ and $\mathbf{H}^y = [\mathbf{h}_1^y, \cdots, \mathbf{h}_n^y]$ are the hash codes that we want to learn. Here, we consider the definition of the likelihood function $p(a_{ij}|\mathbf{h}_i^x, \mathbf{h}_j^y)$. As mentioned earlier, the principle of hashing is to preserve the proximity between instances, i.e., the smaller the Hamming distance between $\mathbf{h}_i^x$ and $\mathbf{h}_j^y$, the higher the probability that a_{ij} will be 1. $\mathbf{h}_i^x$ and $\mathbf{h}_j^y$ here are both binary code vectors, so the Hamming distance between $\mathbf{h}_i^x$ and $\mathbf{h}_j^y$ can be converted into an inner product:

$$dist_H(\mathbf{h}_i^x, \mathbf{h}_j^y) = \frac{1}{2}(c - \left\langle \mathbf{h}_i^x, \mathbf{h}_j^y \right\rangle) \tag{3}$$

where c is the length of hash codes, $\left\langle \mathbf{h}_i^x, \mathbf{h}_j^y \right\rangle$ denotes the inner product of $\mathbf{h}_i^x$ and $\mathbf{h}_j^y$. From Eq. (3), we know that $dist_H(\mathbf{h}_i^x, \mathbf{h}_j^y)$ is negatively correlated with $\left\langle \mathbf{h}_i^x, \mathbf{h}_j^y \right\rangle$. Thus, we define the likelihood function of a_{ij} as follows:

$$\begin{aligned} p(a_{ij}|\mathbf{h}_i^x, \mathbf{h}_j^y) &= \begin{cases} \sigma(\phi_{ij}) & a_{ij} = 1 \\ 1 - \sigma(\phi_{ij}) & a_{ij} = 0 \end{cases} \\ &= \sigma(\phi_{ij})^{a_{ij}}(1 - \sigma(\phi_{ij}))^{1-a_{ij}} \end{aligned} \tag{4}$$

where $\phi_{ij} = \frac{1}{2}\left\langle \mathbf{h}_i^x, \mathbf{h}_j^y \right\rangle$ denotes the inner product, and $\sigma(\phi) = \frac{1}{1+e^{-\phi}}$ denotes the Sigmoid function. The Sigmoid function $\sigma(\phi)$ is positively correlated with argument ϕ, so the smaller the Hamming distance between $\mathbf{h}_i^x$ and $\mathbf{h}_j^y$, the larger the inner product $\left\langle \mathbf{h}_i^x, \mathbf{h}_j^y \right\rangle$, which results in a higher probability that a_{ij} will be 1 and vice versa. Therefore, Eq. (4) is a good approximation of the likelihood function.

Let $\mathbf{F}^x = f^x(\mathbf{x}; \theta_x) \in \mathbb{R}^c$ and $\mathbf{F}^y = f^y(\mathbf{y}; \theta_y) \in \mathbb{R}^c$ denote the learned features of image modality and text modality, while θ_x and θ_y are the parameters of image-net and text-net, respectively. We can define our loss function as follows:

$$\begin{aligned} \min_{\mathbf{B}, \theta_x, \theta_y} \mathcal{J} = & -\sum_{i,j=1}^{n}(a_{ij}\Phi_{ij} - log(1 + e^{\Phi_{ij}})) \\ & +\alpha(||\mathbf{B} - \mathbf{F}^x||_F^2 + ||\mathbf{B} - \mathbf{F}^y||_F^2) \\ & +\beta(||\mathbf{F}^x\mathbf{1}||_F^2 + ||\mathbf{F}^y\mathbf{1}||_F^2) \\ s.t. \quad \mathbf{B} \in & \{+1, -1\}^{c \times n} \end{aligned} \tag{5}$$

where $\mathbf{F}^x \in \mathbb{R}^{c \times n}$ with $\mathbf{F}_{*i}^x = f^x(\mathbf{x}_i; \theta_x)$, $\mathbf{F}^y \in \mathbb{R}^{c \times n}$ with $\mathbf{F}_{*j}^y = f^y(\mathbf{y}_j; \theta_y)$ are outputs (deep features) of image-net and text-net, respectively. $\Phi_{ij} = \frac{1}{2}\left\langle \mathbf{F}_{*i}^x, \mathbf{F}_{*j}^y \right\rangle$ denotes the inner product. $\mathbf{B}_{*i}$ is the unified binary hash code for sample o_i. α, β are hyper-parameters. The first term of the loss function $-\sum_{i,j=1}^{n}(a_{ij}\Phi_{ij} - log(1 + e^{\Phi_{ij}}))$ is the negative log-likelihood of the cross-modal similarities, as defined in Eq. (4). Here we relax the binary constraints and regard the deep features as hash codes. We will binarize these deep features later. We see that minimizing the first term is equal to maximizing the likelihood func-

tion $\sigma(\Phi_{ij})^{a_{ij}}(1-\sigma(\Phi_{ij}))^{1-a_{ij}}$. More intuitively, by minimizing $-\sum_{i,j=1}^{n}(a_{ij}\Phi_{ij}-log(1+e^{\Phi_{ij}}))$, we make two instances with shared label ($a_{ij}=1$) having a large inner product, which implies a small Hamming distance in the hash code. In other words, by minimizing this term, we can preserve the proximity of instances in the hash codes. The second term $\alpha(||\mathbf{B}-\mathbf{F}^x||_F^2+||\mathbf{B}-\mathbf{F}^y||_F^2)$ is the binarization loss. By minimizing this term, we can force $\mathbf{B}$ as close as possible to both $\mathbf{F}^x$ and $\mathbf{F}^y$ and get the unified hash code. The third term $\beta(||\mathbf{F}^x\mathbf{1}||_F^2+||\mathbf{F}^y\mathbf{1}||_F^2)$ is to balance the hash code on all training instances. More intuitively, by minimizing this term, we can make the number of -1 and $+1$ in hash codes as balanced as possible, which helps to generate a lager coding space.

As we can see from the loss function, the effectiveness of NrDCMH depends heavily on the indicator matrix derived from class labels. Thus, label noise significantly compromises the performance of NrDCMH. Other supervised single/cross-modal hashing methods also suffer from this issue [2,49,19,24,25,41]. In the following subsection, we introduce a noisy instance detection module to detect noisy instance pairs and thus alleviate the impact of label noises.

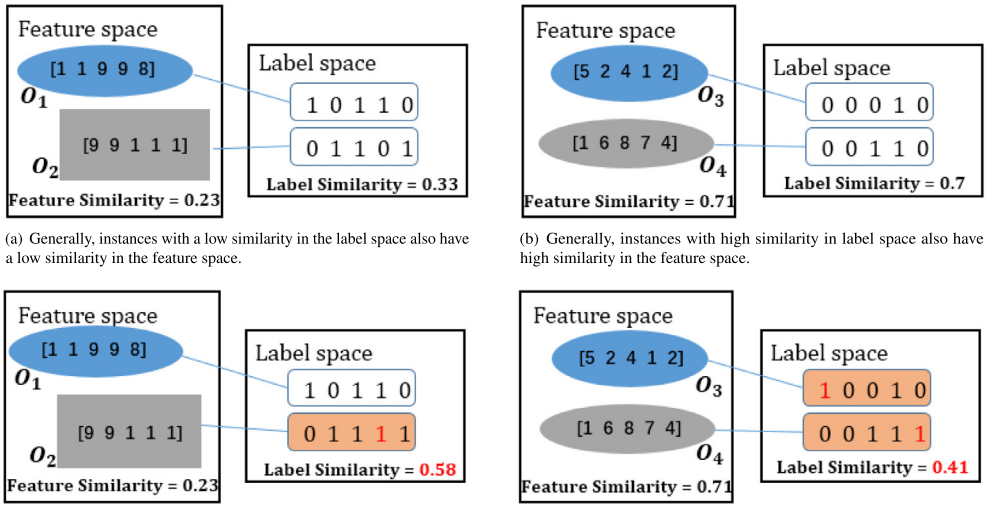
3.3. Noisy instance detection

As discussed earlier, label noise is destructive to the supervised CMH methods. To alleviate the impact of label noise, NrDCMH provides a weighting scheme to reduce the weights of noisy sample pairs in the training procedure. To mine the pairwise relationship in multi-modal data, NrDCMH learns from instance pairs rather than single instances to achieve cross-modal hashing. Here, we assign a weight ω_{ij} for each instance pair (o_i and o_j) rather than a single instance, and update our loss function as follows:

$$\begin{aligned}\min_{\mathbf{B},\theta_x,\theta_y}\mathcal{J}=\ & -\sum_{i,j=1}^{n}\omega_{ij}(a_{ij}\Phi_{ij}-log(1+e^{\Phi_{ij}}))\\ & +\alpha(||\mathbf{B}-\mathbf{F}^x||_F^2+||\mathbf{B}-\mathbf{F}^y||_F^2)\\ & +\beta(||\mathbf{F}^x\mathbf{1}||_F^2+||\mathbf{F}^y\mathbf{1}||_F^2)\\ s.t.\quad \mathbf{B}\in\ & \{+1,-1\}^{c\times n}\end{aligned} \tag{6}$$

ω_{ij} aims to reduce the impact of instances with label noises during model training and thus makes NrDCMH more robust toward label noises. The following elaborates on how to specify ω_{ij}.

Our noisy label detection strategy builds on a proposition that the labels of an instance depend on the features of this instance, and the semantic similarity is positively correlated with the feature similarity of the respective instances [45]. In supervised learning, the learnt features are deemed as highly-coupled with category labels, which also serve as the basis of semi-supervised learning and unsupervised learning. In other words, for a pair of training instances, if the similarity gap is large enough from labels and from features, this pair of training instances have a high probability of being corrupted, where



(a) Generally, instances with a low similarity in the label space also have a low similarity in the feature space.

(b) Generally, instances with high similarity in label space also have high similarity in the feature space.

(c) Labels of o_2 are corrupted by label noise, and thus label similarity becomes higher, which leads to a larger gap between feature similarity and label similarity.

(d) Labels of o_3 and o_4 are corrupted by noise; thus, the gap between feature similarity and label similarity becomes larger.

Fig. 2. Toy example of the positive correlation between label similarity and feature similarity. The negative correlation is often caused by corrupted labels. Under most 'noise-free' conditions, the semantic similarity is positively correlated with the feature similarity, as shown in (a) and (b). In (c) and (d), labels in red are noisy labels and they transform from 0 in (a) and (b), respectively. Label similarity is computed based on the labels of instances and the feature similarity is computed based on the feature vector using the cosine distance metric.

one or both of them is annotated with noisy labels. We use a simple toy example to explain this proposition in Fig. 2. Generally, according to this proposition, under most 'noise-free' conditions, the semantic similarity induced by labels is positively correlated with the feature similarity, as illustrated in Fig. 2(a) and (b). Due to label noise corruption, these two sub-figures change to Fig. 2(c) and (d), where the labels in red are noisy. As shown in Fig. 2(c) and (d), label noise makes the gap between feature similarity and label similarity larger. Therefore, we can detect instances with label noise using this gap to some extent. In some cases, label noise may reduce the gap between feature similarity and label similarity as well. However, we do not consider this because we do not have to get rid of all the noisy instances. Removing a certain amount of noisy instances can already make our model more robust under noisy label scenarios, and it is very difficult to remove all noisy instances.

To realize this idea, we design a noisy instance detection module by comparing the similarities induced from labels and those induced from features. More specifically, to evaluate the similarity in label space, we use cosine similarity and denote it as s_{ij}^{l}. For feature similarity, there are two different modalities (image and text), so we can not calculate the similarity directly. We thus calculate the similarity of each modality separately. For text modal, we use cosine similarity on the text feature vector learned by the per-learned text-net and denote it as s_{ij}^{11}. For image modal, we use cosine similarity on GIST vectors [28] extracted from raw image data and denote the similarity as s_{ij}^{22}. To prevent similarities from being negative, we map s_{ij}^{l}, s_{ij}^{11} and s_{ij}^{22} to [0,1] via normalization. Then we take the average of the two modalities $s_{ij}^{12} = \frac{1}{2}(s_{ij}^{11} + s_{ij}^{22})$ as the final feature similarity to balance the inconsistency between them. We can also use other similarity metrics for s_{ij}^{11} and s_{ij}^{22} here (like mapping the Euclidean distance to [0,1]). In our preliminary experiments, we found these metrics give a similar performance. Thus, for simplicity, we use cosine similarity for multi-modal data, just the same as we did for label similarity.

We quantify the gap between the feature similarity and label similarity Δs as follows:

$$\Delta s_{ij} = |s_{ij}^{12} - s_{ij}^{l}| \tag{7}$$

After preliminary experiments, we find it is more effective to fix ω_{ij} to 0 or 1 according to the ranking order of Δs_{ij}, rather than mapping the gap within [0,1]. Specifically, if Δs_{ij} is in the top $t\%$ of the ranking order, we set ω_{ij} to 0; otherwise, we set it to 1. Here the threshold t is a hyper-parameter. In this way, the number of noisy instance pairs is sharply reduced. Another advantage of this threshold strategy is that a considerable amount of training pairs are removed by setting their weights to 0. In this way, the total time consumption can be reduced, which becomes evident in the experiments.

$$\omega_{ij} = \begin{cases} 0 & \Delta s_{ij} > t\% \text{ elements in } \{\Delta s_{ij}\}, \\ 1 & \text{otherwise} \end{cases} \tag{8}$$

The above thresholding may not use some trustable instance pairs. Since noisy instance pairs often have a higher probability of being removed than trustable pairs, NrDCMH can still achieve a good performance even when some trustable instance pairs are removed. It should be noted that in our method, we use a fixed weight for each instance pair and do not update the difference of similarity Δs_{ij} and the weight obtained from it in each epoch of training. On the one hand, this is for the purpose of improving the training speed; on the other hand, we also experimented with variants with frequently updated weights, which proved that the effect of the method was not significantly improved. The results of the variation experiment will be shown in the section of ablation study.

3.4. Optimization

As we can see, there are three parameters (θ_x, θ_y and $\mathbf{B}$) to be optimized; however, we can not optimize three parameters simultaneously. Here we optimize these parameters by an alternative strategy that iteratively learns one of the three parameters at a time while fixing the other two.

Optimize θ_x (θ_y) We learn the image-net parameter θ_x with θ_y and $\mathbf{B}$ fixed. We use a back-propagation (BP) algorithm and stochastic gradient descent (SGD) to update θ_x iteratively. More specifically, we first calculate $\frac{\partial \mathcal{J}}{\partial \mathbf{F}_{*i}^{x}}$ for each training instance $\mathbf{x}_i$, then calculate $\frac{\partial \mathcal{J}}{\partial \theta_x}$ via chain rule and update θ_x by BP algorithm. $\frac{\partial \mathcal{J}}{\partial \mathbf{F}_{*i}^{x}}$ is calculated as follows:

$$\begin{aligned} \frac{\partial \mathcal{J}}{\partial \mathbf{F}_{*i}^{x}} = & \frac{1}{2}\sum_{j=1}^{n} \omega_{ij}(\sigma(\Phi_{ij})\mathbf{F}_{*j}^{y} - a_{ij}\mathbf{F}_{*j}^{y}) \\ & + 2\alpha(\mathbf{F}_{*i}^{x} - \mathbf{B}_{*i}) + 2\beta\mathbf{F}^{x}\mathbf{1} \end{aligned} \tag{9}$$

Then $\frac{\partial \mathcal{J}}{\partial \theta_x}$ is calculated as follows:

$$\frac{\partial \mathcal{J}}{\partial \theta_x} = \frac{\partial \mathcal{J}}{\partial \mathbf{F}_{*i}^{x}} \frac{\partial \mathbf{F}_{*i}^{x}}{\partial \theta_x} \tag{10}$$

The optimization of θ_y can be similarly made.

Optimize **B** When θ_x and θ_y are fixed, the optimization problem becomes:

$$\begin{aligned} \min_{\mathbf{B}} \quad & ||\mathbf{B}-\mathbf{F}^x||_F^2+||\mathbf{B}-\mathbf{F}^y||_F^2 \\ \text{s.t.} \quad & \mathbf{B}\in\{+1,-1\}^{c\times n} \end{aligned} \tag{11}$$

where

$$\begin{aligned} & ||\mathbf{B}-\mathbf{F}^x||_F^2+||\mathbf{B}-\mathbf{F}^y||_F^2 \\ = \; & tr\Big(2\mathbf{B}\mathbf{B}^T-\mathbf{B}(\mathbf{F}^x)^T-\mathbf{F}^x\mathbf{B}^T-\mathbf{B}(\mathbf{F}^y)^T \\ & -\mathbf{F}^y\mathbf{B}^T+\mathbf{F}^x(\mathbf{F}^x)^T+\mathbf{F}^y(\mathbf{F}^y)^T\Big) \end{aligned} \tag{12}$$

We note that $\mathbf{B}\in\{+1,-1\}^{c\times n}$, so we have $tr(\mathbf{B}\mathbf{B}^T)=nc$. θ_x and θ_y are fixed, thus $\mathbf{F}^x(\mathbf{F}^x)^T$ and $\mathbf{F}^y(\mathbf{F}^y)^T$ are fixed also. Then Eq. (11) can be reformulated as:

$$\begin{aligned} \max_{\mathbf{B}} \quad & tr(\mathbf{B}(\mathbf{F}^x+\mathbf{F}^y)^T) \\ \text{s.t.} \quad & \mathbf{B}\in\{+1,-1\}^{c\times n} \end{aligned} \tag{13}$$

We can easily solve this problem by letting $\mathbf{B}_{ij}$ with the same sign as $(\mathbf{F}^x+\mathbf{F}^y)_{ij}$:

$$\mathbf{B}=sign(\mathbf{F}^x+\mathbf{F}^y) \tag{14}$$

The whole procedure of NrDCMH is listed in Algorithm 1. For instances that are not in our training set (such as instances in the test set), we can obtain their hash codes via the learned hashing functions ($H^x(\mathbf{x})$ and $H^y(\mathbf{y})$). In particular, if we know a query image $\mathbf{x}_q$, we can obtain its hash codes as follows:

$$\mathbf{h}_q^x=H^x(\mathbf{x}_q)=sign(f(\mathbf{x}_q;\theta_x)) \tag{15}$$

and also for the query text. In this way, we can also obtain hash codes of query instances with only one modality.

Algorithm 1. NrDCMH: Deep Noise-robust Cross-Modal Hashing

Input: Data matrix of two modalities (X and Y), label matrix **L** with label noise.
Output: Hash code matrix **B**. Parameters of neural networks θ_x and θ_y.
1: Initialize neural network parameters θ_x and θ_y, mini-batch size $N=128$, and the number of iterations *iter*.
2: Calculate a_{ij}, according to Eq. (1).
3: Calculate ω_{ij}, according to Eq. (8).
4: **for** $t=1,2,\cdots,iter$ **do**
5: **for** $m\in\{x,y\}$ **do**
6: **for** $t_m=1,2,\cdots,\lceil n/N\rceil$ **do**
7: Randomly select N instances in X (or Y) to construct a mini-batch.
8: Calculate $\mathbf{F}_{*i}^x=f(\mathbf{x}_i;\theta_x)$ (or $\mathbf{F}_{*i}^y=g(\mathbf{y}_i;\theta_y)$) for each instance in mini-batch via forward propagation.
9: Calculate the derivative $\frac{\partial\mathcal{J}}{\partial\mathbf{F}_{*i}^x}$ (or $\frac{\partial\mathcal{J}}{\partial\mathbf{F}_{*i}^y}$) according to Eq. (9).
10: Updating θ_x (or θ_y) via the BP algorithm.
11: **end for**
12: **end for**
13: Calculate **B** according to Eq. (14).
14: **end for**

3.5. Multi-modal extension

As we mentioned in Section 3.1, NrDCMH can easily extend to multi-modal situations (number of modalities $\geqslant 3$). Suppose there are m modalities $X^{(1)},X^{(2)},\cdots,X^{(m)}$, we jointly optimize the likelihood loss for every two modalities; thus, the objection function becomes:

$$\begin{aligned} \min_{\mathbf{B},\theta_x,\theta_y} \mathcal{J}= \; & -\sum_{1\leqslant p<q\leqslant m}\sum_{i,j=1}^{n}\omega_{ij}(a_{ij}\Phi_{ij}^{pq}-log(1+e^{\Phi_{ij}^{pq}})) \\ & +\alpha\sum_{i=1}^{m}||\mathbf{B}-\mathbf{F}^{x_i}||_F^2+\beta\sum_{i=1}^{m}||\mathbf{F}^{x_i}\mathbf{1}||_F^2 \\ s.t. \quad & \mathbf{B}\in\{+1,-1\}^{c\times n} \end{aligned} \tag{16}$$

where $\Phi_{ij}^{pq}=\frac{1}{2}\left\langle\mathbf{F}_{*i}^{x_p},\mathbf{F}_{*j}^{x_q}\right\rangle$. The optimization process is similar to that of the two-modality case.

4. Experiments

In this section, we conduct a series of experiments to investigate the effectiveness of NrDCMH under different noisy levels and compare NrDCMH's performance with related methods. In addition, we also design a series of parameter experiments and ablation studies to prove the validity of each part of NrDCMH.

4.1. Experimental setup

We conduct experiments on four benchmark datasets: *Flickr-25 K* [16], *NUS-WIDE* [7], *Wiki* [30], and *Reuters* [1].

The *MirFlickr-25K* dataset contains 25,000 images collected from the Flickr website. Each instance contains an image from Flickr and its corresponding textual tags are annotated with at least one label from a total of 24 different semantic labels. Textual tags of each instance are converted into a 1386-dimensional BOW vector and regarded as text modal. For methods based on hand-crafted features, each image is converted to a 512-dimensional GIST vector.

The *NUS-WIDE* dataset contains 269,648 images and the associated textual tags are collected from the Flickr website. Each instance is annotated with one or more labels from 81 concept labels. Just like instances in *MirFlickr-25K*, we take textual tags as the text modality and convert them into a series of 1000-dimensional BOW vectors. We use 186,577 instances that belong to the most frequent ten concepts. For hand-crafted features based methods, each image is converted into a 500-dimensional bag-of-visual-words (BOVW) vector.

The *Wiki* dataset consists of 2,866 instances collected from the Wikipedia website. Each instance contains a picture-article pair, which is annotated with a topic label from ten categories. Each article is represented by a 10-dimensional semantic feature vector generated by latent Dirichlet allocation (LDA). For hand-crafted features based methods, we use a 128-dimensional BOV vector for each picture.

The *Reuters* dataset is a multi-language textual dataset which contains 18,758 instances. Each instance is a news article written in five languages: *English*, *French*, *German*, *Italian* and *Spanish*. Each language is taken as a kind of modality and each modality is represented by a BOW vector. We convert these vectors into 1000-dimensional semantic feature vectors by LDA.

These datasets are manually labeled; thus, we take these original labels as 'noise-free' labels and inject random noise into these datasets for our noise-robust experiments. We set a noise rate r, which means $r \times 100\%$ labels in the one-hot coding label vector randomly shifted from 0 to 1. For example, there is an instance with one-hot coding label vector $\mathbf{l} = [1, 0, 0, 1, 0]$. Every 0 in l has a probability r of shifting to 1. In our experiments, we test two settings with $r = 0.2$ (low noise level) and $r = 0.4$ (high noise level). Following the canonical setting [19,22], we split each dataset into a training set and a test set (query set), while the whole set is used as the retrieval set. More specifically, after model training with instances in the training set, we obtain hash codes for each instance in the retrieval set and test set through the model. Then we search for similar instances in the retrieval set for each query in the test set. Details of dataset partition are shown in Table 3.

Seven related and representative methods are adopted for empirical comparison, including Deep Cross-Modal Hashing (DCMH) [19], Semantic Correlation Maximization (SCM-seq and SCM-orth) [46], Ranking-based Deep Cross-modal Hashing (RDCMH) [25], Semantics Preserving Hashing (SePH) [22], Collective Matrix Factorization Hashing (CMFH) [8] and Semantic Deep Cross-modal Hashing (SDCH) [41]. SCM-seq, SCM-orth, and SePH are supervised methods with shallow architecture, while DCMH, SDCH and RDCMH are supervised (or semi-supervised) methods with deep architecture. Besides, we also adopt an unsupervised method CMFH [8] for a comprehensive comparison. Because RDCMH cannot apply to datasets with more than two modalities, it is thus not used for the experiments on *Reuters*. For each comparison method, we use the hyper-parameters recommended in the corresponding article or shared codes. For NrDCMH, we set $\alpha = \beta = 1$, we let $t = 30$ when $r = 0.2$ while we let $t = 40$ when $r = 0.4$.

To quantitatively study the performance, we choose the widely used evaluation metric, mean average precision (MAP), to quantify the performance of hashing retrieval methods [2,46]. The definition of MAP is as follows:

$$MAP = \frac{1}{|\mathcal{Q}|}\sum_{i=1}^{|\mathcal{Q}|}\frac{1}{m_i}\sum_{j=1}^{m_i}P(j) \tag{17}$$

Table 3
Dataset partition.

Dataset	Modality	Size	Training set	Test set
MirFlickr	2	20,015	10,000	2,000
NUS-wide	2	186,577	10,000	2,000
Wiki	2	2,866	2,150	700
Reuters	5	18,758	10,000	2,000

where $\mathcal{Q}$ is the query set, m_i is the number of ground-truth labeled relevant instances in the retrieval set for the i-th query. For each query, we obtain a ranking list of the retrieval set according to the Hamming distance between the query and each instance in the retrieval set. From the first instance in the rank list to the j-th relevant instance, we get the top j retrieval subset and $P(j)$ is the precision of this subset. More intuitively, for each instance in query set $\mathcal{Q}$, MAP first calculates an average precision (AP) value and then takes the mean value of these APs. A larger MAP value implies a better hashing retrieval performance.

To make the evaluation more convincing, we use Normalized Discounted Cumulative Gain (NDCG) as another evaluation metric. NDCG is also widely-used to evaluate the performance of retrieval methods. The definition of NDCG at m is as follows:

$$NDCG@m = \frac{DCG@m}{IDCG@m} \tag{18}$$

where DCG@m is the Discounted Cumulative Gain and IDCG@m is the maximum of DCG@m, DCG@m is calculated as follows:

$$DCG@m = \sum_{j=1}^{m} \frac{r(j)}{\log_2(j+1)} \tag{19}$$

where $r(j)$ is the similarity relation of query instance and the j-th retrieval result. Here we set $r(j) = 1$ if the query instance is similar with the j-th retrieval result (the two instances belong to at least one same category) and otherwise $r(j) = 0$. A larger NDCG value implies a better retrieval performance. To avoid random factors, we repeat the experiments five times and report the average MAP and NDCG@100.

4.2. Results and analysis

4.2.1. Results on MirFlickr, NUS-wide and Wiki

We conducted our experiments under three different settings: 'noise-free' scenario (noise rate $r = 0$), low noise level ($r = 0.2$) and high noise level ($r = 0.4$). We report the experimental results in Table 4–9. In the tables, 'I to T' denotes that the query is an image and we want to search texts in the retrieval set. 'T to I' denotes that we search images with query text. From Table 4–9, we can make the following observations:

(i) Under the 'noise-free' setting, RDCMH and SDCH achieve a better performance than NrDCMH, while the performance of NrDCMH is comparable with DCMH. As we stated, NrDCMH targets cross-modal hashing with label noise and can be seen as a weighted version of DCMH with a noise detection module. NrDCMH degenerates to DCMH under a 'noise-free' setting. They both consider only the pairwise rank between instances, so NrDCMH holds a similar performance as DCMH. In contrast, RDCMH adopts the triplet loss to explore high-order rank information among instances and thus performs better in this setting. Although DCMH, NrDCMH, and SDCH use similar deep feature learning networks, SDCH adopts an extra deep network to learn hash codes and an extra encoding loss to preserve semantic similarity further. Therefore, SDCH performs better under the 'noise-free' setting.

(ii) All compared methods manifest reduced MAP values and NDCG values even with a low ratio of noisy labels. This is because label information has a significant impact on the effectiveness of supervised/semi-supervised hashing methods, which supports our motivation to model the noisy labels for robust cross-modal hashing explicitly. We see that

Table 4
Results (mAP) on MirFlickr-25K under different noise rates r.

	noise rate r	0				0.2				0.4			
	hash bits c	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
I to T	SCM-seq	0.628	0.635	0.632	0.621	0.607	0.609	0.612	0.620	0.577	0.578	0.581	0.588
	SCM-orth	0.580	0.581	0.590	0.587	0.561	0.565	0.568	0.572	0.545	0.546	0.554	0.557
	SePH	0.657	0.660	0.667	0.662	0.596	0.602	0.608	0.607	0.510	0.514	0.522	0.527
	CMFH	0.610	0.612	0.624	0.625	0.608	0.617	0.627	0.626	0.607	0.615	0.620	0.623
	DCMH	0.741	0.746	0.754	0.749	0.711	0.715	0.719	0.714	0.660	0.679	0.682	0.685
	RDCMH	**0.772**	0.773	0.772	0.768	0.727	0.730	0.732	0.735	0.672	0.675	0.679	0.671
	SDCH	0.771	**0.780**	**0.781**	**0.779**	0.725	0.730	0.733	0.731	0.665	0.670	0.672	0.671
	NrDCMH	0.752	0.745	0.745	0.742	**0.729**	**0.732**	**0.737**	**0.736**	**0.687**	**0.690**	**0.697**	**0.699**
T to I	SCM-seq	0.619	0.630	0.635	0.640	0.608	0.610	0.621	0.617	0.570	0.574	0.577	0.564
	SCM-orth	0.590	0.598	0.610	0.602	0.588	0.580	0.598	0.597	0.565	0.554	0.567	0.569
	SePH	0.648	0.652	0.655	0.658	0.627	0.633	0.638	0.635	0.533	0.534	0.541	0.539
	CMFH	0.625	0.629	0.637	0.640	0.621	0.629	0.636	0.644	0.628	0.632	0.635	0.640
	DCMH	0.782	0.790	0.794	0.779	0.758	0.767	0.762	0.768	0.721	0.732	0.735	0.737
	RDCMH	0.793	0.792	0.797	0.795	0.754	0.759	0.765	0.761	0.715	0.718	0.722	0.719
	SDCH	**0.806**	**0.812**	**0.817**	**0.810**	0.765	0.772	0.774	0.778	0.730	0.736	0.741	0.739
	NrDCMH	0.779	0.796	0.790	0.784	**0.768**	**0.774**	**0.781**	**0.779**	**0.738**	**0.746**	**0.748**	**0.751**

Table 5
Results (mAP) on NUS-WIDE under different noise rates *r*.

	noise rate *r*	0				0.2				0.4			
	hash bits *c*	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
I to T	SCM-seq	0.452	0.463	0.467	0.462	0.412	0.426	0.428	0.431	0.392	0.404	0.407	0.409
	SCM-orth	0.431	0.440	0.447	0.451	0.403	0.407	0.415	0.412	0.382	0.384	0.391	0.390
	SePH	0.487	0.492	0.497	0.503	0.435	0.447	0.448	0.452	0.419	0.426	0.428	0.430
	CMFH	0.449	0.457	0.468	0.467	0.452	0.455	0.460	0.468	0.447	0.460	0.466	0.465
	DCMH	0.624	0.630	0.641	0.645	0.589	0.592	0.598	0.594	0.554	0.559	0.561	0.564
	RDCMH	0.623	0.631	0.632	0.635	0.590	0.598	0.603	0.601	0.557	0.564	0.569	0.570
	SDCH	**0.645**	**0.658**	**0.662**	**0.664**	**0.606**	0.608	0.611	0.620	0.545	0.550	0.558	0.563
	NrDCMH	0.621	0.625	0.640	0.639	0.605	**0.612**	**0.624**	**0.623**	**0.562**	**0.576**	**0.581**	**0.583**
T to I	SCM-seq	0.495	0.502	0.498	0.497	0.467	0.471	0.475	0.478	0.426	0.428	0.431	0.435
	SCM-orth	0.481	0.492	0.495	0.489	0.452	0.458	0.461	0.462	0.413	0.415	0.421	0.428
	SePH	0.514	0.526	0.529	0.532	0.476	0.487	0.489	0.488	0.447	0.451	0.463	0.465
	CMFH	0.477	0.479	0.484	0.481	0.469	0.480	0.485	0.487	0.472	0.479	0.483	0.485
	DCMH	0.653	0.664	0.665	0.668	0.621	0.638	0.642	0.643	0.601	0.612	0.619	0.617
	RDCMH	0.659	0.665	0.669	0.668	0.628	0.631	0.635	0.637	0.610	0.621	0.625	0.627
	SDCH	**0.672**	**0.684**	**0.687**	**0.689**	0.632	0.635	0.647	0.646	0.612	0.619	0.626	0.629
	NrDCMH	0.651	0.662	0.667	0.665	**0.641**	**0.652**	**0.660**	**0.664**	**0.621**	**0.634**	**0.639**	**0.639**

Table 6
Results (mAP) on Wiki under different noise rates *r*.

	noise rate *r*	0				0.2				0.4			
	hash bits *c*	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
I to T	SCM-seq	0.245	0.251	0.258	0.250	0.215	0.229	0.228	0.214	0.197	0.186	0.194	0.199
	SCM-orth	0.175	0.185	0.184	0.189	0.164	0.166	0.159	0.162	0.157	0.149	0.142	0.151
	SePH	0.475	0.503	0.492	0.499	0.421	0.419	0.414	0.418	0.370	0.383	0.379	0.371
	CMFH	0.375	0.383	0.387	0.390	0.376	0.384	0.385	0.390	0.372	0.390	0.385	0.399
	DCMH	0.477	0.501	0.506	0.508	0.422	0.425	0.423	0.434	0.361	0.365	0.356	0.371
	RDCMH	0.481	0.491	0.509	0.512	0.413	0.435	0.438	0.431	0.387	0.389	0.390	0.386
	SDCH	**0.512**	**0.523**	**0.521**	**0.518**	0.438	0.436	0.447	0.439	0.392	0.389	0.391	0.385
	NrDCMH	0.472	0.491	0.507	0.510	**0.445**	**0.449**	**0.451**	**0.441**	**0.411**	**0.415**	**0.395**	**0.412**
T to I	SCM-seq	0.257	0.261	0.268	0.254	0.232	0.235	0.229	0.237	0.214	0.209	0.201	0.215
	SCM-orth	0.148	0.156	0.158	0.167	0.132	0.135	0.149	0.143	0.125	0.132	0.129	0.127
	SePH	0.550	0.567	0.562	0.572	0.501	0.498	0.515	0.514	0.449	0.451	0.450	0.461
	CMFH	0.436	0.437	0.445	0.456	0.434	0.436	0.449	0.458	0.437	0.432	0.449	0.453
	DCMH	0.601	0.612	0.615	0.620	0.551	0.549	0.547	0.555	0.503	0.498	0.496	0.502
	RDCMH	0.581	0.611	0.623	0.615	0.555	0.542	0.552	0.560	0.501	0.505	0.513	0.508
	SDCH	**0.623**	**0.651**	**0.642**	**0.644**	0.571	0.574	0.572	0.569	0.521	0.519	0.515	0.525
	NrDCMH	0.605	0.614	0.609	0.618	**0.576**	**0.581**	**0.577**	**0.575**	**0.531**	**0.535**	**0.530**	**0.538**

Table 7
Results (NDCG@100) on MirFlickr-25K under different noise rates *r*.

	noise rate *r*	0				0.2				0.4			
	hash bits *c*	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
I to T	SCM-seq	0.348	0.351	0.352	0.354	0.322	0.321	0.325	0.327	0.301	0.303	0.304	0.307
	SCM-orth	0.349	0.352	0.348	0.350	0.315	0.317	0.318	0.320	0.295	0.298	0.304	0.301
	SePH	0.387	0.388	0.391	0.358	0.361	0.365	0.366	0.368	0.337	0.335	0.338	0.336
	CMFH	0.323	0.325	0.330	0.328	0.323	0.324	0.331	0.330	0.325	0.325	0.329	0.330
	DCMH	0.408	0.412	0.420	0.422	0.367	0.369	0.371	0.377	0.341	0.344	0.353	0.357
	RDCMH	0.413	0.415	0.422	0.421	0.368	0.373	0.379	0.381	0.354	0.355	0.359	0.358
	SDCH	**0.421**	**0.425**	**0.424**	**0.427**	**0.387**	0.390	0.389	0.392	0.361	0.365	0.366	0.364
	NrDCMH	0.410	0.415	0.421	0.430	0.390	**0.391**	**0.395**	**0.394**	**0.371**	**0.375**	**0.374**	**0.373**
T to I	SCM-seq	0.375	0.382	0.384	0.388	0.332	0.335	0.338	0.340	0.312	0.317	0.319	0.316
	SCM-orth	0.378	0.380	0.381	0.383	0.330	0.328	0.335	0.337	0.310	0.309	0.315	0.314
	SePH	0.403	0.405	0.407	0.403	0.352	0.356	0.358	0.357	0.323	0.322	0.331	0.328
	CMFH	0.344	0.351	0.348	0.350	0.345	0.350	0.344	0.348	0.341	0.345	0.350	0.348
	DCMH	0.448	0.451	0.452	0.455	0.397	0.395	0.401	0.403	0.371	0.368	0.372	0.375
	RDCMH	0.454	0.458	0.456	0.460	0.403	0.405	0.408	0.410	0.379	0.383	0.385	0.382
	SDCH	**0.465**	**0.468**	**0.470**	**0.469**	0.421	0.420	0.425	0.428	0.390	0.395	0.394	0.402
	NrDCMH	0.450	0.455	0.452	0.454	**0.431**	**0.429**	**0.433**	**0.435**	**0.405**	**0.412**	**0.411**	**0.415**

Table 8
Results (NDCG@100) on NUS-WIDE under different noise rates *r*.

	noise rate *r*	0				0.2				0.4			
	hash bits *c*	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
I to T	SCM-seq	0.312	0.321	0.325	0.319	0.289	0.287	0.291	0.292	0.266	0.259	0.260	0.263
	SCM-orth	0.295	0.299	0.296	0.287	0.261	0.265	0.267	0.268	0.241	0.243	0.242	0.249
	SePH	0.347	0.349	0.352	0.348	0.312	0.311	0.315	0.318	0.280	0.283	0.279	0.281
	CMFH	0.294	0.295	0.299	0.296	0.294	0.297	0.298	0.290	0.296	0.298	0.301	0.299
	DCMH	0.368	0.372	0.375	0.377	0.321	0.325	0.328	0.330	0.302	0.298	0.299	0.305
	RDCMH	0.369	0.371	0.377	0.380	0.325	0.326	0.331	0.327	0.308	0.305	0.306	0.306
	SDCH	**0.375**	**0.385**	**0.388**	**0.384**	0.339	0.334	0.341	0.345	0.313	0.318	0.315	0.317
	NrDCMH	0.367	0.369	0.376	0.379	**0.342**	**0.345**	**0.349**	**0.348**	**0.321**	**0.319**	**0.325**	**0.324**
T to I	SCM-seq	0.324	0.336	0.338	0.338	0.291	0.294	0.299	0.298	0.274	0.278	0.277	0.280
	SCM-orth	0.317	0.322	0.325	0.327	0.286	0.288	0.295	0.291	0.261	0.263	0.266	0.265
	SePH	0.366	0.368	0.371	0.375	0.326	0.328	0.331	0.329	0.301	0.305	0.312	0.310
	CMFH	0.310	0.312	0.316	0.318	0.311	0.315	0.319	0.322	0.311	0.316	0.319	0.322
	DCMH	0.390	0.391	0.401	0.398	0.344	0.342	0.345	0.348	0.322	0.325	0.328	0.324
	RDCMH	0.399	0.398	0.399	0.402	0.349	0.348	0.349	0.347	0.335	0.334	0.334	0.331
	SDCH	**0.412**	**0.414**	**0.418**	**0.416**	0.361	0.368	0.362	0.362	0.340	0.344	0.345	0.348
	NrDCMH	0.391	0.390	0.397	0.395	**0.378**	**0.376**	**0.365**	**0.368**	**0.357**	**0.359**	**0.358**	**0.356**

Table 9
Results (NDCG@100) on Wiki under different noise rates *r*.

	noise rate *r*	0				0.2				0.4			
	hash bits *c*	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit	16bit	32bit	64bit	128bit
I to T	SCM-seq	0.261	0.266	0.268	0.271	0.230	0.233	0.238	0.241	0.211	0.218	0.220	0.223
	SCM-orth	0.251	0.255	0.257	0.257	0.213	0.217	0.220	0.220	0.187	0.187	0.189	0.195
	SePH	0.335	0.338	0.341	0.338	0.291	0.301	0.305	0.308	0.271	0.276	0.275	0.278
	CMFH	0.287	0.285	0.288	0.290	0.290	0.295	0.297	0.298	0.289	0.291	0.291	0.293
	DCMH	0.351	0.355	0.357	0.360	0.318	0.319	0.316	0.319	0.287	0.291	0.295	0.297
	RDCMH	0.358	0.356	0.362	0.361	0.320	0.321	0.317	0.323	0.289	0.302	0.301	0.308
	SDCH	**0.362**	**0.368**	**0.370**	**0.365**	0.328	0.331	0.329	0.329	0.308	0.309	0.309	0.302
	NrDCMH	0.350	0.348	0.351	0.355	**0.332**	**0.334**	**0.333**	**0.335**	**0.315**	**0.313**	**0.316**	**0.312**
T to I	SCM-seq	0.303	0.305	0.308	0.311	0.261	0.263	0.263	0.264	0.241	0.245	0.244	0.248
	SCM-orth	0.298	0.302	0.302	0.305	0.253	0.252	0.258	0.254	0.230	0.229	0.231	0.234
	SePH	0.378	0.376	0.380	0.382	0.331	0.336	0.341	0.342	0.310	0.311	0.321	0.318
	CMFH	0.310	0.312	0.314	0.317	0.312	0.310	0.315	0.315	0.316	0.318	0.315	0.319
	DCMH	0.412	0.423	0.421	0.422	0.367	0.375	0.365	0.368	0.344	0.345	0.344	0.348
	RDCMH	0.425	0.424	0.427	0.431	0.370	0.371	0.378	0.373	0.348	0.350	0.350	0.352
	SDCH	**0.435**	**0.438**	**0.440**	**0.437**	0.381	0.388	0.386	0.388	0.361	0.364	0.363	0.365
	NrDCMH	0.411	0.418	0.420	0.422	**0.389**	**0.389**	**0.388**	**0.391**	**0.371**	**0.377**	**0.377**	**0.376**

NrDCMH is much less impacted by label noise than the other methods, especially under a low level of label noises. For example, in Table 4 (the same conclusion is made from other tables), we observe that under low noise level ($r = 0.2$), MAP of NrDCMH only reduces by $1 \sim 2\%$, while DCMH reduces about 3%, RDCMH reduces by about 3.5% and SDCH reduces by about 5%. RDCMH uses the triplet ranking loss, which performs better at preserving cross-modal similarity, and is also more susceptible to label noise. SDCH uses the similarity calculated from class labels in both the feature learning procedure and hash code learning procedure; thus, it suffers more from label noise. NrDCMH reduces the weights of noisy instance pairs and thus alleviates the impact caused by label noise. Generally, CMH methods with shallow architecture (SCM and SePH) are outperformed by methods with deep architecture, and they also show a reduced performance with the label noise. On the whole, NrDCMH shows a better noise-robustness than other supervised methods under a low noise level.

(iii) Under high-level label noise ($r = 0.4$), the MAP and NDCG values of all the supervised methods, including NrDCMH, reduce much more. However, compared with DCMH (MAP reduces by 6%, NDCG reduces by 6%), SDCH (MAP reduces by 7%, NDCG reduces by 6.5%), and other compared methods, NrDCMH (MAP reduces by 4.5%, NDCG reduces by 4.5%) still shows a better noise-robustness under such a high noise level.

(iv) CMFH is an unsupervised CMH method that does not take advantage of label information and thus performs stably under different ratios of noisy labels. The MAP and NDCG values of CMFH are generally lower than that of supervised methods under the 'noise-free' scenarios. Nevertheless, CMFH is comparable with SCM-seq and SCM-orth (on *MirFlickr-25 K* and *NUS-wide*), and even outperforms them (on *Wiki*). The reason is that SCM simply uses a linear transformation to learn the hash function while trying to reconstruct the cosine similarity matrix with obtained hash codes, and it ignores structure information. In contrast, CMFH learns latent semantic space by decomposing the feature

matrix. The shortcomings of the SCM are more apparent in the *Wiki* dataset. As we can see, *Wiki* is a single label dataset, and cosine similarity is not a good semantic measure for single label instances. Hence, SCM performs poorly on *Wiki*. With the noise rate increase, the MAP and NDCG values of supervised methods reduce sharply at the interference of label noise. CMFH outperforms SePH and SCM when $r = 0.4$. Although corrupted by label noise, there is valuable semantic information of class label. Therefore, supervised deep CMH methods (DCMH, RDCMH, SDCH and NrDCMH) performs better than CMFH.

4.2.2. Results on Reuters

To evaluate the effectiveness of different methods in multi-modal datasets (number of modalities $\geqslant 3$), we conduct experiments on the *Reuters* dataset, and report the results in Table 10. RDCMH is not listed, because its source codes can not be easily adapted for multi-modal data. Just like 'I to T' and 'T to I' in Tables 4–6, 'ENG' in Table 10 means the query instance is in English and we search text with the same topic in other languages. So as 'FRE', 'GER', 'ITA' and 'SPA'. From Table 10, we observe:

(i) Under 'noise-free' settings, SDCH achieves a better performance than NrDCMH while the performance of NrDCMH is comparable with DCMH. NrDCMH outperforms SCM-seq, SCM-orth, and SePH even under 'noise-free' settings.
(ii) With noise rate increases, NrDCMH shows a better noise-robustness than the other methods and outperforms almost all other methods under different levels of label noise.
(iii) NrDCMH can also achieve better performance than CMFH, although CMFH is not affected by noise.

All the reasons are the same as for our analysis in the previous section, which is why we do not repeat them here. On the whole, from the experiments on *Reuters*, we can conclude that NrDCMH can be easily adapted to multi-modal data and that it outperforms other comparison methods under high noise rate situations.

4.2.3. Parameter analysis

In the NrDCMH model, there are four hyper-parameters: α, β, t, and hash bits c (also referred to as code length). The impact of c is reported in the results shown in Tables 4–6. When the hash code length c increases, the MAP of NrDCMH also increases. However, when c increases to a particular value (about 64), MAP decreases slowly or at least begins to decrease. As we can see, this phenomenon occurs in almost all the CMH methods. This is caused by the increase in the hash code length, where the coding space first gets larger, resulting in a better performance. However when the coding space gets too large, the model learns a lot of redundant features, which in turn harms the effectiveness.

We further conduct a series of experiments to verify how these parameters affect our model. In addition, these parameter analysis experiments can also verify the validity of each term in our final object function in Eq. (6).

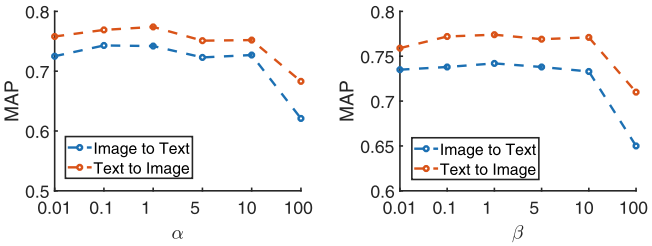
For parameter α, we conduct a series of experiments on the *MirFlicrk-25 K* dataset where the noise rate $r = 0.2$, code length $c = 32, \beta = 1$ and $t = 30$. Fig. 3 shows that when $\alpha < 1$, with the increase of α, MAP also increases, and when $\alpha > 1$, MAP starts to decrease. On the whole, NrDCMH is not very sensitive to α when $0.01 < \alpha < 10$, and MAP falls sharply when $10 < \alpha < 100$ because, when $\alpha > 10$, the role of the second term of Eq. (6) is overemphasized, while the coherence between semantic labels and features of samples are under-weighted. Thus, the model learns less information from the first term, which mines the information from both the label and feature spaces and also from the pairwise relationship between samples within/between modalities, so the overall effectiveness of NrDCMH falls when α is too large.

For parameter β, just like parameter α, we conduct a series of experiments on the *MirFlicrk-25 K* dataset where the noise rate $r = 0.2$, code length $c = 32, \alpha = 1$, and $t = 30$. Fig. 3 shows that NrDCMH is not sensitive to parameter β on the scale of $0.01 < \beta < 10$, and when $\beta > 10$, MAP decreases sharply. The reason is just the same as that for α. When $\beta > 10$, the third term (for balanced hash codes learning) of Eq. (6) is overemphasized, while the unified hash codes are less focused, leading to a fall in the overall effectiveness of NrDCMH.

Next, we focus on t, which is a hyper-parameter that determines how many pairs in Eq. (6) are induced from instances with noisy labels. To reduce the impact of instances corrupted with noisy labels, we exclude the pairs ranked in top $t\%$ for optimization. We conduct a series of experiments on the *MirFlickr-25K* dataset and study the sensitivity of t by varying it

Table 10
Results (mAP) on Reuters under different noise rates r.

noise rate r	0					0.2					0.4				
query modal	ENG	FRE	GER	ITA	SPA	ENG	FRE	GER	ITA	SPA	ENG	FRE	GER	ITA	SPA
SCM-seq	0.229	0.230	0.221	0.240	0.238	0.208	0.201	0.199	0.211	0.223	0.209	0.208	0.205	0.197	0.205
SCM-orth	0.230	0.240	0.231	0.250	0.248	0.218	0.211	0.209	0.221	0.233	0.219	0.218	0.215	0.207	0.215
SePH	0.310	0.308	0.291	0.304	0.315	0.287	0.284	0.276	0.298	0.287	0.254	0.256	0.248	0.244	0.259
CMFH	0.287	0.289	0.277	0.284	0.278	0.292	0.287	0.271	0.285	0.273	0.291	0.284	0.276	0.286	0.271
DCMH	0.341	0.345	0.317	0.342	0.338	0.314	0.311	0.289	0.310	0.301	0.286	0.269	0.258	0.274	0.269
SDCH	**0.361**	**0.381**	**0.344**	**0.351**	**0.355**	**0.326**	0.320	0.317	0.328	0.320	0.287	0.286	0.279	0.293	0.284
NrDCMH	0.339	0.342	0.329	0.344	0.341	0.314	**0.322**	**0.319**	**0.330**	**0.328**	**0.301**	**0.308**	**0.298**	**0.304**	**0.299**



(a) α vs. MAP (b) β vs. MAP

Fig. 3. Parameter analysis for α and β on the *MirFlickr-25 K* dataset.

from 0 to 50. Results are shown in Fig. 4. We find that the MAP values steadily increase at the beginning, and hold a relatively stable performance when $t \approx 30$. The MAP value starts to fall as t increases further. This occurs because as t increases, more and more training pairs with a high probability of being corrupted are removed (training weight ω_{ij} in Eq. (6) is set to 0). As a result, the impact of noisy labels gradually reduces and an improved MAP value is obtained. However, some pairs induced from instances without corrupted labels are also removed by mistake. This creates some kind of equilibrium whereby the NrDCMH results increase only a little as t increases further. When t becomes large enough, more training pairs induced from the noise-free training instances are mistakenly removed, the 'equilibrium' is broken, and the effectiveness starts to fall. Thus, we should choose a proper value for t under different noise levels. Generally, to not significantly reduce the number of training pairs, a relatively large t usually leads to a better performance; thus, we adopt $t = 30$ for a low noise level ($r = 0.2$) and $t = 40$ for a high noise level ($r = 0.4$).

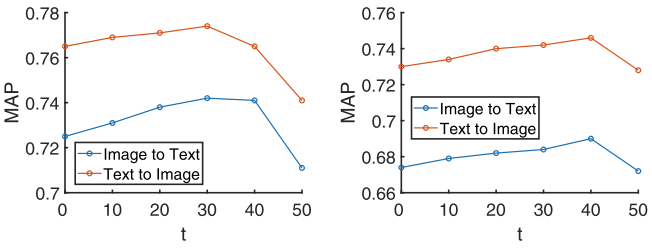
4.2.4. Ablation study

In this section, we conduct a series of experiments on *MirFlickr* dataset to evaluate the effectiveness of different modules of NrDCMH. We use two variations of NrDCMH: (i) **NrDCMH-n** removes the noise detection module, (ii)**NrDCMH-e** updates weight ω_{ij} in each epoch using the currently learned features.

The recorded results are listed in Table 11 and Table 12. We can observe that while the noise detection module does not manifest a better performance in the case of noise-free $r = 0$, it holds a clear better performance in the case of noise $r > 0$. This proves the effectiveness of our noise detection module. Although NrDCMH-e updates weights in each epoch, it gives a similar performance as NrDCMH. Given that, we use the weights originally calculated by GIST as we stated in Section 3.3 to save the computation time.

4.2.5. Runtime analysis

Generally, the training speed of models with shallow architectures is much faster than models with deep architectures. Here we show the average training time of the different methods with deep architectures in Table 13. We set $r = 0.2$ and $c = 32$. All experiments are conducted on a server with Intel E5-2650v3 and Tesla K40. The missing data occurs because RDCMH can not be easily adapted for a dataset with more than two modalities. We observe that NrDCMH is the fastest method because NrDCMH excludes a part of the training pairs by setting their weight to 0 and thus reduces the calculation time. SDCH is slow because of its deeper architecture. SDCH adds one more fully connected layer, which is time-consuming to achieve a better performance. Thus, SDCH is slower than DCMH and NrDCMH. RDCMH is slower because it uses a triplet ranking loss, which is with the time complexity of $O(n^3)$. While other methods use cross-entropy loss or likelihood loss with time complexity of $O(n^2)$.



(a) $r = 0.2$ (b) $r = 0.4$

Fig. 4. Parameter sensitive analysis for t (a portion of training pairs is removed) on the *MirFlickr-25K* dataset.

Table 11
Ablation study results (mAP) on MirFlickr-25K under different noise rates *r*.

	noise rate *r*	0			0.2			0.4		
	hash bits *c*	16bit	32bit	64bit	16bit	32bit	64bit	16bit	32bit	64bit
I to T	NrDCMH	**0.752**	0.745	0.745	0.729	0.732	**0.737**	0.687	0.690	**0.697**
	NrDCMH-n	0.746	**0.747**	**0.749**	0.710	0.712	0.715	0.667	0.669	0.682
	NrDCMH-e	0.741	0.740	0.743	**0.731**	**0.733**	0.734	**0.688**	**0.691**	0.695
T to I	NrDCMH	0.779	0.796	**0.790**	0.768	**0.774**	**0.781**	0.738	**0.746**	**0.748**
	NrDCMH-n	**0.783**	**0.798**	**0.790**	0.754	0.758	0.762	0.721	0.723	0.735
	NrDCMH-e	0.765	0.767	0.771	**0.775**	**0.774**	0.779	**0.745**	**0.746**	0.747

Table 12
Ablation study results (NDCG@100)) on MirFlickr-25K under different noise rates *r*.

	noise rate *r*	0			0.2			0.4		
	hash bits *c*	16bit	32bit	64bit	16bit	32bit	64bit	16bit	32bit	64bit
I to T	NrDCMH	0.410	0.415	0.421	0.390	0.391	0.395	**0.371**	**0.375**	0.374
	NrDCMH-n	**0.412**	**0.418**	**0.425**	0.380	0.387	0.391	0.357	0.356	0.357
	NrDCMH-e	0.401	0.408	0.407	**0.391**	**0.395**	**0.397**	0.370	0.373	**0.378**
T to I	NrDCMH	0.450	0.455	0.452	0.431	0.429	0.433	0.405	**0.415**	0.411
	NrDCMH-n	**0.461**	**0.463**	**0.462**	0.424	0.428	0.426	0.396	0.399	0.402
	NrDCMH-e	0.442	0.440	0.443	**0.438**	**0.443**	**0.437**	**0.408**	0.412	**0.416**

Table 13
Training time costs (seconds) of the different methods on four datasets.

Methods	MirFlickr	NUS-wide	Wiki	Reuters
DCMH	32,658	34,872	8,434	31,519
RDCMH	48,362	50,325	10,571	–
SDCH	36,522	37,215	8,729	36,624
NrDCMH	28,451	30,014	6,921	28,695

5. Conclusion

Label noises significantly compromise the effectiveness of supervised machine learning methods, including supervised cross-modal hashing methods. Most existing CMH methods do not take label noises into consideration. We consider the label noise and propose a novel cross-modal hashing method NrDCMH (Noise-robust Deep Cross-Modal Hashing). NrDCMH detects instance pairs with label noise by leveraging the feature similarity and label similarity, and excludes training instance pairs with a high probability of being corrupted for training. NrDCMH achieves a better performance in noise scenarios than other state-of-the-art methods. In the future, we will consider noise-robust cross-modal hashing on datasets with weak pairwise relationship and many tail labels.

CRediT authorship contribution statement

Runmin Wang: Data curation, Methodology, Writing - original draft. **Guoxian Yu:** Supervision, Conceptualization, Methodology, Writing - review & editing. **Hong Zhang:** Project administration, Resources, Writing - review & editing. **Maozu Guo:** Funding acquisition, Formal analysis. **Lizhen Cui:** Resources, Writing - review & editing. **Xiangliang Zhang:** Investigation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the Natural Science Foundation of China (61872300 and 62031003).

References

[1] M.R. Amini, N. Usunier, C. Goutte, Learning from multiple partially observed views -an application to multilingual text categorization, in: Advances in Neural Information Processing Systems, 2009, pp. 28–36.
[2] M.M. Bronstein, A.M. Bronstein, F. Michel, N. Paragios, Data fusion through cross-modality metric learning using similarity-sensitive hashing, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 3594–3601.
[3] Y. Cao, M. Long, J. Wang, Q. Yang, P.S. Yu, Deep visual-semantic hashing for cross-modal retrieval, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1445–1454.
[4] Z. Cao, M. Long, J. Wang, Q. Yang, Transitive hashing network for heterogeneous multimedia retrieval, in: AAAI Conference on Artificial Intelligence, 2017, pp. 81–87.
[5] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, in: British Machine Vision Conference, 2014, pp. 1–12.
[6] S. Chen, Y. Shi, Y. Zhang, J. Zhao, C. Zhang, T. Pei, Local multi-feature hashing based fast matching for aerial images, Information Sciences 442–443 (2018) 173–185.
[7] T. Chua, J. Tang, R. Hong, H. Li, Z. Luo, Y. Zheng, Nus-wide: a real-world web image database from national university of singapore, in: International Conference on Image and Video Retrieval, 2009, p. 48.
[8] G. Ding, Y. Guo, J. Zhou, Collective matrix factorization hashing for multimodal data, in: IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 2075–2082.
[9] A. Ghosh, H. Kumar, P. Sastry, Robust loss functions under label noise for deep neural networks, in: AAAI Conference on Artificial Intelligence, 2017, pp. 1919–1925..
[10] A. Ghosh, N. Manwani, P. Sastry, Making risk minimization tolerant to label noise, Neurocomputing 160 (2015) 93–107.
[11] J. Goldberger, E. Ben-Reuven, Training deep neural-networks using a noise adaptation layer, in: International Conference on Learning Representations, 2017, pp. 1–9.
[12] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in: Advances in Neural Information Processing Systems, 2014, pp. 2672–2680.
[13] B. Han, Q. Yao, X. Yu, G. Niu, M. Xu, W. Hu, I. Tsang, M. Co-teaching Sugiyama, Robust training of deep neural networks with extremely noisy labels, in: Advances in Neural Information Processing Systems, 2018, pp. 8527–8537.
[14] D.R. Hardoon, S. Szedmak, J. Shawe-Taylor, Canonical correlation analysis: An overview with application to learning methods, Neural Computation 16 (12) (2004) 2639–2664.
[15] D. Hendrycks, M. Mazeika, D. Wilson, K. Gimpel, Using trusted data to train deep networks on labels corrupted by severe noise, in: Advances in Neural Information Processing Systems, 2018, pp. 10456–10465.
[16] M.J. Huiskes, M.S. Lew, The mir flickr retrieval evaluation, in: ACM International Conference on Multimedia Information Retrieval, 2008, pp. 39–43.
[17] Z. Ji, Y. Sun, Y. Yu, Y. Pang, J. Han, Attribute-guided network for cross-modal zero-shot hashing, IEEE Transactions on Neural Networks and Learning Systems 31 (1) (2020) 321–330.
[18] L. Jiang, Z. Zhou, T. Leung, L. Li, F. Mentornet Li, Learning data-driven curriculum for very deep neural networks on corrupted labels, in: International Conference on Machine Learning, 2018, pp. 2309–2318.
[19] Q. Jiang, W. Li, Deep cross-modal hashing, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3232–3240.
[20] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, 2012, pp. 1097–1105.
[21] Y. Li, J. Yang, Y. Song, L. Cao, J. Luo, L.J. Li, Learning from noisy labels with distillation, in: IEEE International Conference on Computer Vision, 2017, pp. 1910–1918.
[22] Z. Lin, G. Ding, M. Hu, J. Wang, Semantics-preserving hashing for cross-view retrieval, in: IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3864–3872.
[23] X. Liu, Z. Li, J. Wang, G. Yu, C. Domeniconi, X. Zhang, Cross-modal zero-shot hashing, in: IEEE International Conference on Data Mining, 2019, pp. 449–458.
[24] X. Liu, X. Nie, W. Zeng, C. Cui, L. Zhu, Y. Yin, Fast discrete cross-modal hashing with regressing from semantic labels, in: ACM International Conference on Multimedia, 2018, pp. 1662–1669.
[25] X. Liu, G. Yu, C. Domeniconi, J. Wang, Y. Ren, M. Guo, Ranking-based deep cross-modal hashing, in: AAAI Conference on Artificial Intelligence, 2019, pp. 4400–4407.
[26] X. Liu, G. Yu, C. Domeniconi, J. Wang, G. Xiao, M. Guo, Weakly-supervised cross-modal hashing, IEEE Transactions on Big Data 99 (1) (2019) 1–14.
[27] D.G. Lowe, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision 60 (2) (2004) 91–110.
[28] A. Oliva, A. Torralba, Modeling the shape of the scene: A holistic representation of the spatial envelope, International Journal of Computer Vision 42 (3) (2001) 145–175.
[29] G. Patrini, A. Rozza, A. Krishna Menon, R. Nock, L. Qu, Making deep neural networks robust to label noise: A loss correction approach, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 1944–1952.
[30] N. Rasiwasia, J. Costa Pereira, E. Coviello, G. Doyle, G.R. Lanckriet, R. Levy, N. Vasconcelos, A new approach to cross-modal multimedia retrieval, in: ACM International Conference on Multimedia, 2010, pp. 251–260.
[31] M. Ren, W. Zeng, B. Yang, R. Urtasun, Learning to reweight examples for robust deep learning, in: International Conference on Machine Learning, 2018, pp. 4331–4340.
[32] Y. Shen, Y. Feng, B. Fang, M. Zhou, S. Kwong, B. hua Qiang, Dsrph: Deep semantic-aware ranking preserving hashing for efficient multi-label image retrieval, Information Sciences 539 (2020) 145–156..
[33] S. Su, Z. Zhong, C. Zhang, Deep joint-semantics reconstructing hashing for large-scale unsupervised cross-modal retrieval, 2019, pp. 3027–3035.
[34] S. Sukhbaatar, R. Fergus, Learning from noisy labels with deep neural networks, in: International Conference on Learning Representations, 2014, pp. 1–9.
[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, in: Advance in Neural Information Processing Systems, 2017, pp. 6000–6010.
[36] A. Veit, N. Alldrin, G. Chechik, I. Krasin, A. Gupta, S. Belongie, Learning from noisy large-scale datasets with minimal supervision, in: IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 839–847.
[37] D. Wang, X. Gao, X. Wang, L. He, Semantic topic multimodal hashing for cross-media retrieval, in: International Joint Conference on Artificial Intelligence, 2015, pp. 3890–3896..
[38] W. Wang, H. Zhang, Z. Zhang, L. Liu, L. Shao, Sparse graph based self-supervised hashing for scalable image retrieval, Information Sciences 547 (2021) 622–640.
[39] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: Advances in Neural Information Processing Systems, 2009, pp. 1753–1760.
[40] G. Wu, Z. Lin, J. Han, L. Liu, G. Ding, B. Zhang, J. Shen, Unsupervised deep hashing via binary latent factor models for large-scale cross-modal retrieval, 2018, pp. 2854–2860.
[41] C. Yan, X. Bai, S. Wang, J. Zhou, E.R. Hancock, Cross-modal hashing with semantic deep embedding, Neurocomputing 337 (2019) 58–66.
[42] E. Yang, C. Deng, W. Liu, X. Liu, D. Tao, X. Gao, Pairwise relationship guided deep hashing for cross-modal retrieval, in: AAAI Conference on Artificial Intelligence, 2017, pp. 1618–1625.

[43] K. Yi, J. Wu, Probabilistic end-to-end noise correction for learning with noisy labels, in: IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 7017–7025.
[44] G. Yu, X. Liu, J. Wang, C. Domeniconi, X. Zhang, Flexible cross-modal hashing, IEEE Transactions on Neural Networks and Learning Systems 99 (1) (2021) 1–10.
[45] T. Yu, G. Yu, J. Wang, M. Guo, Partial multi-label learning with label and feature collaboration, in: International Conference on Database Systems for Advanced Applications, 2020, pp. 621–637.
[46] D. Zhang, W. Li, Large-scale supervised multimodal hashing with semantic correlation maximization, in: AAAI Conference on Artificial Intelligence, 2014, pp. 2177–2183.
[47] J. Zhang, Y. Peng, M. Yuan, Sch-gan: Semi-supervised cross-modal hashing by generative adversarial network, IEEE Transactions on Cybernetics 50 (2) (2020) 489–502.
[48] Z. Zhang, M. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in: Advances in Neural Information Processing Systems, 2018, pp. 8778–8788.
[49] Y. Zhen, D.Y. Yeung, A probabilistic model for multimodal hash function learning, in: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2012, pp. 940–948.
[50] J. Zhou, G. Ding, Y. Guo, Latent semantic sparse hashing for cross-modal similarity search, in: ACM SIGIR Conference on Research & Development in Information Retrieval, 2014, pp. 415–424.

Runmin Wang is an MPhil student at the College of Computer and Information Sciences, Southwest University, Chongqing, China. He received his B.Sc. degree in Mathematics and Applied Mathematics from Nankai University, Tianjin, China in 2016. His research interests include data mining and machine learning.

Guoxian Yu is a Professor at the School of Software, Shandong University, Jinan, China. He received the Ph.D. in Computer Science from South China University of Technology, Guangzhou, China in 2013. His current research interests include data mining and bioinformatics. He serves as reviewers for TKDE, TNNLS, TCYB, TCBB, KDD, ICDM, IJCAI and other prestigious journals and conferences.

Hong Zhang is an Associate Professor at the College of Computer and Information Sciences, Southwest University, Chongqing, China. She received her PhD degree from Southwest University, Chongqing, China in 2006. Her research interests include machine learning and its extensive applications.

Maozu Guo is a Professor at the College of Electrical and Information Engineering, Beijing University of Civil Engineering and Architecture, Beijing, China. He received the Ph.D. degree in Computer Science and Technology from Harbin Institute of Technology in 1997. His research interests include bioinformatics, machine learning, and data mining.

Lizhen Cui is a Professor at the School of Software, Shandong University, Jinan, China. He received BSc, MPhil and PhD from Shandong University in 1999, 2002 and 2005 respectively. His current research interests include big data analysis, crowd science and engineering, intelligent medical analysis. He regularly serves as PC for prestigious conferences (i.e., KDD, IJCAI, CIKM and DASFAA) and reviewers for IEEE/ACM Transaction Journals.

Xiangliang Zhang is an Associate Professor at the Department of Computer Science, King Abdullah University of Science and Technology (KAUST), Saudi Arabia. She earned her PhD degree in Computer Science with great honors from INRIA-University Paris-Sud 11, France, in 2010. Her main research interests and experiences are in diverse areas of machine learning and data mining.