Liu *et al. Journal of Big Data* (2024) 11:54
https://doi.org/10.1186/s40537-024-00919-4

Journal of Big Data

METHODOLOGY Open Access

An improved deep hashing model for image retrieval with binary code similarities

Huawen Liu[1*], Zongda Wu[1], Minghao Yin[2], Donghua Yu[1], Xinzhong Zhu[3] and Jungang Lou[4]

*Correspondence:
liu@usx.edu.cn

[1] Department of Computer Science, Shaoxing University, 508 West Huancheng Road, Shaoxing 312000, People's Republic of China
[2] School of Information Science and Technology, Northeast Normal University, Changchun, People's Republic of China
[3] School of Computer Science and Technology, Zhejiang Normal University, Jinhua, People's Republic of China
[4] School of Information Engineering, Huzhou University, Huzhou, People's Republic of China

Abstract

The exponential growth of data raises an unprecedented challenge in data analysis: how to retrieve interesting information from such large-scale data. Hash learning is a promising solution to address this challenge, because it may bring many potential advantages, such as extremely high efficiency and low storage cost, after projecting high-dimensional data to compact binary codes. However, traditional hash learning algorithms often suffer from the problem of semantic inconsistency, where images with similar semantic features may have different binary codes. In this paper, we propose a novel end-to-end deep hashing method based on the similarities of binary codes, dubbed CSDH (Code Similarity-based Deep Hashing), for image retrieval. Specifically, it extracts deep features from images to capture semantic information using a pre-trained deep convolutional neural network. Additionally, a hidden and fully connected layer is attached at the end of the deep network to derive hash bits by virtue of an activation function. To preserve the semantic consistency of images, a loss function has been introduced. It takes the label similarities, as well as the Hamming embedding distances, into consideration. By doing so, CSDH can learn more compact and powerful hash codes, which not only can preserve semantic similarity but also have small Hamming distances between similar images. To verify the effectiveness of CSDH, we evaluate CSDH on two public benchmark image collections, i.e., CIFAR-10 and NUS-WIDE, with five classic shallow hashing models and six popular deep hashing ones. The experimental results show that CSDH can achieve competitive performance to the popular deep hashing algorithms.

Keywords: Hash learning, Big data, Image retrieval, Deep learning

Introduction

The advent of information technology has led to an exponential increase in data accumulation across a multitude of domains [1]. For example, Facebook users upload over a billion images each month, and the daily generation of log files approximates 300 TB. Furthermore, the duration of videos uploaded by YouTube users in the past month surpasses the total video content broadcasted by ABC, NBC, and CBS since 1948. This surge in big data presents a formidable challenge to both academia and industry: how to effectively and efficiently harness such a vast amount of data to extract and analyze valuable information or knowledge for various applications [2]. This challenge has greatly catalyzed an increasing interest in information retrieval, i.e., the development of novel

techniques to explore and analyze big data effectively and efficiently [1, 3]. These techniques have demonstrated potential advantages in assisting businesses and scholars to extract valuable insights from big data across a wide range of real-world applications, from healthcare [4], privacy protection [5], financial analysis [6], image retrieval, to natural language processing [7], among others.

It is important to note that the mere collection and storage of massive amounts of data are not the ultimate objectives. The full potential benefits of big data should be exploited to address specific real-world problems. However, as the scale of data increases exponentially, the expansion of useful information underlying big data is relatively mild. This implies that the density of information or knowledge deeply embedded in big data is extremely low, leading to a scenario characterized by data bloom but knowledge scarcity [8].

The extraction of interesting or valuable information from large-scale data is a challenging yet fundamental task in big data analysis and information retrieval [8]. Given a query, it can be relatively straightforward to precisely identify its proximate or similar objects from a small-scale and low-dimensional data collection using the technique of k nearest neighbors (kNN), which is a classic and popular neighbor search technique in information retrieval. However, as the scale of data increases, locating desirable information precisely using traditional search techniques becomes unfeasible. For instance, the efficiency of kNN dramatically deteriorates even on a mid-scale data collection, albeit its computational complexity is linear. This greatly limits its practical applications. In many scenarios, approximate search methods are preferred over exact search techniques as they offer high efficiency and robustness to large-scale data without significantly compromising retrieval performance.

Hash learning is a representative and popular approximate search technique for big data. It primarily transforms high-dimensional data into binary representations via projection technology [8]. In the context of binary representations, the storage cost of big data can be significantly reduced, making it possible to store big data within the main memory. Moreover, the objective of neighbor search can be achieved by computing the distance or similarity between binary codes using bit operations such as `XOR` and `POPCNT`, thereby making the search process extremely fast. Owing to these advantages, hash learning has attracted increasing attention in various domains, including image retrieval, information retrieval, and natural language processing [9]. Over the past decades, numerous hashing methods have been developed. Generally, they can be categorized into two types: data-oblivious (also known as data-independent) and data-aware (also known as data-dependent) hashing techniques [9].

Data-oblivious hashing techniques project data objects from the Euclidean space into binary representations in the Hamming space in a straightforward manner, while data-aware hashing techniques derive binary representations based on the inherent properties of data objects [9]. Notable examples of this kind of techniques include LSH (Locality-Sensitive Hashing) [10] and ITQ (ITerative Quantization) [11], respectively. It is noticeable that both of them construct retrieval models based on handcrafted features, constraining their retrieval performance greatly.

In contrast, deep hashing leverages semantic features of data to construct retrieval models using deep neural networks, such as CNN, AlexNet, ResNet, and BERT [12, 13].

Deep neural networks are known to effectively extract high-level and rich semantic features from data in an unsupervised manner [14, 15]. For instance, Fig. 1 visualizes the feature map (i.e., the output) of the fifth layer of a convolution neural network. It is evident that the extracted features effectively capture sketch information of images. This suggests that they contain rich semantic information, which cannot be fully captured by those handcrafted ones [16]. Owing to this fact, deep hashing usually outperforms conventional hashing models and can derive more compact binary codes [17].

Deep hashing has emerged as a new trend, and a variety of deep hashing techniques have been developed [13]. Prominent deep hashing methods include CNNH (Convolutional Neural Network Hashing) [18], DPSH (Deep Pairwise Supervised Hashing) [19] and DHN (Deep Hashing Network) [20]. For instance, UDQH-IQ (Unsupervised Deep Quadruplet Hashing with Isometric Quantization) [21] uses the quadruplet-based loss as the input of a deep network to explore the underlying semantic similarity of images and employs Hamming-isometric quantization to maximize the consistency of semantic similarity. SPL-UDH (Soft-Pseudo-Label-based Unsupervised Deep Hashing) [22] utilizes an auto-encoder to derive soft pseudo-labels and local similarities of images simultaneously. Based on these, inter- and intra-cluster similarities of images can be further learned by a deep hashing network. Despite the

Fig. 1 The visualization of feature map output by the fifth layer of convolution neural network

popularity of deep hashing, there are limitations that require further exploration and more endeavors. For example, early deep hashing techniques separate deep feature learning and binary code generation. Moreover, unsupervised deep hashing does not fully consider semantic information. Although supervised ones take label and semantic similarity information into consideration when generating binary codes, they do not consider semantic structures in Hamming space [17].

In this work, we propose a novel end-to-end deep hashing model for image retrieval with binary code similarities, dubbed CSDH, to address the problems above. It extracts deep features to capture semantic structural information of images using a pre-trained deep convolution neural network (CNN). To generate binary codes, a hidden and fully connected layer is attached at the end of the deep network. This hidden layer is used to construct hash functions and is referred to as a hash layer, where the activation status of each unit is served as a hash bit. Unlike deep hashing models that directly use a classification layer to generate binary codes, our hashing model with the hash layer can better preserve the semantic similarities of images because the hash layer can capture high-level semantic information related to labels. Furthermore, to enhance the consistency of similarity preservation, the embedding distances of images in Hamming space are incorporated into the loss function. This embedding property ensures that the generated binary codes are of higher quality and possess greater capability.

To sum up, the main contributions of this work are briefly highlighted as follows:

- The proposed model, CSDH, is an end-to-end deep hashing model for image retrieval that uses a pre-trained deep convolution neural network (CNN) to extract deep features and capture semantic structural information of images.
- A hidden and fully connected layer, referred to as a hash layer, is attached at the end of the deep network to generate binary codes. This approach better preserves the semantic similarities of images compared to models that directly use a classification layer; that is, the images with more similar codes have a higher probability of becoming the same category.
- The model incorporates the embedding distances of images in Hamming space into the loss function to enhance the consistency of similarity preservation, resulting in higher quality binary codes with greater capability.

The remainder of the paper is organized as follows. Section briefly discusses related studies about hash learning. Section presents the proposed hashing method for image retrieval in detail, followed by the experimental results and discussions in Sect. . Finally, the conclusion and future studies are given in Sect. .

Related work

Hash learning, with its many potential advantages including high efficiency and low storage cost, has quickly become a leading technique in image retrieval and big data analysis. A multitude of hash learning algorithms have been witnessed to date. These hash learning techniques can be grouped into different categories, such as supervised

and unsupervised hashing, data-oblivious and data-aware (also known as data-independent and data-dependent, respectively) hashing, based on their properties or perspectives [13, 17].

Depending on which type of features has been adopted, hashing models can also be classified into shallow hashing and deep hashing in a board way. Shallow hashing techniques rely solely on handcrafted features for the training of hashing models and the generation of binary codes. One of the most notable examples of this is Locality-Sensitive Hashing (LSH), which encodes data into a binary representation via random projections [10]. LSH is extremely efficient, but the retrieved neighbors are random to some extent and may not exact neighbors. To this end, Iterative Quantization (ITQ) generates binary codes predicated on the principal components of data [11], while Spectral Hashing (SH) seeks the eigenvectors of graph Laplacian to achieve data projection [23]. Unfortunately, the similarity structure of data may not be preserved in ITQ, and the optimization problem of SH is NP-hard one. Kernelized Supervised Hashing (KSH) employs kernel functions to handle non-linear data when designing hash function design [24]. Although it requires less supervised information, its performance heavily relies on kernel functions, incurring some cumbersome model training. Fast Supervised Hashing (FastH) utilizes boosting trees, a simple yet effective regression of the class labels, to tackle the high-dimensional problem [25], and Robust Supervised Discrete Hashing (RSDH) employs the Cauchy loss to measure the error of label matrix decomposition, thereby enhancing model robustness [26]. However, large quantization errors and suboptimal solutions may be inevitably induced when relaxing the discrete constraint on hash codes. Moreover, as previously discussed, the shallow hashing algorithms concern the data with handcrafted features, which are designed for specific tasks during the process of collecting data, without involving strongly semantic information. Thus, the performance of constructed hashing models is limited, albeit some of them take label information into account.

In contrast, deep hashing leverages deep features extracted by deep neural networks, such as CNN, VGG, AlexNet, ResNet, and BERT, to construct hashing models and generate binary codes [12, 13]. These deep features encapsulate rich semantic and structural information, exhibiting strong discriminative capabilities. As a result, deep hashing has been extensively studied and widely used in image retrieval. Representative examples of deep hashing include CNNH [18], DPSH [19], DHN [20] and DQN (Deep Quantization Network) [27]. Despite that these deep hashing models have competitive performance, they still have some limitations required to be addressed. For example, CNNH can not handle those images with different scales and positions. DPSH requires a large amount of labeled data to train hashing models, while the optimal policy of DQN is non-deterministic.

Recently, HashSIM [28] guides the generation of binary codes by using semantically structural similarities derived from highly confident images. Besides, the independent property of hash bits has also been considered. However, it likely requires a great number of highly confident images, which are difficult to obtain in reality. Cui et al. [29] first extracted binarized representation embeddings of data via metric learning, then constructed a hashing model using a group similarity preservation strategy. A limitation of this hashing model is that when the length of binary codes is extremely short, the discriminability of deep representations is relatively poor. DUDH (Deep Uncoupled

Discrete Hashing) [30] adopts a similarity-transfer matrix to bridge the gap between query and image similarities, thereby reducing quantization error and preserving image semantic similarities. However, only the process of preserving similarity takes the quantization error into account, resulting in the limited improvement of retrieval performance. It should be pointed out that the unsupervised hashing models above, despite enhancing retrieval performance to some extent, have not considered label information.

Supervised deep hashing methodologies, which leverage both deep features and expert-provided labels as semantic information to help the generation of binary codes, have been demonstrated to outperform their unsupervised counterparts in retrieval performance, thereby garnering significant interest in the field of image retrieval. Noteworthy supervised hashing algorithms include DSHTL (Deep Supervised Hashing with Triplet Labels) [31] and DSDH (Deep Discrete Supervised Hashing) [32], where DSHTL maximizes the likelihood of triplet similarities of labels to learn deep features and binary codes simultaneously. However, obtaining the triplet similarities of labels is non-trivial. For DSDH, it harnesses pairwise supervised information to directly extract deep features and promote the generation of discrete codes.

DDH-LDL (Deep Discrete Hashing for Label Distribution Learning) [33] incorporates label distribution learning to model implicit semantic relationships, which were subsequently preserved through message aggregation operations on a graph convolutional network. As we know, the feature distribution may be inevitably distorted by the aggregation operations, leading to retrieval performance decline. DAHP (Deep Attention-Guided Hashing With Pairwise Labels) [34] utilized anchors as supervised information to extract the contextual information of features, thereby enhancing the representational capacity of the hashing model constructed on the ResNet with position and channel attention mechanisms. Although the attention-guided hash codes can instruct the training of hashing network, they may contain repetitive and highly correlated information. Hu et al. [35] employed the cosine similarities of images to preserve semantic distributions and utilized cosine-distance entropy to mitigate quantization errors for imbalanced data. However, local features had not been considered when learning the contextual information. Much more recently, SPL-UDH (Soft-Pseudo-Label-based Unsupervised Deep Hashing) [36] obtains binary representations by performing Bayesian theory on local similarities and soft pseudo-labels, which are derived by a deep auto-encoder network. PLDH (Pseudo-Labels Deep Hashing) [37] also exploits pseudo-labels extracted by a deep neural network to guide the generation of hash codes. It is worth noting that pseudo-labels are not really ones and may contains inconsistent semantic information. Moreover, existing supervised algorithms primarily focus on the semantic information in Euclidean space, neglecting those in Hamming space.

Methodology

Problem statement

Assume that $\boldsymbol{x} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^{n}$ is an image (or data) collection comprising of n images (or data objects), where $\boldsymbol{x}_i \in \mathcal{R}^d$ (i=1..n) is the i-th image, represented as a vector of d dimensions. The vector $\boldsymbol{y}_i \in \{0,1\}^l$ refers to the label information corresponding to the i-th image $\boldsymbol{x}_i$, where l is the number of labels. $\boldsymbol{X}$ is called a single-label or normal collection

if there is only one label marked to $\boldsymbol{x}_i$ (i=1..n); that is, for each label vector $\boldsymbol{y}_i$ of $\boldsymbol{x}_i$, we have $\Sigma_{j=1}^{l} y_{ij} = 1$. Otherwise, $\boldsymbol{X}$ is a multi-label collection for supervised learning.

Let $\boldsymbol{h}(\boldsymbol{X})$ be a hash function of $\boldsymbol{X}$. Mathematically, it is defined as follows:

$$\boldsymbol{h} : \boldsymbol{X} \mapsto b \in \{0, 1\}, \tag{1}$$

where b is a binary value. From the definition, we know that the hash function $\boldsymbol{h}(\boldsymbol{X})$ encodes the image $\boldsymbol{X}$ into a binary value, i.e., 0 or 1, which is also called hash bit in the literature. If we have m hash functions $\boldsymbol{h}_i$ ($i = 1..m$) and perform them on $\boldsymbol{X}$, we can receive m binary values (i.e., hash bits) b_i ($i = 1..m$). In this case, the image $\boldsymbol{X}$ can be transformed to a vector of binary representations $\boldsymbol{b} = [b_1, b_2, .., b_m] \in \{0, 1\}^m$, as b_i is assembled together directly.

Hash learning aims to construct a variety of hash functions $\boldsymbol{H} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, .., \boldsymbol{h}_m\}$, so that each image $\boldsymbol{X}$ can be represented as a binary vector $\boldsymbol{b} = [b_1, b_2, .., b_m] \in \{0, 1\}^m$. Under this context, the image collection $\boldsymbol{X}$ is transformed into the binary representations $\boldsymbol{b} = \boldsymbol{H}(\boldsymbol{X})$, i.e., $\boldsymbol{b} = \{\boldsymbol{b}_i\}_{i=1}^{n} \in \{0, 1\}^{n \times m}$, where $\boldsymbol{b}_i$ is the binary representation of $\boldsymbol{x}_i$, after the hash functions $\boldsymbol{H}$ are performed on $\boldsymbol{X}$. Generally, the number of binary values is far less than the quantities of image dimensions, i.e., $m \ll d$. From this perspective, hash learning can effectively benefit big data analysis in the aspects of storage cost and computational efficiency. For the sake of discussion, hereafter the binary values are represented as -1 and 1, rather than 0 and 1; that is, $\boldsymbol{b} \in \{-1, 1\}^{n \times m}$.

Generally, the objective function of hash learning is formally represented as follows.

$$\ell_{\boldsymbol{H}}(\boldsymbol{X}, \boldsymbol{b}) = \sum_{i=1}^{n} \sum_{j=1}^{m} \ell_{\boldsymbol{h}_j}(\boldsymbol{x}_i, b_i), \tag{2}$$

where $b = \boldsymbol{h}(\boldsymbol{X})$ and $\ell_{\boldsymbol{h}}(\boldsymbol{X}, b)$ is the quantization error of $\boldsymbol{X}$ after the hash function $\boldsymbol{h}$ exerted. This definition implies that the error should be minimized when learning the hash functions $\boldsymbol{H}$; that is, the information loss should be less as much as possible.

Model architecture

It is still a formidable challenge to learn and construct effective hashing functions. A multitude of techniques for constructing hash function have been proposed, including Locality-Sensitive Hashing (LSH), which employs projection techniques to randomly generate $\boldsymbol{H}$, and Iterative Quantization (ITQ), which utilizes the principal components of $\boldsymbol{X}$ as $\boldsymbol{H}$. However, these shallow hashing methods do not take into account the inherent properties of the data, resulting in the relatively poor quality of binary codes generated by them. Furthermore, these methods rely solely on handcrafted features, which contain limited semantic information, making their retrieval performance less competitive. In contrast, deep hashing techniques extract deep features from data, encapsulating rich semantic and structural information for hash function generation. Consequently, deep hashing methods typically outperform their shallow counterparts.

In this work, we introduce a novel end-to-end deep hashing method, termed Code Similarity-based Deep Hashing (CSDH), for image retrieval. CSDH employs a tailored deep convolutional neural network to represent images and extract deep features for binary code generation. The specific framework of CSDH, illustrated in Fig. 2, serves two

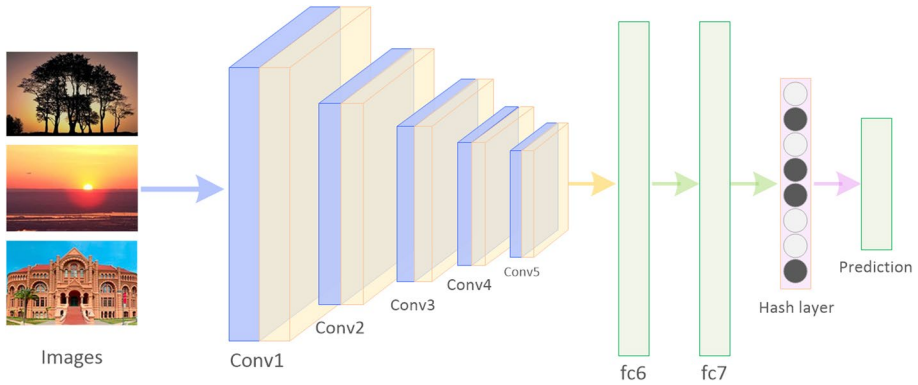


Fig. 2 The deep hashing model framework of CSDH

primary objectives: data representation and similarity preservation. The former extracts high-level semantic features from images, while the latter ensures that the semantic similarities of images can be preserved when generating corresponding binary codes. In other words, the embedded similarities of images should be preserved and consistent when transitioning from Euclidean space to Hamming space.

Specifically, we utilize a tailored AlexNet architecture, a renowned eight-layer Convolutional Neural Network (CNN), as the backbone of CSDH to extract high-level semantic features from images. Due to its superior performance, the popularity of AlexNet is quickly increasing since it has been introduced. Indeed, the features captured by the AlexNet network contain significantly more semantic information than manually-designed features. Traditionally, the architecture of AlexNet comprises five convolutional layers and two fully connected layers, along with one prediction layer. In this work, we use the AlexNet network to represent images and extract their deep features for discrimination.

To achieve the purpose of hashing, we further customize AlexNet by supplying a hidden hash layer between the second fully connected layer and the prediction layer. This hash layer takes the output of the second fully connected layer as input and outputs binary values by transforming continuous values into binary ones via a given quantization function. The hash layer contains k units, where k refers to the length of the desired binary codes. For each unit within the hash layer, it is assumed to be associated with a latent attribute, which will be used to determine the ultimate category of images. If a unit is activated, its output is 1; otherwise, its status is -1. Consequently, each image can be represented as a binary code with k bits based on the activated status of the k units. For two similar images, we expect their binary codes to be similar or proximate after the binary representation or activation operations are performed; conversely, if they are dissimilar, their corresponding codes should also be dissimilar or far from each other in Hamming space. With this kind of similarity preservation, the probability of belonging to the same category of two images becomes higher if their binary codes are similar.

It is noticeable that information will be lost inevitably during the quantization process. To mitigate quantization errors and preserve similarities, selecting an appropriate activation function for the hash layer plays a crucial role. By now, a plethora of activation functions, such as Tanh, Sigmoid, ReLU, Softmax, Swish, among others, have been

proposed and are ready-made in the literature [38]. Considering gradient vanishing and smooth properties, we take the `softsign` function, whose definition is given as follows [38], as the activation function for the hash layer.

$$Softsign(x) = \frac{x}{1+|x|}, \tag{3}$$

where x is a continuous value.

Loss function

The final layer of AlexNet is utilized to predict the category to which an image belongs via a loss function. From this perspective, loss functions play a pivotal role in deep learning as they directly influence the semantic information of deep features and the prediction performance of deep neural networks. To generate high-quality of binary codes, here we adopt a cross-entropy loss, in conjunction with a Hamming-embedding loss, for the prediction layer of the customized AlexNet network.

The cross-entropy loss quantifies the divergence degree between two probability distributions. Given two distinct probability distributions of a feature (a.k.a. variable) X, denoted as $p(x)$ and $q(x)$, their cross-entropy $H(p, q)$ is formally defined as:

$$H(p,q) = -\sum_{x\in X} p(x)\log q(x). \tag{4}$$

From this definition, it can be inferred that if two distributions are similar or proximate, their cross-entropy is small. Based on this rationale, we incorporate this concept into our loss function to predict the category labels of images. Specifically, let $\boldsymbol{X}$ be a normal image collection, where each image is tagged with a single label. For the i-th image $\boldsymbol{x}_i \in \boldsymbol{x}$, the output of prediction layer is represented as:

$$\boldsymbol{o}_i = \boldsymbol{W}\boldsymbol{b}_i + \boldsymbol{v}, \tag{5}$$

where $\boldsymbol{W} \in \mathcal{R}^{l\times k}$ is the weight matrix of the prediction layer, $\boldsymbol{b}_i$ is the output of hash layer for $\boldsymbol{x}_i$. $\boldsymbol{v} \in \mathcal{R}^l$ is the bias of prediction. Thus, the cross-entropy loss of the neural network can be summarized as follows.

$$\ell_E(\boldsymbol{X}) = -\frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{\exp(\boldsymbol{o}_{i\boldsymbol{y}_i})}{\sum_{c=1}^{l}\exp(\boldsymbol{o}_{ic})}\right). \tag{6}$$

In a similar vein, if $\boldsymbol{X}$ is a multi-label image collection, where each image can be associated to multiple labels, we can treat it as l independent binary classifiers. In this case, the cross-entropy loss can be represented as

$$\ell_E(\boldsymbol{X}) = -\frac{1}{nl}\sum_{i=1}^{n}\sum_{j=1}^{l}\boldsymbol{y}_{ij}\log\frac{1}{1+\exp(-\boldsymbol{o}_{ij})} + (1-\boldsymbol{y}_{ij})\log\left(1-\frac{1}{1+\exp(-\boldsymbol{o}_{ij})}\right). \tag{7}$$

As the cross-entropy loss only concerns the prediction performance of deep features, it alone cannot guarantee the quality of binary codes generated by the hashing model. As discussed earlier, the property of similarity-preservation is also very crucial for the

generation of binary codes. To achieve this purpose, we also take Hamming embedding into consideration when constructing the hashing model.

Assume $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$ are two binary codes derived from the hash layer for $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ images, respectively. The Hamming distance between them is

$$\begin{aligned}(b)dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j) &= \frac{1}{2}(k - \boldsymbol{b}_i^T \boldsymbol{b}_j) \\ &= \frac{k}{2}(1 - cos(\boldsymbol{b}_i, \boldsymbol{b}_j)),\end{aligned} \tag{8}$$

where k is the length of binary codes. According to the definition, the Hamming distance is an inverse proportion of cosine value. This property can be exploited to measure the similarity of binary codes. In Hamming space, two binary codes, $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, are considered to be semantically similar, if $dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j)$ is small enough; Otherwise, they are semantically dissimilar to each other. Let s_{ij} be a semantically label between $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$, where $s_{ij} = 1$ when $\boldsymbol{b}_i$ is semantically similar to $\boldsymbol{b}_j$; Otherwise, $s_{ij} = 0$. Under this context, this kind of Hamming embedding can also be used to estimate the quality of binary codes.

Suppose that $\mathcal{S} \in \{0, 1\}^{n \times n}$ is the semantically pairwise similarity of binary codes, each entry $s_{ij} \in \mathcal{S}$ denotes the semantic label between $\boldsymbol{b}_i$ and $\boldsymbol{b}_j$. The Hamming embedding loss refers to the total summary of pairwise Hamming distances, i.e.,

$$\ell_H(\boldsymbol{X}) = \frac{1}{|\mathcal{S}|} \sum_{s_{ij} \in \mathcal{S}} \mathcal{H}(\boldsymbol{b}_i, \boldsymbol{b}_j), \tag{9}$$

where $|\mathcal{S}|$ denotes the total number of semantic similarity labels. $\mathcal{H}(\boldsymbol{b}_i, \boldsymbol{b}_j)$ is the Hamming embedded distance shown as follows.

$$\mathcal{H}(\boldsymbol{b}_i, \boldsymbol{b}_j) = \begin{cases} dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j), & s_{ij} = 1 \\ \max(0, 1 - dist_H(\boldsymbol{b}_i, \boldsymbol{b}_j)). & otherwise \end{cases} \tag{10}$$

According to Eq. (3), we know that the output of the hash layer is a vector of continuous values. Thus, the Hamming embedded distance above can be represented as

$$\mathcal{H}(\boldsymbol{h}_i, \boldsymbol{h}_j) = \begin{cases} dist_H(\boldsymbol{h}_i, \boldsymbol{h}_j), & s_{ij} = 1 \\ \max(0, 1 - dist_H(\boldsymbol{h}_i, \boldsymbol{h}_j)). & otherwise \end{cases} \tag{11}$$

where $\boldsymbol{h}_i$ is the output vector of hash layer for the i-th image $\boldsymbol{x}_i$.

Based on the Hamming embedded loss, the customized AlexNet network can be iteratively updated by the technique of gradient descent. For $dist_H(\boldsymbol{h}_i, \boldsymbol{h}_j)$, its gradient can be easily estimated. When $s_{ij} = 1$, its gradient is

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{h}_i} = -\frac{1}{|\mathcal{S}|} \sum_{s_{ij} \in \mathcal{S}} \frac{1}{\|\boldsymbol{h}_i\|} \left(\frac{\boldsymbol{h}_j}{\left\|\boldsymbol{h}_j\right\|} - \cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \frac{\boldsymbol{h}_i}{\|\boldsymbol{h}_i\|} \right). \tag{12}$$

when $\boldsymbol{h}_i$ is not semantically similar to $\boldsymbol{h}_j$, i.e., $s_{ij} = 0$, the gradient of $\mathcal{H}(\boldsymbol{h}_i, \boldsymbol{h}_j)$ is

$$\frac{\partial \mathcal{H}}{\partial \boldsymbol{h}_i} = \begin{cases} \frac{1}{|\mathcal{S}|} \sum_{s_{ij} \in \mathcal{S}} \frac{1}{\|h_i\|} \left(\frac{h_j}{\|h_j\|} - \cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \frac{h_i}{\|h_i\|} \right), & \cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) > 0 \\ 0. & \cos\left(\boldsymbol{h}_i, \boldsymbol{h}_j\right) \leq 0 \end{cases} \tag{13}$$

Algorithm details

In summary, the objective function of our deep hashing model is to minimize the following loss function

$$\min_{\boldsymbol{\Theta}} \ell_E(\boldsymbol{X}) + \gamma \ell_H(\boldsymbol{X}), \tag{14}$$

where $\boldsymbol{\Theta}$ denotes hyper-parameters of the deep neural network. γ is a trade-off factor to make a balance between the cross-entropy loss and the Hamming embedded loss.

Based on the statement above, the implementation details of deep hashing model with binary code similarities is given as follows.

Algorithm 1 Binary code similariy based deep hashing

Training phase:
Input: A training collection $\boldsymbol{X} = \{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^n$, and the code length k.
Output: The hyper-parameters $\boldsymbol{\Theta}$ of the deep neural network.
1: Get initial parameters with T epochs and batch size m by using a pre-training AlexNet;
2: **For** $epoch = 1, 2, .., T$
3: **For** $iter = 1, 2, .., |\boldsymbol{X}|/m$
4: Randomly select a mini-batch training data $\{\boldsymbol{x}_i\}_{i=1}^m$;
5: Obtain $\{\boldsymbol{o}_i\}$ by the forward-feedback propagation strategy;
6: Calculate the cross-entropy loss $\ell_E(\boldsymbol{x}_i)$ via Eq. (6) or Eq. (7);
7: Calculate the Hamming embedded loss $\ell_H(\boldsymbol{x}_i)$ via Eq. (9);
8: Update the hyper-parameters of deep model by the back-propagation strategy;
9: **End**
10:**End**
Query phase:
Input: Query images $\boldsymbol{X}_q$ and an image collection $\boldsymbol{X}_d$.
Output: Binary codes $\boldsymbol{B}_q$ of $\boldsymbol{X}_q$ and $\boldsymbol{B}_d$ of $\boldsymbol{X}_d$.
1: Get the output $\{\boldsymbol{h}_i\}_{i=q,d}$ of the hash layer of the deep hashing model above;
2: Generate the binary codes $\boldsymbol{b}_i = sgn(\boldsymbol{h}_i)$ of $\{\boldsymbol{h}_i\}_{i=q,d}$.

Experimental results and discussion

To evaluate the competitiveness of CSDH, we conducted a series of comparative experiments with five classical hashing algorithms and six popular deep hashing algorithms on two public image collections. This section elucidates the experimental results and discussions.

Experimental settings

Two frequently-used benchmark image datasets, CIFAR-10 and NUS-WIDE, were employed to evaluate the performance of CSDH against the baseline algorithms. The CIFAR-10 dataset comprises 60,000 colorful images, each of size 32 × 32 pixels, distributed evenly across ten classes. These classes involve various animals (e.g., bird, dog, cat, deer, bird, horse and frog) and public transport vehicles (like truck, car, airplane and ship). Each class contains 6000 images. As CIFAR-10 is a single-label dataset, each image is associated with only a single label. For our experiments, we

randomly selected 100 images per class as query images and 500 images per class for training. Consequently, the training dataset comprised 5000 colorful images while the query dataset contained 1000 images.

NUS-WIDE is a multi-label image dataset where each image is simultaneously associated with multiple labels. It consists of 269,648 color images tagged with eighty-one class labels, such as dog, bird, car, and so on, totally. These images were collected from Flickr. Following conventional practices in the literature, we selected 195,834 images tagged with the top-21 labels from the eighty-one ground-truth labels for our experiments. Each class contained at least 5000 images. We adopted the similar strategy to generate query and training data as with CIFAR-10: 100 images were randomly selected as queries and 500 images were used as training data for each class label. As a result, the query dataset contained 2100 images while the database included 193,734 images of which 10,500 were designated as training data.

For a fair comparison, we considered two kinds of hashing techniques: shallow hashing and deep hashing as baselines in our experiments. The shallow hashing algorithms included ITQ [11], SH [23], KSH [24], FastH [25] and RSDH [26]. As stated above, SH and ITQ are unsupervised hashing algorithms while KSH, FastH and RSDH are supervised ones. The deep hashing methods included CNNH [18], DPSH [19], DHN [20], DQN [27], DSHTL [31], DSDH [32], PLDH [37], DDH-LDL [33] and DAHP [34]. It should be pointed out that DPSH, DSDH and DSHTL employed VGG-F as their backbone of convolution neural network. VGG-F is structurally similar to AlexNet comprising five convolution layers and two fully-connected layers. For fairness in comparison, the shallow hashing models were constructed on deep features where the CIFAR-10 collection was represented by 512 GIST features. For NUS-WIDE collection, each image was represented as 1134 low-level features including color histograms (64), color correlation (144), edge histograms (73), wavelet textures (128), color blocks (225) and SIFT features (500). In contrast, the deep models were trained directly on the original images.

Following traditional strategies in comparative experiments for hash learning, we considered image similarities as the ground truth in the following manner: Two single-label images were deemed to be similar if they were tagged with the same label; otherwise they were considered dissimilar. For multi-label images, they were considered similar if they shared at least one class label; otherwise they were dissimilar to each other.

Three frequently-used evaluation protocols were adopted to testify retrieval performance of the hashing models in the experiments. They were mean average precision (mAP), precision and precision-recall curve. Let $\boldsymbol{Q} = \{q_i\}_{i=1}^{t}$ be a query collection. For each query $q \in \boldsymbol{Q}$, its retrieval precision refers to the ratio of the quantity of similar images to the total number of retrieval results, i.e.,

$$Pre_q(R) = \frac{\sum_{r=1}^{R} \delta(\ell_r = \ell_q)}{R}, \tag{15}$$

where R is the total number of retrieval results, and ℓ_r denotes the label of the r-th retrieval result. $\delta(\cdot)$ is an indication function. $\delta(\ell_r{=}\ell_q){=}1$ if the r-th retrieval result has the same label to the query q. On the contrary, $\delta(\ell_r{=}\ell_q){=}0$, if there is no same label between them. In our experiments, we retrieved 5,000 images for each query; that is,

$R = 5000$. Based on the above formula, the mean average precision of the query set $\boldsymbol{Q}$ is formally represented as

$$mAP(\boldsymbol{Q}) = \frac{1}{|\boldsymbol{Q}|}\sum_{q\in\boldsymbol{Q}}\frac{\sum_{r=1}^{R} Pre_q(r)\delta(\ell_r = \ell_q)}{\sum_{r=1}^{R}\delta(\ell_r = \ell_q)}. \tag{16}$$

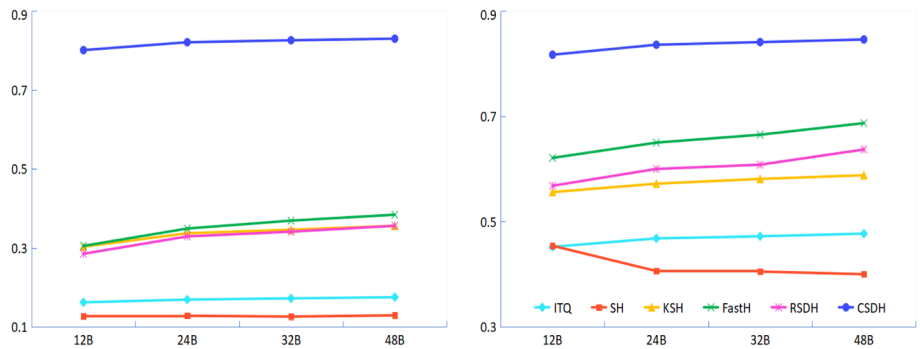
The Precision-Recall curve delineates the interplay between precision and recall, two pivotal metrics in information retrieval. Precision quantifies the relevance of retrieved results, while recall measures the proportion of truly relevant results that are successfully retrieved. An optimal retrieval model is characterized by high precision, ensuring the accuracy of results, and high recall, guaranteeing the retrieval of a substantial fraction of positive results. The Precision-Recall curve, therefore, serves as a critical tool for evaluating the performance of retrieval models.

We implemented the CSDH model by using PyTorch, an open-source machine learning framework. For the hyperparameters of the deep neural network, they were meticulously fine-tuned in a back-propagation way. Specifically, we utilized the mini-batch stochastic gradient descent as the optimizer for CSDH. Throughout the experimental process, we set the size of batch, weight decay, and momentum of the optimizer to 32, 0.0005, and 0.9, respectively. Meanwhile, the learning rate was initially assigned to 0.001 and subsequently decayed a time after 40 training epochs.

Results and discussion

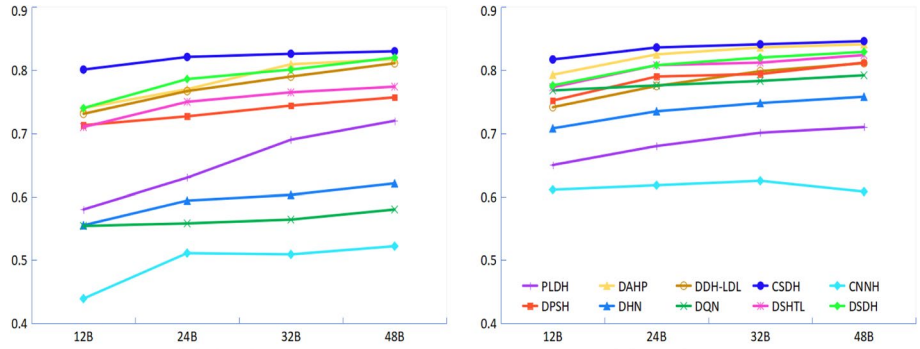
As we know, the mean Average Precision (mAP) is one of the most widely-used metric for evaluating retrieval performance of hashing models. In line with this convention, here we also adopted the mAP metric to make a comparison of the retrieval performance between CSDH and the baselines. Figures 3 and 4 present the comparison scores of mAP for 5000 query results returned by the shallow baseline models and the deep baseline models with varying quantities of hash bits, respectively.

From the experimental results presented in Figs. 3, 4, we can easily conclude that the proposed hashing method exhibits competitive performance compared to the baseline models. For example, when compared to FastH, CSDH boosted the retrieval performance of mAP on the CIFAR-10 and NUS-WIDE collections by 46.77% and 17.92%, respectively. In a similar vein, the mAP scores of CSDH were higher than



(a) CIFAR-10 (b) NUS-WIDE

Fig. 3 The mAP comparison of CSDH to the shallow hashing algorithms with different quantities of hash bits



(a) CIFAR-10 (b) NUS-WIDE

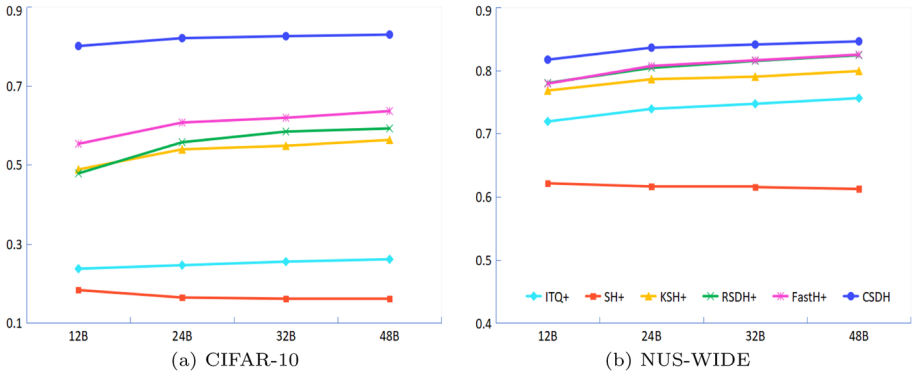
Fig. 4 The mAP comparison of CSDH to the deep hashing algorithms with different quantities of hash bits

those of DSDH, a state-of-the-art deep hashing model, by 3.27% and 2.67% on these two image collections, respectively. Another interesting fact is that the mAP scores of supervised hashing algorithms, e.g., RSDH, KSH and FastH, were significantly higher than those of unsupervised hashing ones, e.g., SH and ITQ. This can be attributed to the fact that the class labels embody a kind of semantic information that aids hashing models in generating informative hash bits.

Note that the deep hashing models were generally superior to the shallow ones, particularly on the single-label data. It sounds reasonable because the deep techniques leveraged high-level semantic features to train hashing models. Among the deep models, DSDH, DAHP, DSHTL and DPSH achieved comparable performance to CNNH, DHN, PLDH and DQN. This can be attributed to their use of data or label similarities to guide binary code generation. Since our hashing model, CSDH, exploited the cross-entropy information, as well as semantic similarities, within the loss function, it achieved better retrieval performance than other models, especially when fewer hash bits (e.g., 12 or 24 bits) were used. Indeed, the cross-entropy loss has been shown to effectively capture rich semantic information in the literature.

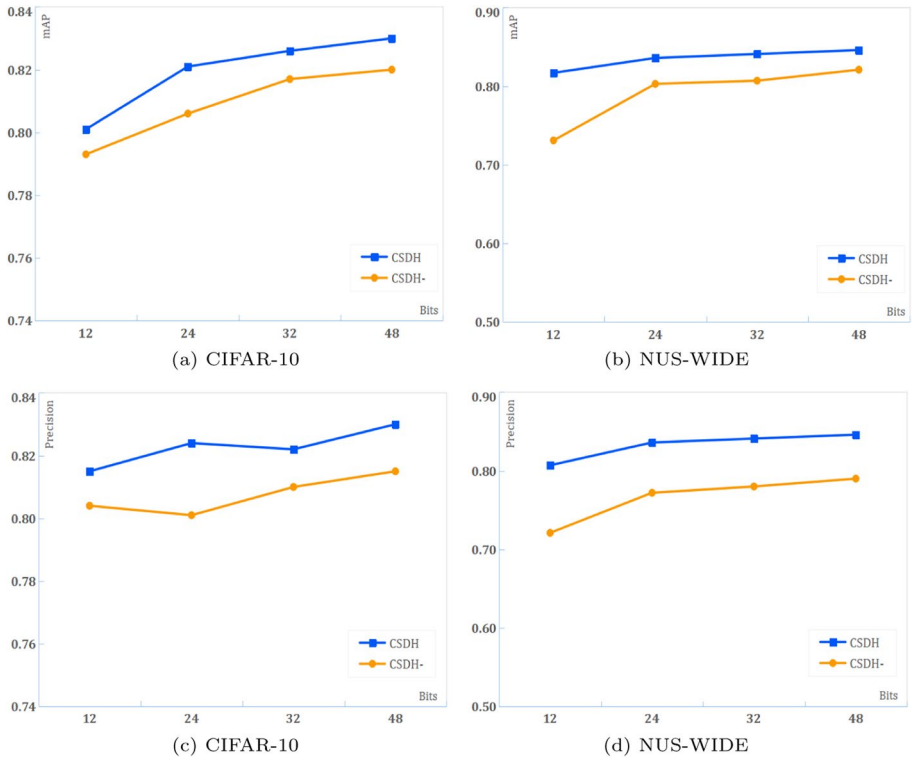
As stated above, convolution neural network can extract deep features to capture semantic information, thereby enhancing model performance. To validate this assertion, we carried out additional experiments using the shallow hashing algorithms with deep features on the image collections. Specifically, we extracted deep features with 4096 dimensions from images using VGG-F, i.e., the output of the last layer of VGG-F. Then the shallow hashing algorithms were performed with these deep features to generate hash bits. The experimental results are provided in Fig. 5.

According to the mAP scores in Fig. 5, one can observe that the deep features could significantly strength the retrieval performance of the shallow hashing models. Broadly speaking, the shallow hashing algorithms with deep features significantly outperformed those without deep features. For example, the mAP score of FastH increased from 0.305 to 0.553 on CIFAR-10 when the number of hash bits was 12. Particularly on multi-label collections like NUS-WIDE, the performance of the shallow hashing algorithms was comparable to that of the deep hashing ones (see Fig. 4).



(a) CIFAR-10 (b) NUS-WIDE

Fig. 5 The mAP comparison of CSDH to the shallow hashing methods with deep features by VGG-F



(a) CIFAR-10 (b) NUS-WIDE

(c) CIFAR-10 (d) NUS-WIDE

Fig. 6 The performance of our model with (or without) the Hamming embedding loss, denoted as CSDH and CSDH-, respectively

Ablation analysis

The aforementioned discussions show that the image similarity can bring benefits to the performance of hashing models, as evidenced by DSDH, DAHP, DSHTL and DPSH. Unlike those deep hashing models only with label similarities, our model extends beyond by integrating Hamming embedding distances into the loss function. To testify the contribution of Hamming embedding distances to retrieval performance, we conducted additional experiments on image collections using our proposed model with and without Hamming embedding distances, denoted as CSDH and CSDH-, respectively.

Figure 6 illustrates the mAP and precision of the proposed hashing model with different quantities of hash bits, where CSDH- denotes CSDH without the Hamming embedding distances. From the experimental results in Fig. 6, we can observe a fact that the Hamming embedding distances can significantly strength retrieval performance from the perspectives of both mAP and precision. This is reasonable and consistent with the intuitive understanding that the Hamming embedding distances can preserve the consistency of semantic similarity of data to some extent. For the multi-label collection, i.e., NUS-WIDE, the performance improvement was particularly pronounced due to the rich semantic information contained in multi-label images, which makes the model more effective.

Figure 7 presents the precision-recall curves of CSDH and its counterpart without the Hamming embedding loss (i.e., CSDH-) on the NUS-WIDE collection when 12 and 48 hash bits were used, respectively. The precision-recall curves further confirmed the effectiveness of the Hamming embedding loss, which could preserve the semantic similarities of data during the process of hash projection. This conclusion holds true for other quantities of hash bits as well; however, due to space constraints, we have not provided these results individually.

Conclusions

In this work, we proposed a novel end-to-end deep hashing model, CSDH, for image retrieval that leverages binary code similarities. The hashing model first employs a pre-trained deep convolutional neural network to extract deep features, capturing the semantic structural information of images. A hidden and fully connected layer is attached to the end of the deep network. This hidden layer referred to as the hash layer, transforms the continuous values outputted by the last layer into binary ones via an activation function. To hold the consistency in similarity preservation, the Hamming embedding distances are also introduced into the loss function. The superiority of CSDH was validated through extensive experiments on two public image collections. The experimental results verify that CSDH exhibits competitive performance compared to popular deep hashing models.

Note that the proposed hashing model, taking AlexNet as its foundational architecture, may inevitably encounter the well-documented issue of gradient vanishing. Besides,

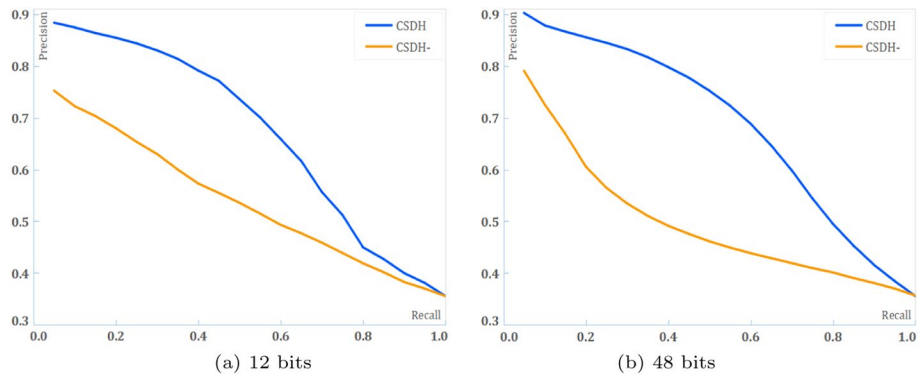


Fig. 7 The precision-recall curves of our model with (or without) the Hamming embedding loss, denoted as CSDH and CSDH-, respectively, on NUS-WIDE

the depth of AlexNet is relatively shallow, potentially limiting the complexity of the deep features it can extract. Thus, our future work will explore the integration of more modern neural network architectures, such as GoogLeNet, ResNet, and DenseNet, into our hashing model. These models offer increased depth and innovative structures, suggesting that they may enhance the feature extraction capabilities of our model, as well as mitigating the gradient vanishing problem. By incorporating these advanced architectures, we aim to improve the robustness and performance of our hashing model.

Acknowledgements
Not applicable.

Author contributions
H.L. and Z.W. designed and implemented the proposed algorithms. D.Y. actively involved in the algorithm designing, debugging and experiment works. M.Y., X.Z. and J.L. provided valuable high-level guidance during algorithm designing, implementation. H.L. wrote the paper. All authors read and approved the final manuscript.

Funding
This work was partially funded by the Natural Science Foundation (NSF) of China (No. 61976195, 61976196) and the Natural Science Foundation of Zhejiang Province (No. LZ23F020003, LR23F020001, LZ22F020010), Outstanding Talents of "Ten Thousand Talents Plan" in Zhejiang Province (No. 2018R51001).

Availability of data and materials
Not applicable.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
The authors have read and approved the final manuscript.

Competing interests
The authors declare that they have no Competing interests.

Received: 1 June 2023 Accepted: 7 April 2024
Published online: 18 April 2024

References

1. Abdalla HB. A brief survey on big data: technologies, terminologies and data-intensive applications. J Big Data. 2022;9(1):107.
2. Tsai C, Lai C, Chao H, Vasilakos AV. Big data analytics: a survey. J Big Data. 2015;2:21.
3. Adadi A. A survey on data-efficient algorithms in big data era. J Big Data. 2021;8(1):1–54.
4. Batko KM, Slezak A. The use of big data analytics in healthcare. J Big Data. 2022;9(1):3.
5. Biswas S, Khare N, Agrawal P, Jain P. Machine learning concepts for correlated big data privacy. J Big Data. 2021;8(1):157.
6. Senoguchi J. Forecast of complex financial big data using model tree optimized by bilevel evolution strategy. J Big Data. 2021;8(1):116.
7. Seliya N, Zadeh AA, Khoshgoftaar TM. A literature review on one-class classification and its potential applications in big data. J Big Data. 2021;8(1):122.
8. Liu H, Li X, Zhang S, Tian Q. Adaptive hashing with sparse matrix factorization. IEEE Trans Neural Net Learn Syst. 2020;31(10):4318–29.
9. Wang J, Zhang T, Song J, Sebe N, Shen HT. A survey on learning to hash. IEEE Trans Pattern Anal Mach Intell. 2018;40(4):769–90.
10. Jafari O, Maurya P, Nagarkar P, Islam K.M, Crushev C. A survey on locality sensitive hashing algorithms and their applications. CoRR abs/2102.08942, 2021;1–23.
11. Gong Y, Lazebnik S, Gordo A, Perronnin F. Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans Pattern Anal Mach Intelli. 2013;35(12):2916–29.
12. Subakti A, Murfi H, Hariadi N. The performance of BERT as data representation of text clustering. J Big Data. 2022;9(1):15.
13. Alzubaidi L, Bai J, Al-Sabaawi A, Santamaria J, Albahri AS, Al-dabbagh BSN, Fadhel MA, Manoufali M, Zhang J, Al-Timemy AH, Duan Y, Abdullah A, Farhan L, Lu Y, Gupta A, Albu F, Abbosh A, Gu Y. A survey on deep learning tools dealing with data scarcity: definitions, challenges, solutions, tips, and applications. J Big Data. 2023;10:46.

14. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili AQ, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J Big Data. 2021;8(1):53.
15. Kaur G, Sharma A. A deep learning-based model using hybrid feature extraction approach for consumer sentiment analysis. J Big Data. 2023;10(1):5.
16. Naik D, Jaidhar CD. Semantic context driven language descriptions of videos using deep neural network. J Big Data. 2022;9(1):17.
17. Singh A, Gupta S. Learning to hash: a comprehensive survey of deep learning-based hashing methods. Knowl Inf Syst. 2022;64(10):2565–97.
18. Xia R, Pan Y, Lai H, Liu C, Yan S. Supervised hashing for image retrieval via image representation learning. In: Proc. the 28th AAAI Conf. Artifi Intell (AAAI'14), pp. 2156–2162, 2014.
19. Li W-J, Wang S, Kang W-C. Feature learning based deep supervised hashing with pairwise labels. In: Proc. the 25th Int. Joint Conf. Artifi Intelli (IJCAI'16), pp. 1711–1717, 2016.
20. Zhu H, Long M, Wang J, Cao Y. Deep hashing network for efficient similarity retrieval. In: Proc. the 30th AAAI Conf. Artifi Intell (AAAI'16), pp. 2415–2421, 2016.
21. Qin Q, Huang L, Wei Z, Nie J, Xie K, Hou J. Unsupervised deep quadruplet hashing with isometric quantization for image retrieval. Inf Sci. 2021;567:116–30.
22. Sun Y, Ye Y, Li X, Feng S, Zhang B, Kang J, Dai K. Unsupervised deep hashing through learning soft pseudo label for remote sensing image retrieval. Knowl Based Syst. 2022;239: 107807.
23. Weiss Y, Torralba A, Fergus R. Spectral hashing. In: Proc. the 21nd Int. Conf. Neural Info. Processing Syst. (NIPS'08), pp. 1753–1760, 2008.
24. Liu W, Wang J, Ji R, Jiang Y-G, Chang S-F. Supervised hashing with kernels. In: Proc. IEEE Conf. Comp. Vision Pattern Recogn. (CVPR'12), pp. 2074–2081, 2012.
25. Lin G, Shen C, Qinfeng S, Hengel A, David S. Fast supervised hashing with decision trees for high-dimensional data. In: Proc. IEEE Conf. Comp. Vision Pattern Recogn. (CVPR'14), pp. 1971–1978, 2014.
26. Xiao Y, Zhang W, Dai X, Dai X, Zhang N. Robust supervised discrete hashing. Neurocomputing. 2022;483:398–410.
27. Cao Y, Long M, Wang J, Zhu H, Wen Q. Deep quantization network for efficient image retrieval. In: Proc. the 30th AAAI Conf. Artifi. Intelli. (AAAI'16), pp. 3457–3463, 2016.
28. Luo X, Ma Z, Cheng W, Deng M. Improve deep unsupervised hashing via structural and intrinsic similarity learning. IEEE Signal Process Lett. 2022;29:602–6.
29. Cui Q, Chen Z-M, Yoshie O. Delving into the representation learning of deep hashing. Neurocomputing. 2022;494:67–78.
30. Wu D, Dai Q, Li B, Wang W. Deep uncoupled discrete hashing via similarity matrix decomposition. ACM Trans Mult Comp Comm Appl. 2023;19(1):22.
31. Wang X, Shi Y, Kitani KM. Deep supervised hashing with triplet labels. In: Proc. the 13th Asian Conf. Computer Vision (ACCV'16), pp. 70–84, 2016.
32. Jiang Q-Y, Cui X, Li W-J. Deep discrete supervised hashing. IEEE Trans Image Process. 2018;27(12):5996–6009.
33. Zhang Z, Zhu L, Li Y, Xu Y. Deep discrete hashing for label distribution learning. IEEE Signal Process Lett. 2022;29:832–6.
34. Li X, Yu J, Wang Y, Chen J-Y, Chang P-X, Li Z. DAHP: deep attention-guided hashing with pairwise labels. IEEE Trans Circ Syst Video Tech. 2022;32(3):933–46.
35. Hu W, Wu L, Jian M, Chen Y, Yu H. Cosine metric supervised deep hashing with balanced similarity. Neurocomputing. 2021;448:94–105.
36. Sun Y, Ye Y, Li X, Feng S, Zhang B, Kang J, Dai K. Unsupervised deep hashing through learning soft pseudo label for remote sensing image retrieval. Knowl Based Syst. 2022;239: 107807.
37. Liu H, Yin M, Wu Z, Zhao L, Li Q, Zhu X, Zheng Z. PLDH: pseudo-labels based deep hashing. Mathematics. 2023;11:2175.
38. Dubey SR, Singh SK, Chaudhuri BB. Activation functions in deep learning: a comprehensive survey and benchmark. Neurocomputing. 2022;503:92–108.

Publisher's Note