Hindawi
Security and Communication Networks
Volume 2018, Article ID 8172725, 13 pages
https://doi.org/10.1155/2018/8172725

*Research Article*

# Deep Learning Hash for Wireless Multimedia Image Content Security

**Yu Zheng**,[1] **Jiezhong Zhu**,[1] **Wei Fang**,[1] **and Lian-Hua Chi**[2]

[1]*School of Computer & Software, Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing 210044, Jiangsu, China*
[2]*Department of Computer Science and Information Technology, La Trobe University, VIC 3086, Australia*

Correspondence should be addressed to Yu Zheng; yzheng@nuist.edu.cn

Received 24 July 2018; Accepted 30 August 2018; Published 25 September 2018

Academic Editor: Weizhi Meng



With the explosive growth of the wireless multimedia data on the wireless Internet, a large number of illegal images have been widely disseminated in wireless networks, which seriously endangers the content security of wireless networks. However, how to identify and classify illegal images quickly, accurately, and in real time is a key challenge for wireless multimedia networks. To avoid illegal images circulating on the Internet, each image needs to be detected, extracted features, and compared with the image in the feature library to verify the legitimacy of the image. An improved image deep learning hash (IDLH) method to learn compact binary codes for image search is proposed in this paper. Specifically, there are three major processes of IDLH: the feature extraction, deep secondary search, and image classification. IDLH performs image retrieval by the deep neural networks (DNN) as well as image classification with the binary hash codes. Different from other deep learning-hash methods that often entail heavy computations by using a conventional classifier, exemplified by $K$ nearest neighbor (K-NN) and support vector machines (SVM), our method learns classifiers using binary hash codes, which can be learned synchronously in training. Finally, comprehensive experiments are conducted to evaluate IDLH method by using CIFAR-10 and Caltech 256 image library datasets, and the results show that the retrieval performance of IDLH method can effectively identify illegal images.

## 1. Introduction

With the rapid development of multimedia technology, the retrieval of wireless multimedia data has become very convenient and easy. The security of multimedia data has attracted more and more attention from researchers. Multimedia data content security belongs to the branch of information security and requires direct understanding and analysis of the information content transmitted in the network. Judging rapidly from the massive information, filtering, and monitoring the abnormal information in the network are the key to ensure the security of the wireless network content. Meanwhile, a large number of illegal images are disseminated in the network, which seriously endangers the security of network content. So, it is very important practical significance to research the content security-oriented image recognition technology and identify and supervise illegal image information in the network. Although there are a lot of image retrieval methods currently, due to the various disadvantages, such as the low expression ability of image feature, high dimension of feature, and low precision of image retrieval, the retrieval results are not always effective.

How to retrieve the large-scale image resources quickly and effectively meet the needs is urgently to solve. Since the hash-based learning algorithm can effectively preserve the similarity between the original feature spaces and the hash code spaces, more and more scholars have drawn attention to it. Particularly, learning deep hash algorithm has greatly improved the retrieval performance.

Besides the widely used text-based search methods, content-based image retrieval (CBIR) has attracted widespread attention of more and more scholars in the past decade [1]. As we all know, the nearest neighbor search algorithm is an effective method for searching for similar data samples [2]. We should not only consider the scalability issue, but also consider most practical large-scale applications which are

affected by dimensional disasters [3]. The high-level encoding complexity prevents widespread adoption in real-time multimedia systems [4]. One of the many practical applications, the approximate nearest neighbors (ANN), is very efficient. However, the method requires huge storage costs and hard to handling high-dimensional data. Hence, quantization techniques have been proposed to encode high-dimensional data vectors recently. Due to the fact that hashing-based ANN search techniques can reduce the storage via storing the compact binary codes, hash technology is widely used in computer vision, machine learning, information retrieval, and other related fields, in view of the retrieval of large-scale data. The goal of hash is to turn high-dimensional data into the low-dimensional compact binary codes [5]. For example, $d$ dimension data is turned into $r$ dimension (d>>r), and generally $r$ dimension data is between dozens of bits and hundreds of bits. After turning high-dimensional data into hash code, we calculate the distance or similarity between the data rapidly. Because of the high efficiency of binary hash code in Hamming distance calculation and the advantages of storage space, hash code is very efficient in large-scale image retrieval.

One of the most famous jobs in hash and the most applied job in the industry was locality sensitive hashing (LSH) [3], which was put forward by Gionis, Indyk, and Motwani in 1999[1]. LSH is one of the most popular data independent methods, which generates hash functions by random projections [6]. In addition to traditional Euclidean distance LSH has been generalized to accommodate other distance and similarity measures such as p-norm distance [3], Mahalanobis metric [7], and kernel similarity [8]. And L. Qi, X. Zhang, W. Dou, and Q. Ni put forward another application direction of LSH in 2017 [9]. They proposed a privacy-preserving and scalable service recommendation approach based on distributed LSH (SerRecdistri-LSH). The purpose of this application is to use SerRecdistri-LSH method to handle service recommendation in distributed cloud environment.

A disadvantage of the LSH family is that LSH usually needs long bit length ($\geq 1000$) to achieve both high precision and recall. This may leads to a huge storage overhead and thus limits the sale at which an LSH algorithm may be applied. In order to achieve the desired search accuracy, LSH often requires the long hash codes, thereby reducing the recall rate. This problem can be alleviated by using multiple hash tables, but it greatly increases storage costs and query time. [10].

So, learning-based data-dependent hashing methods have become increasingly popular because of the benefit that learned compact binary codes can effectively and highly efficiently index and organize massive data [3]. The goal of the data-dependent hashing methods is to generate short hash codes (typically $\leq 200$) using available training data. Various hashing algorithms have been proposed in the literature, of which a large category focuses on linear hashing algorithms which learn a set of hyperplanes as linear hash functions. The representative algorithms in this category include unsupervised PCA Hashing [11], Iterative Quantization (ITQ) [12], and Isotropic Hashing [13] and supervised Minimal Loss Hashing (MLH) [14], Semisupervised Hashing (SSH) [11], Supervised Discrete Hashing (SDH) [6], Ranking-Based Supervised Hashing [15], FastHash [16], etc. A bilinear form of hash functions was introduced by [17, 18].

As an extension of linear hash functions, a variety of algorithms have been proposed to generate nonlinear hash functions in a kernel space, including Binary Reconstructive Embedding (BRE) [10], Random Maximum Margin Hashing (RMMH) [14], Kernel-Based Supervised Hashing (KSH) [12], and the kernel variant of ITQ [13]. In parallel, harnessing nonlinear manifold structures has been shown to be effective in producing compact neighborhood-preserving hash codes. The early algorithm in this fashion is Spectral Hashing (SH) [19], which produces hash codes through solving a continuously relaxed mathematical program similar to Laplacian Eigenmaps. More recently, Anchor Graph Hashing (AGH) [20] leveraged anchor graphs for solving the eigenfunctions of the resulting graph Laplacians, making hash code training and out-of-sample extension to novel data both tractable and efficient for large-scale datasets. Shen et al. [6, 21, 22] proposed a general induction.

The rest of this paper is organized as follows. Section 2 briefly reviews the related works about deep-learning-hashing. And we present framework and steps of the improved deep learning-based hash, IDLH, algorithm in detail in Section 3. Section 4 evaluates the effectiveness of the IDLH through a series of contrastive experiments and carefully analyzes the experimental results, followed by the conclusion in Section 5.

## 2. Related Work

Recently, deep-learning-hashing, as a popular research topic, has drawn increasing attention and research efforts in information retrieval, computer vision, and machine learning. Semantic hashing is recognized as the earliest starting deep learning hash [23]. This method establishes a deep generative model to discover hidden binary features. Such a deep model is made as a stack of restricted Boltzmann machines (RMBs) [24]. After learning a multilayer RBM by pretraining and fine-tuning the document collection, the hash code of any document is obtained by simply thresholding the deepest output. Such a hash code provided by deep RBM is shown to maintain a semantically similar relationship of input documents into code space, where each hash code (or hash key) is used as a memory address to locate the corresponding document. In this way, semantically similar documents are mapped to adjacent memory addresses, enabling efficient searching through hash table lookups. In order to improve the performance of deep RBMS, a supervised version was proposed in [25] and the idea of nonlinear neighborhood component analysis (NCA) embedding in [26] was adopted. The supervised information is derived from training a given neighbor/nonneighbor relationship between samples. Then, based on the depth RBM, the objective function of the NCA is optimized so that the depth RBM yields a hash code. Note that supervised deep RBMS can be applied to wide data fields other than text data. In [25], the depth RBMs are supervised using a Gaussian distribution, and the models of

the visible units in the first layer are successfully applied to the processing of massive image data.

In [27], a deep neural network was developed to learn multilevel nonlinear transformations, mapping the original image to a compact binary hash code to support large-scale image retrieval for learning binary image representations. A deep hashing model is established under the three-layer constraint on the deep network: (1) the reconstruction error between the original real-valued image feature vector and the resulting binary code is minimized; (2) each bit of the binary code has a balance; (3) all bits are independent of each other. Similar constraints were used in previous unsupervised hash or binary coding methods, such as iterative quantization (ITQ) [28]. In [27], a supervised version is called supervised depth hash 3, in which a discriminative item containing two pairs of supervised information is added to the objective function of the deep hash model.

It is worth noting that, in the training of sparse neural networks, in addition to the sparse similarity maintaining hash, depth hash, and supervised depth hash, the pretraining phase is not included. Instead, the hash codes are learned from scratch using a set of training data. However, no pretraining can make the general hash code less efficient. In particular, the sparse similarity keep hash method is found to be inferior to the existing supervised hashing method, i.e., kernel-based supervised hashing (KSH) [29], in terms of search accuracy on some image datasets [30]; the deep hash method and its supervised version are slightly better than ITQ and its supervised version CCA+ ITQ, respectively [31]. Note that KSH, ITQ, and CCA+ITQ develop a shallow learning framework.

One of the main purposes of deep learning is to learn the robust and powerful representation of complex data. It is very natural to use deep learning to explore compact hash codes, which can be thought of as binary representation of the data. The deployed CNN consists of three convolution pools, including rectified linear activation, maximum pool merging, and local contrast normalization, a standard fully connected layer, and an output layer with softmax activation functions. In [32], a new method called deep semantic sorting hashing is proposed to learn hash values, thereby maintaining multilevel semantic similarity between multilabel images. This method is combined with convolutional neural network hashing method, taking image pixels as input, training depth CNN, and jointly learning image feature representation and hash value by this method. The deployed CNN consists of five convolution-pooling layers, two fully connected layers, and a hash layer (i.e., output layer).

Indexing massive amounts of multimedia data, such as images and videos, is a natural application based on learning hashing. In particular, due to the well-known semantic divide, hashing methods have been extensively studied for image search and retrieval, as well as mobile product search [33, 34]. Although hashing techniques have been applied to the active learning framework to cope with big data applications, these hash algorithms are rarely applied to image classification. For the image recognition problem with many categories, the computational and memory overhead primarily stems from the large number of classifiers to be learned. Table 1 show that the complexities can be high at the stages of both training and deploying these classifiers. Inspired by [35, 36], we proposed a combination classifiers with DNN and binarizing classifiers to solve some classic security calculation problems [37–39]. Different from other deep learning-hash methods that often entails heavy computations by using a conventional classifier, exemplified by K-NN and SVM, our method learns classifiers using binary hash codes, which are simultaneously learned from the training data. The combination classifiers can provide both image features and accelerate image classification, and thus it make the large-scale image recognition faster. The advantages of the extending hashing techniques from fast image search to image classification inspire us to apply them to deep learning hash framework.

Table 1: A huge number of parameters need to be learned and stored with different datasets. Considering a classification task with C different categories and D-dimensional feature representation, even the simplest linear models are comprised of D×C parameters.

| Image Dataset | Categories | Dimensions | Parameters |
|---|---|---|---|
| ImageNet | 21,841 | 4,096 | 89,460,736 |
| ILSVRC | 1,000 | 4,096 | 4,096,000 |
| SUN397 | 397 | 12,288 | 4,878,336 |
| CIFAR-10 | 10 | 4,096 | 40,960 |
| Caltech-256 | 256 | 4,096 | 1,048,576 |

With the development of Internet technology, the spreading form of illegal information on the Internet is changing gradually. Traditionally, the form of communication based on word description has been transformed into a diversified form of communication based on video and image. Therefore, the original keyword blocking, web content grading, blacklist restriction, and other filtering methods have not been able to block illegal information dissemination. At present, network illegal image recognition is mainly divided into the following categories.

*(1) Erotic Image Recognition*. The identification of online pornographic images mainly includes methods based on skin color feature extraction detection and judgment and methods based on limb judgment. The limb state is determined by extracting the divided skin color information and the connection feature of the human body posture and further determining whether the image transmitted on the network contains pornographic content.

*(2) Images Involving State Secrets or Military Secrets*. The recognition of secret-related images mainly involves steganography, which hides secret information in image, video, and other carriers for secret transmission. At present, the high-order statistical features based on modeling the complex correlation of image neighborhood become the mainstream features in the field of steganalysis. SRM (Spatial Rich Model), PSRM (Projection Speciation Rich Model), and other models are based on such high-order, high-dimensional features and have achieved good detection results. Steganalysis based on depth learning is a hotspot in

the field of information hiding in order to identify the illegal secret-related images accurately and quickly.

*(3) Images Containing Antihuman Content such as Terrorist Violence*. The identification of such images is mainly based on image contrast techniques. Image comparison includes techniques such as image feature extraction, high-dimensional spatial feature index establishment, and similarity measure. It is a very worthwhile to study how to quickly compare the massive network images to the illegal target images, so that the recall rate and the precision rate can be taken into account.

## 3. The Proposed Method

In this section, we will present the notations, as summarized in Table 2 firstly. The concept of deep learning stems from the field of artificial neural networks. Deep learning is deep neural network learning and is a learning structure with multiple hidden layers. In the process of deep learning, the network is trained layer by layer. Each layer of the learning network extracts certain features and information and takes the training result as a deeper input. Finally, the entire network is fine-tuned with a top-down algorithm.

Through deep learning, complex function expressions can be learned, thereby completing the concept of high-level abstraction from the underlying information. It has been widely used in language understanding, target recognition, and speech perception.

Figure 1 shows that the proposed framework IDLH includes three components. The first component is preprocessing layer on the image dataset. The second component is the training self-coding network with a layer-by-layer greedy learning algorithm to obtain the feature expression function of the image. The third is hash layer which retrieves images similar to the query image with compact binary codes and categorizes the query one by the majority semantic label within the hashing bucket.

*3.1. Preprocessing.* Since the depth learning algorithm used in this paper is an unsupervised learning algorithm, it can automatically learn the deep features of the image from the original pixel information of the image. Therefore, the original pixel value of the whole image can be directly used as input data for the deep learning model. In order to facilitate the training of the network, it is necessary to preprocess the image. Through preprocessing, the image is simply scaled, sample-by-sample mean-value reduction, and whitening is processed to reduce the redundant information in the image and facilitate the deep learning network for training and calculation. Preprocessing can be further grouped into three suboperations.

*(1) Normalization*. Normalization can prevent neuron output saturation caused by excessive net input absolute value. We use the sigmoid function to do normalization, as shown as follows:

Table 2: Summary of notations.

| Symbol | Definition |
|---|---|
| $x_i$ | the gray value of the image pixels |
| $\mu^{(i)}$ | mean pixels |
| U | an arbitrary orthogonal matrix and defines in the ZCA whitening |
| J(W,b) | the quantization loss between the learned binary values and the real values |
| $\lambda$ | a weight attenuation parameter |
| $S_i$ | $ith$ set |
| a, b | two thresholds |
| D | dimensionality of data points |
| M, K | number of candidate images |
| E | the objective function |

$$x_i = \frac{1}{(1 + e^{-x_i})} \tag{1}$$

$x_i$ is the gray value of the image pixels.

*(2) Pointwise Mean Reduction*. This process mainly is to get rid of the redundant information of the image, the mean value is eliminated for each point of the image, the average brightness of the image is removed, and the DC component of the data is eliminated. Assuming that $x^{(i)} \in R^n$ is the gray value of each pixel of image I, we use formulae (2) and (3) to zero-mean image.

$$\mu^{(i)} = \frac{1}{n}\sum_{j=1}^{n} x_j^{(i)} \tag{2}$$

$$x_j^{(i)} = x_j^{(i)} - \mu^{(i)} \tag{3}$$

*(3) Whitening*. Whitening is an important pretreatment process; its purpose is to reduce the redundancy of input data, so that the whitened input data has the following properties: (i) low correlation between features; (ii) all features having the same variance, then the formula is as formula (4) .

In formula (4) the rotation matrix of $x_{rot,i}$ is $U^T x_i$. Generally, when $x$ is in interval [-1,1], $\varepsilon \approx 10^{-5}$.

$$x_{ZCAwhite} = U\frac{U^T x_i}{\sqrt{\lambda_i + \varepsilon}} = U\frac{x_{rot,i}}{\sqrt{\lambda_i + \varepsilon}} \tag{4}$$

*3.2. Training Stack Sparse Self-Encoding Network.* Stack self-coding neural network has strong expressive ability, which mainly benefits from its hierarchical feature representation. Through one level of feature learning, we can learn the hierarchical structure between features. Stack self-encoding neural network is a neural network model composed of multilayer sparse self-encoder, that is, the output of the former self-encoder as the input of the latter self-encoder.

In the training the original input $x^{(k)}$ is used as input to train the first self-encoded neural network. At this point, for each training sample $x^{(k)}$, the output $h_1^{(k)}$ of the hidden

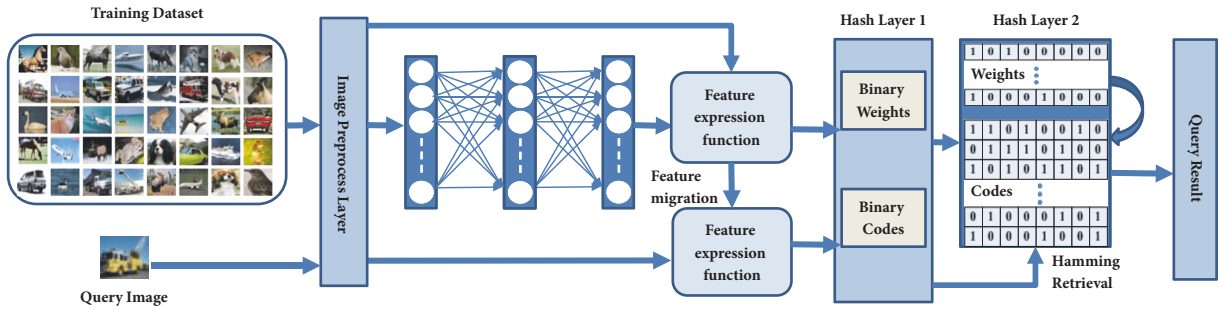



Figure 1: Deep learning-hash retrieval framework. IDLH consists of three main components (preprocessing layer, deep neural network layer, and hash layer). The object of the first layer is simply scaled, sample-by-sample mean-value reduction, and whitening. In the second component, we develop a deep neural network to obtain the feature expression function of the image. And the classifier weights and feature binary codes are simultaneously learned in the last component-hash layer.

layer can be obtained, and the output of the hidden layer can be used as the input of the second self-encoder to continue training the second self-encoder. Then, the output $h_2^{(k)}$ of the second hidden layer of the self-encoder can be obtained. The output $h_1^{(k)}$ of the first hidden layer of the self-encoder is called a first-order feature, and the output $h_2^{(k)}$ of the second hidden layer of the self-encoder is called a second-order feature. In order to classify, the two-order feature $h_2^{(k)}$ can be used as the input of Softmax regression.

Figure 2 shows the flowchart of the proposed method. And there are mainly three processes (supervised pretraining, rough image retrieval, and accurate image retrieval). The object of the first process is to transform the high-dimensional feature vector into a low-dimensional compact two value codes through hash function. In the second procedure, we pick out M candidate images by calculating Hamming distance. In the third process, we calculate the Euclidean distance between the candidate image and the image to be retrieved and accurately extract K images from the M candidate images.

Figure 3 shows a block diagram of a self-encoding neural network. The stacking self-encoding network contains 3 hidden layers (feature layers). The input layer inputs the original data i into the first layer of the feature layer, the output result of the former layer serves as the input of the next layer, and the output of the third layer serves as the feature expression of the image. In our method, it is also used as the input of the hash classifier, and it is possible to use the characteristics of the STD neural network to classify the features.

By using the matrix representation of the binary codes vectors and the output of the 3th layer of the network, we use the gradient descent method to solve the neural network.

For a single sample (x, y), the cost function is as shown in

$$J(W,b;x,y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2 \tag{5}$$

For datasets containing m samples, the optimization cost function is formulated by the following formula:

$$\begin{aligned}\min_{W,b} J &= \left[\frac{1}{m}\sum_{i=1}^{m} J\left(W,b;x^{(i)},y^{(i)}\right)\right] \\ &\quad + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2 \\ &= \left[\frac{1}{m}\sum_{i=1}^{m}\left(\frac{1}{2}\left\|h_{W,b}\left(x^{(i)}\right) - y^{(i)}\right\|^2\right)\right] \\ &\quad + \frac{\lambda}{2}\sum_{l=1}^{n_l-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(W_{ji}^{(l)}\right)^2\end{aligned} \tag{6}$$

The first term $J(W,b)$ represents the mean variance term. The second term aims to prevent the data from overfitting by reducing the magnitude of the weight. $\lambda$ is a weight attenuation parameter. It is used to balance the relative importance of mean square deviation terms and weight attenuation terms. Our purpose is to minimize the quantization loss $J(W,b)$ between the learned binary values and the real values of an input image according to parameters $W$ and $b$.

*3.3. Hash Algorithm Retrieval.* The image retrieval method based on hash algorithm maps the high-dimensional content features of images into Hamming space (binary space) and generates a low-dimensional hash sequence to represent a picture. This method reduces the requirement of computer memory space for image retrieval system, improves the retrieval speed, and better adapts to the requirements of mass image retrieval.

Inspired by [6, 8], we use a set of hash functions to hash data into different buckets. After we do some hash mapping on the original image feature data, we hope that the original two adjacent feature data can be hash into the same bucket with the same bucket number. And then, after hash mapping of all the data in the original feature set, we can get a hash table. These original feature data sets are scattered into hash table buckets, and the data belonging to the same bucket is probably adjacent to the original data. However, there is also

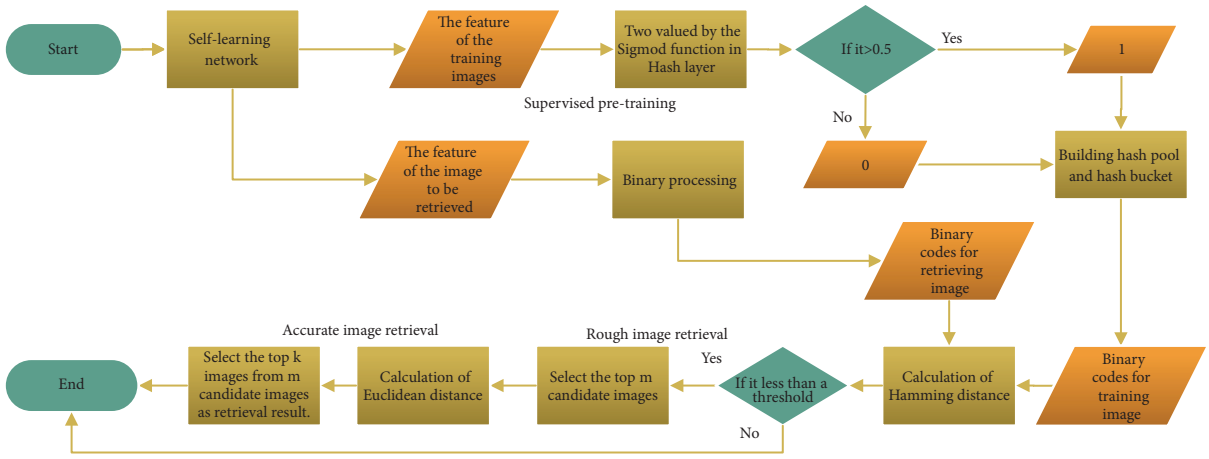



Figure 2: Deep learning-hash retrieval flowchart. IDLH mainly includes three processes (supervised pretraining, rough image retrieval, and accurate image retrieval).

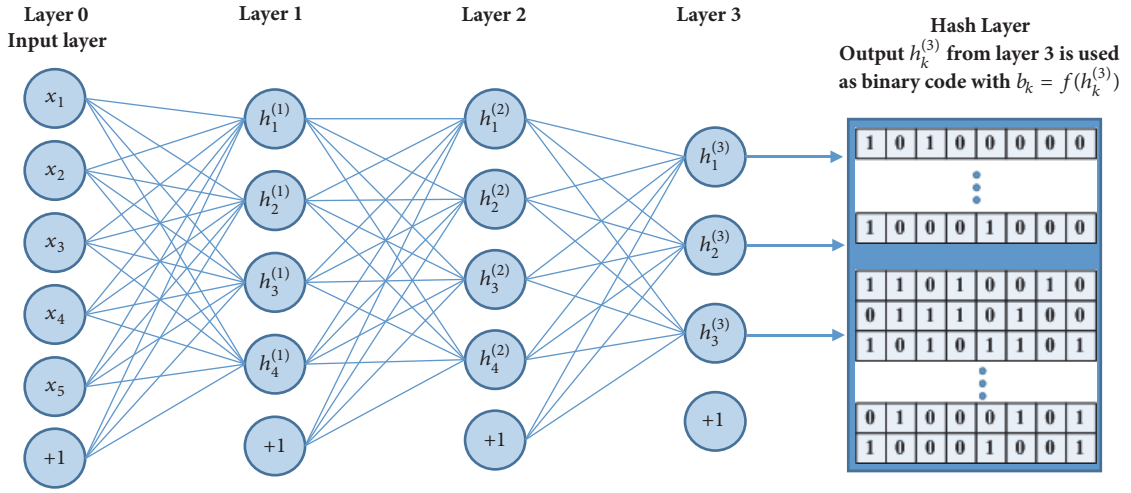



Figure 3: Self-learning network based on stack self-encoding network. The neurons labeled $x_i$ is the input of the neural network and "+ 1" are the offset nodes (intercept entries) of the neural network. The layer 0 is the input layer of neural network, and layer 3 is the output layer of neural network. The middle layers of layer 0 layer to layer 3 are the hidden layers of neural network.

a small probability in events; that is, the nonadjacent data is hash to the same barrel. Set the hash function as the follows:

$$h_k(x) = \operatorname{sgn}\left(w_k^t x + b_k\right) \tag{7}$$

Here $w_k$ is the projection vector and $b_k$ is the corresponding intercept. The code value generated by formula (7) is $\{-1, 1\}$, and we use the following formula to convert it into two value codes:

$$y_k = \frac{1}{2}\left(1 + h_k(x)\right) \tag{8}$$

Given a sample point $\in R^D$, we can compute a K-bit binary code $y$ for $x$ with formula (9). The hash function performs the mapping as $h_k : R^D \longrightarrow B$

$$y = \{h_1(k), h_2(k), \ldots, h_k(k)\} \tag{9}$$

Then, for a given set of hash functions, we can map them to a set of corresponding binary codes by formula (10).

$$Y = H(X) = \{h_1(X), h_2(X), \ldots, h_k(X)\} \tag{10}$$

Here $X = \{x_n\}_{n=1}^N \in R^{D\times N}$ is the feature data matrix with points as columns. Such a binary encoding process can also be viewed as mapping the original data point to a binary valued space, namely, Hamming space.

*3.4. Similarity Measure.* After obtaining the binary hash code of the image, it is necessary to measure similarity between the retrieved image and the library image in the Hamming space. The smaller the Hamming distance is, the closer distance between the two data is, and the degree of similarity is higher; otherwise, the two data similarity is lower.

$$d_H\left(y_i, y_j\right) = y_i \oplus y_j \tag{11}$$

Table 3: Image library image storage structure.

| Hash Sequence ID | Hash Code | Image ID |
|---|---|---|
| 0 | 010011101011 | Cat1.jpg |
| 1 | 001110101010 | Cat2.jpg |
| ..... | ..... | ..... |
| 200 | 101010101001 | Cat200.jpg |
| ..... | ..... | ..... |

Here $\oplus$ is an XOR operation. The two sets of $y_i$ and $y_j$ represent the hash code of the search image feature and the image library is mapped through the hash function. The new image features learned by the stack self-encoding network are generated by the hash function. The storage structure of the image feature vectors is shown in Table 3.

As can be seen from Table 3, the hash code of the image is related to the image *ID* and the image name one by one. In the process of searching, the image feature vector is obtained through deep learning by a hash function, the original data is mapped into a new data space, and a corresponding hash code is obtained. The hash code is used to calculate the Hamming distance in the Hamming space as a measure of similarity between images. Finally, the storage structure of the image feature vector is used to find the corresponding image *ID* of the hash code, and the output retrieval result is output to the user.

*3.5. Image Secondary Search.* In the first-level search phase, the features learned from the deep learning network are mapped into the Hamming space using the hash function. In the similarity measurement phase, the traditional Euclidean distance is abandoned. Measure the similarity between images by comparing the Hamming distance between the image features of the query image and the image of the library image. In order to further improve the accuracy of retrieval without affecting the real-time performance, we can retrieve the image by the second level retrieval. These steps are described in detail as follows: After one level retrieval, we choose the $K$ images with the most similarity in the first-level retrieval result and then calculate the Euclidean distance between the original feature vector of the $K$ images and the original feature vector of the query image. The results obtained as the similarity measure of the images and output the retrieval result that has been ranked from the high and low with the similarity distance.

Although the Hash algorithm maps the high-dimensional feature vectors of the image into a hash-coded form, the problem of "dimensional disasters" is solved, and the retrieval efficiency is greatly accelerated. However, when the similarity comparison is performed, the Hamming distances of the image features are simply compared using the results of the primary search, and occasionally undesirable results may still appear on the search results. If we want to increase the accuracy of the search, we must increase the hash code length. However, excessively long codes will increase the amount of calculations, increase the memory burden, and reduce the real-time nature of retrieval, failing to achieve the goal of reducing the size of data. In order to solve this problem, keep the retrieval efficiency, and further improve the retrieval accuracy, we propose a search strategy for secondary retrieval, the specific steps of which are as follows.

*Step 1.* Through the first-level search in the Hamming space, the similarity degree of the images is sorted, and the top $K$ sorting images are selected.

*Step 2.* For the $k$ images in Step 1, calculate the Euclidean distance one by one from its original image feature vector to the image feature vector of the query image.

*Step 3.* The Euclidean distance calculated in Step 2 is sorted. The smaller the calculated value is, the higher similarity between images is, and the similarity is sorted from high to low and output as the final search result.

In the second search, it is necessary to pay attention to the selection of the $K$ value, although the larger the $K$ value is, the better the search effect is, but accordingly, the longer the time is consumed. Therefore, it is necessary to combine various factors to select the appropriate $K$ value.

## 4. Experimental and Performance Analysis

In this section, we thoroughly compare the proposed approach with the improved deep learning hash retrieval methods on several benchmark datasets. Through a series of experiments, the effectiveness and feasibility of the proposed algorithm are verified.

*4.1. Database.* Two mostly used databases in the recent deep learning hash works are taken into our evaluation. The two image libraries are derived from the CIFAR-10[11] core experimental image library dataset and the Caltech 256 image library dataset.

CIFAR-10 dataset contains 10 object categories and each class consists of 6,000 images, resulting in a total of 60,000 images. The dataset is split into training and test sets which are averagely divided into 10 object classes. The Caltech 256 image library dataset contains 29,780 color images which are grouped intro 256 classes.

First, test the CIFAR-10 image dataset. There are a total of 50,000 training samples which are used for training on the deep learning network. The remaining 10,000 images are used as test samples. And then we randomly select 50 images from database as the query images. For the Hidden Image Retrieval algorithm based on deep learning mentioned in this paper, the image pixel data is directly used as input, while for other algorithms, the 512-dimensional GIST feature is used as the feature expression of the image. Note quantization all images into $32*32$ sizes before experiment.

For the Caltech 256 image, a total of 256 classes are included, and each class contains at least 70 images. Therefore, 70 images of each class, a total of 17,920 images, are randomly selected and are used as training images. The remaining images are used as test samples. In addition, all of the images' size is set to $64*64$ again when training.

*4.2. Evaluation Metrics.* We measure the performance of compared methods using Precision-recall and Average-Retrieval Precision (ARP) curves. Precision is the ratio of the correct number of images m in the search result to the number k of all returned images. The formula is as follows:

$$precision = \frac{m}{k} \times 100\% \tag{12}$$

Recall is the ratio of the correct number of images m in the search results to the number g of images in the image library. The formula is as follows:

$$recall = \frac{m}{q} \times 100\% \tag{13}$$

Assume that the search result of the query image $i$ is $B_i$ and $A_i$ means that the category is the same between the query image and the return image, then the accuracy rate for the image query result $P(i)$ can be defined by the following formula:

$$P(i) = \frac{|A(i) \cap B(i)|}{|B(i)|} \tag{14}$$

Average-Retrieval Precision (ARP): the average value of all the images in the same class as the retrieval rate obtained from the retrieval image is defined as follows:

$$ARP(ID_m) = \frac{1}{N} \sum_{id(i)=ID_m} P(i) \tag{15}$$

Here $ID_m$ is the category index number of the image, $m$ is the category index, $N$ is the number of images whose category is $ID_m$, and $id(i)$ is the category index number of the query image.

*4.3. Performance Analysis.* In the proposed algorithm, IDLH, the length of the hash sequence and the depth of the hidden layer in the deep learning network are two key parameters. When the hash sequence length is small, different feature vectors can easily be mapped into the same hash sequence, so the retrieval accuracy is low. However, if the hash sequence is too long, a large storage space is required and a long time is consumed, which reduces the real-time performance. For the number of hidden layers, the number of layers in the hidden layer is too small, which is not conducive to learning strong image features. However, if the depth of the hidden layer is too large, the difficulty of training is increased. In order to verify the effectiveness and feasibility of our algorithm, we conducted the following experiments.

*(1) Results on CIFAR-10 Dataset*. Figure 4(a) shows the search results of the Average-Retrieval Precision using our proposed algorithm, IDLH, compared with the LSH algorithm [3] and other three deep learning algorithms, the DH algorithm [21], the DeepBit algorithm [40], and the UH-BDNN algorithm [41] on CIFAR-10 dataset with 8, 16, 32, 48, 64, 96, 128, and 256 bits. Figure 4(b) shows the Precision-recall curve under 48-bit encoding. It can be seen that the algorithm has a higher precision than the other hashing algorithms with the same recall rate. However, the advantage is not obvious, and the average accuracy is slightly higher than other hash algorithms.

In order to overcome the above defects, we use deep learning to perform hash mapping on image features, perform hash encoding of different bits on the same feature, calculate the Precision-recall of the search results under the condition of different coded bits, and determine the impact of the encoding length on the retrieval results.

As Figure 5 shows, with the increase in the number of coded bits, the Precision-recall is continuously increasing. With the increase in the number of coded bits, the image is better expressed. However, after the number of coded bits reaches 64, even if the number of coded bits increases, the average accuracy rate increases relatively slowly. Because the information of the tiny image is relatively simple, when the number of encoded bits reaches 64 bits, a relatively good image expression has been obtained, and the performance of the algorithm has basically stabilized. At this time, despite increase in the number of encoding bits, it is not very helpful to improve the accuracy rate.

In addition, we want to test the influence of the number of hidden layers in the deep learning network on the retrieval result by changing the number of hidden layers.

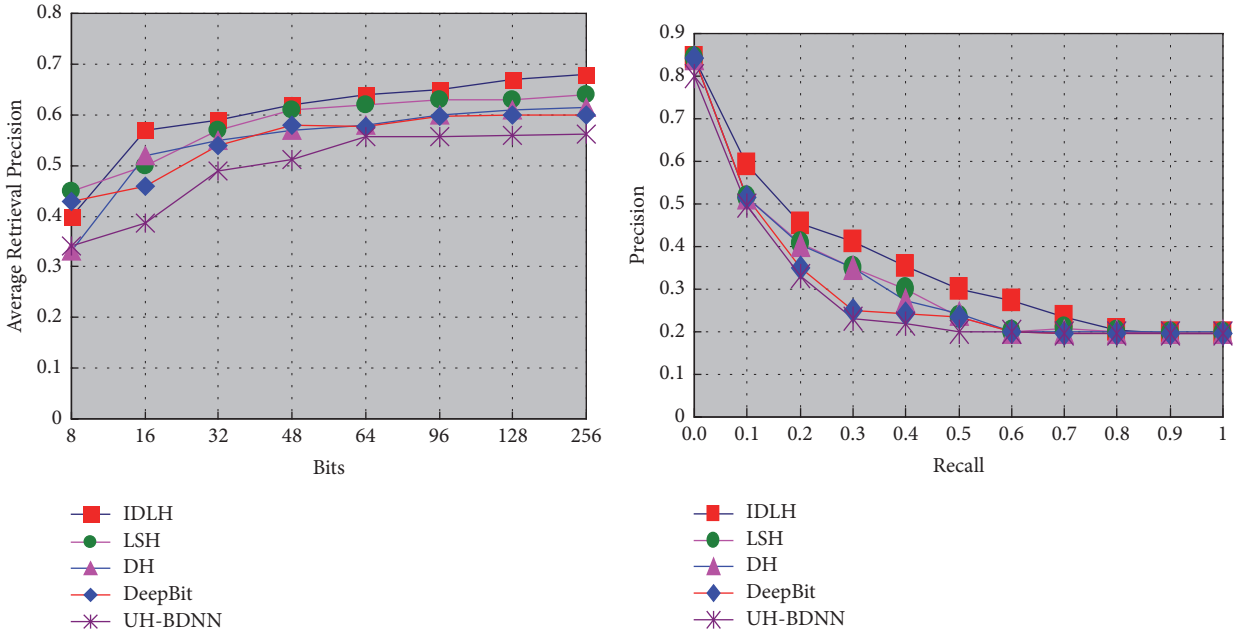
Figure 6 shows the effect of deep learning networks on experimental results in the case of different hidden layer numbers.

It can be seen that deeper networks do not have much improvement in performance, which is different from the expectation that more hidden layers will help learn stronger image features. Since the image library data used in the experiment is a tiny image library, relatively good image characteristics can be learned using a deep learning network with fewer layers. However, if the image library is replaced with a more colorful image, the deep neural network can acquire more detailed image features, and the deepening of the learning network will greatly help the study of image features.

*(2) Results on Caltech 256 Image Data Set*. Figure 7(a) shows the results of the Average-Retrieval Precision results when the number of coded bits is different. Compared with the black and white image library, the proposed algorithm embodies the advantage of image feature learning and leads the Average-Retrieval Precision to other hash retrieval algorithms. In Figure 7(b), we can also see that the algorithm proposed in this paper has a higher Precision-recall than other algorithms under the same recall rate, and it has better search performance.

As shown in Figure 8, as the number of coded bits increases, the precision rate increases with the same recall rate. This feature of the color image library is more pronounced than the black and white image library. Because the color image contains more information, more coding is needed to express it, and the increase of encoding helps to learn the features of the image. The experimental results also show that the deep learning network has learned more excellent image features.

In Figure 9, the precision rate is significantly improved by the increase in the number of hidden layers in the




(a) Average-Retrieval Precision

(b) Precision-recall at 48 bits

Figure 4: Five kinds of algorithm retrieval performance comparison on CIFAR-10 dataset.

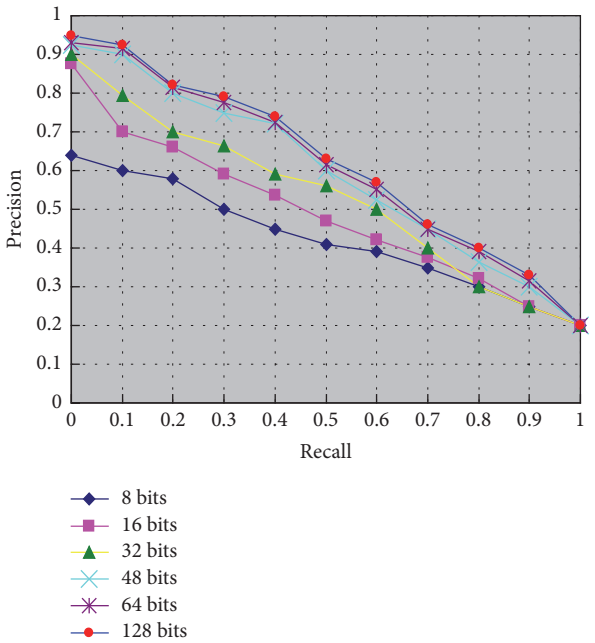



Figure 5: Precision-recall curves with lengths.

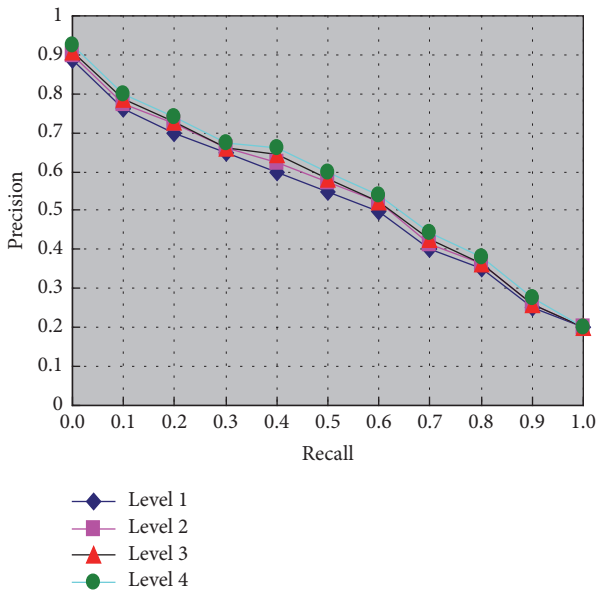
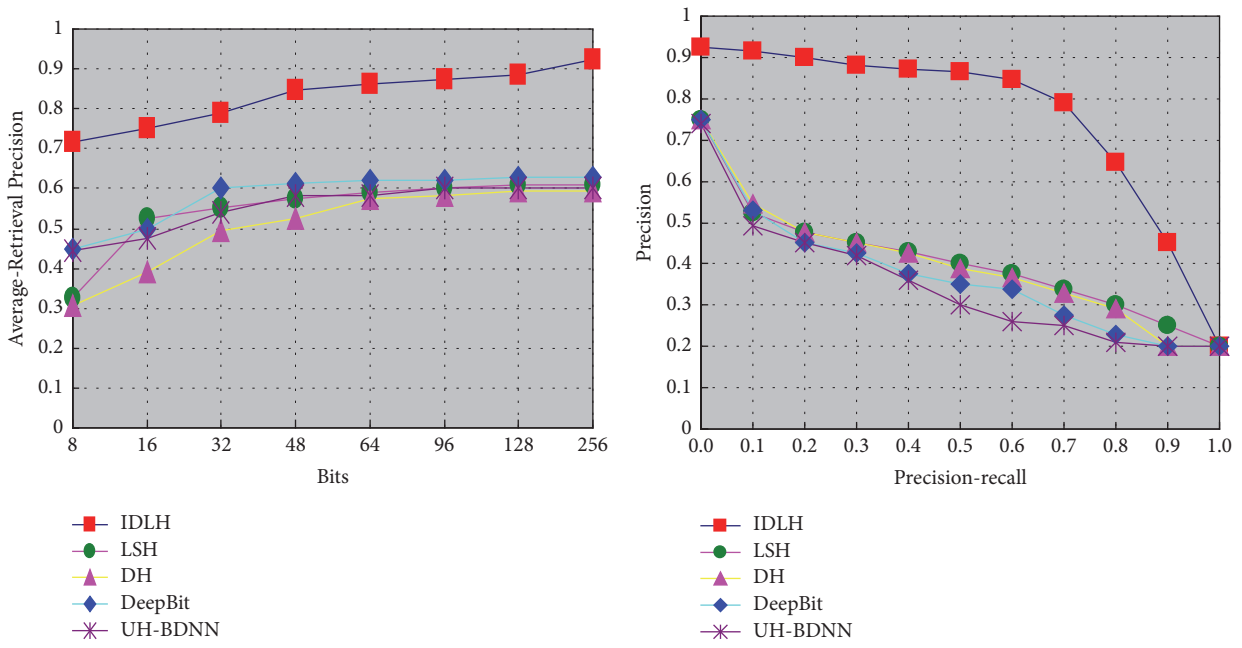



Figure 6: Precision-recall curves with different code different hidden layers.

color image library Caltech 256. This is because the information contained in a more colorful image is more complex. Adding a hidden layer can learn more details of the image and help improve the accuracy of the search.

Next, we tested the performance of secondary image retrieval. The value of k in the secondary search is 20, and the experimental results are shown in Figure 10. As can be seen from the results, secondary retrieval can effectively improve the retrieval accuracy when the number of coded bits is small. However, with the increase in the number of encoding bits, the results of the secondary search and the accuracy of the primary search are not much different. This is because the shorter the hash sequence is, the easier the feature vectors with different original features are mapped to the same hash code. In order to make up for the errors




(a) Average-Retrieval Precision

(b) Precision-recall at 48 bits

Figure 7: Five kinds of algorithm retrieval performance comparison on Caltech 256 set.

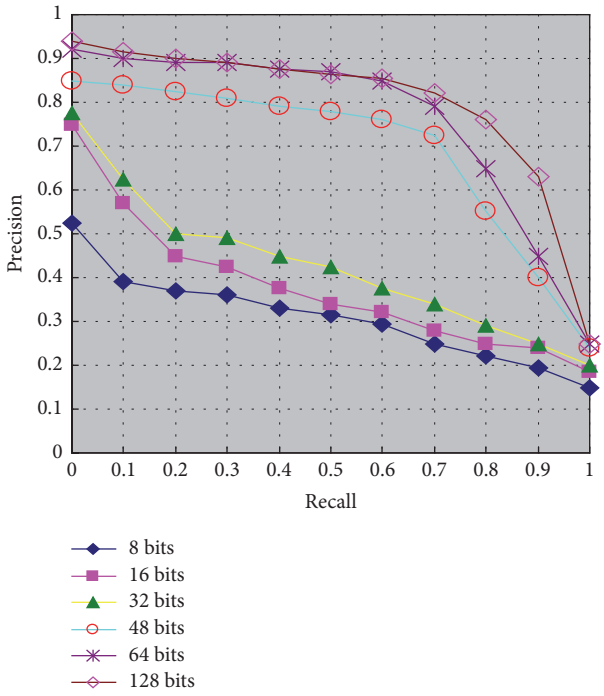



Figure 8: Precision-recall curves with different code lengths.

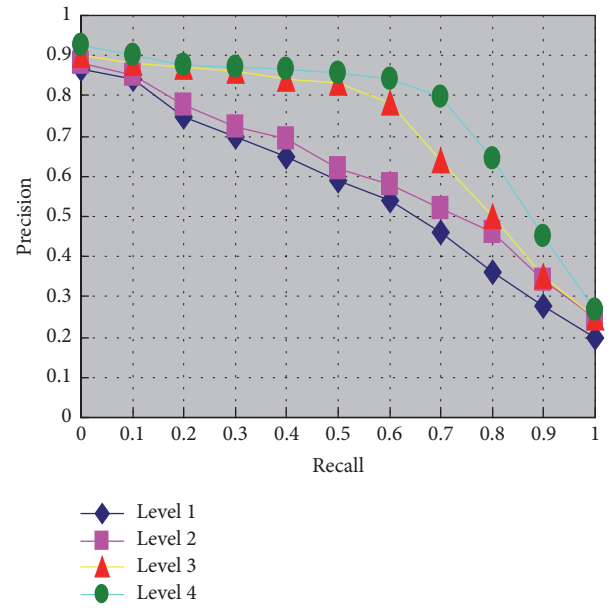



Figure 9: Precision-recall curves with different hidden layers.

caused by the short hash code, it is necessary to perform secondary search. When the number of encoding bits is small, a secondary retrieval method is used in IDLH, and the search accuracy rate can be improved at the expense of a small search speed.

## 5. Conclusion

With the rapid development of data storage and digital process, more and more digital information is transformed and transmitted over the Internet day by day, which brings people a series of security problems as well as convenience [42]. The researches on digital image security, that is, image encryption, image data hiding, and image authentication, become more important than ever [38, 39]. The most essential problem of

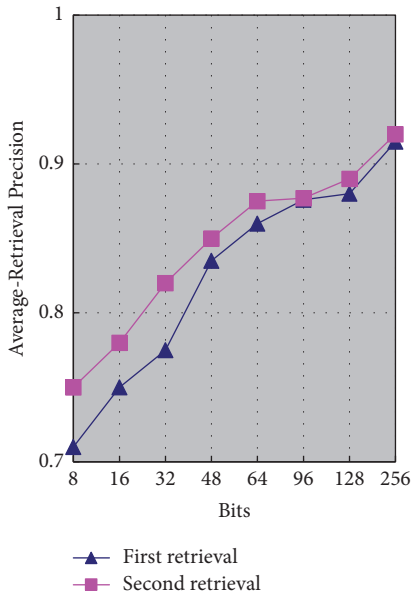



Figure 10: Average-Retrieval Precision with first and second retrieval.

image recognition is to extract robust features. The quality of feature extraction is directly related to the effect of recognition, so most of the previous work on image recognition is spent on artificial design features [43]. In recent years, the emergence of deep learning technology has changed the status of artificial design classification characteristics. Deep learning technology simulates the mechanism of human visual system information classification processing, from the most primitive image pixels to lower edge features, then to the target components that are combined on the edge, and finally to the whole target, depth learning can be combined by layer by layer. The high-level feature is the combination of low level features. From low level to high level, features are more and more abstract and show semantics more and more. From the underlying features to the combination of high-level features, it is the depth of learning that is done by itself. It does not require manual intervention. Compared with the characteristics of the artificial design, this combination of features can be closer to the semantic expression.

In terms of illegal image retrieval, the traditional recognition method should establish a recognition model for each type of recognition task. In the actual application, a recognition model needs a recognition server. If there are many identification tasks, the cost is too high. We used the deep neural network to recognize the illegal image, it only needs to collect the samples of every kind of illegal image and participate in the training of the deep neural network. Finally, a multiclassification recognition model is trained. When classifying unknown samples, deep neural network accounting calculates the probability that the image belongs to each class.

We all know that, in the image detection process, the accuracy and recall rate are mutually influential. Ideally, both must be high, but in general, the accuracy is high and the recall rate is low; the recall rate is high and the accuracy is low. For image retrieval, we need to improve the accuracy under the condition of guaranteeing the recall rate. For image disease surveillance and anti-illegal images, we need to enhance the recall under the condition of ensuring accuracy. Therefore, in different application scenarios, in order to achieve a balance between accuracy and recall, perhaps some game theory (such as Nash Equilibrium [44, 45]) and penalty function [46–48] can provide related optimization solutions.

In this paper, we proposed an improved deep-learning-hashing approach, IDLH, which optimized over two major image retrieval process.

(a) In the feature extraction process, the self-encoded network of the look-ahead type is trained by using unlabeled image data, and the expression of robust image features is learned. This unlabeled learning method does not require image library labeling and reduces the requirements for the image library. At the same time, it also takes advantage of the deep learning network's strong learning ability and obtains better image feature expression than ordinary algorithms.

(b) On the index structure, a secondary search is proposed, which further increases the accuracy of the search, at the expense of very little retrieval time.

Through experiments, the algorithm proposed in this paper is compared with other classic hashing algorithms on multiple evaluation indicators. Firstly, we tested the learning networks of different code lengths and depths in order to test their effect on the retrieval system and then tested the performance of the secondary search. Through the above-mentioned series of experiments for different parameters, the effectiveness of the improved deep learning hash retrieval algorithm proposed in this paper is verified, and through the experimental data, the good retrieval results are proved. In addition, the proposed deep hashing training strategy can also be potentially applied to other hashing problems involving data similarity computation.

## Data Availability

The data used to support the findings of this study are included within the article.

## Conflicts of Interest

The authors declare that they have no conflicts of interest.

## Acknowledgments

The work was funded by the National Natural Science Foundation of China (Grants nos. 61206138 and 61373016).

## References

[1] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Image retrieval: ideas, influences, and trends of the new age," *ACM Computing Surveys*, vol. 40, no. 2, article 5, 2008.

[2] G. Shakhnarovich, T. Darrell, and P. Indyk, *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*, MIT Press, Cambridge, MA, USA, 2006.

[3] A. Gionis, P. Indyk, and R. Motwani, "Similarity search in high dimensions via hashing," in *25th Int. Conf*, pp. 518–529, 1999.

[4] Z. Pan, J. Lei, Y. Zhang, and F. L. Wang, "Adaptive fractional-Pixel motion estimation skipped algorithm for efficient HEVC motion estimation," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 14, no. 1, pp. 1–19, 2018.

[5] G.-L. Tian, M. Wang, and L. Song, "Variable selection in the high-dimensional continuous generalized linear model with current status data," *Journal of Applied Statistics*, vol. 41, no. 3, pp. 467–483, 2014.

[6] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni, "Locality-sensitive hashing scheme based on p-stable distributions," in *Proceedings of the 20th Annual Symposium on Computational Geometry (SCG '04)*, pp. 253–262, ACM, June 2004.

[7] B. Kulis, P. Jain, and K. Grauman, "Fast similarity search for learned metrics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2143–2157, 2009.

[8] M. Raginsky and S. Lazebnik, "Locality-sensitive binary codes from shift-invariant kernels," in *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems, NIPS 2009*, pp. 1509–1517, Canada, December 2009.

[9] L. Qi, X. Zhang, W. Dou, and Q. Ni, "A distributed locality-sensitive hashing-based approach for cloud service recommendation from multi-source data," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2616–2624, 2017.

[10] M. A. Carreira-Perpiñán and R. Raziperchikolaei, "Hashing with binary autoencoders," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 557–566, USA, June 2015.

[11] J. Wang, S. Kumar, and S.-F. Chang, "Semi-supervised hashing for large-scale search," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 12, pp. 2393–2406, 2012.

[12] Y. Gong, S. Lazebnik, and A. Gordo, "Iterative quantization: a Procrustean approach to learning binary codes for large-scale image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '11)*, pp. 2916–2929, June 2011.

[13] W. Kong and W. J. Li, "Isotropic hashing," *NIPS*, vol. 25, 2012.

[14] M. Norouzi and D. J. Fleet, "Minimal loss hashing for compact binary codes," in *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 353–360, USA, July 2011.

[15] J. Wang, W. Liu, A. X. Sun, and Y.-G. Jiang, "Learning hash codes with listwise supervision," in *Proceedings of the 2013 14th IEEE International Conference on Computer Vision, ICCV 2013*, pp. 3032–3039, Australia, December 2013.

[16] G. Lin, C. Shen, Q. Shi, A. Van Den Hengel, and D. Suter, "Fast supervised hashing with decision trees for high-dimensional data," in *Proceedings of 27th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*', pp. 1971–1978, USA, 2014.

[17] Y. Gong, S. Kumar, H. A. Rowley, and S. Lazebnik, "Learning binary codes for high-dimensional data using bilinear projections," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2013*, pp. 484–491, USA, June 2013.

[18] W. Liu, J. Wang, Y. Mu, and S. Kumar, "Compact hyperplane hashing with bilinear functions," in *The 29th International Conference on Machine Learning (ICML12)*, pp. 467–474, 2012.

[19] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Proceedings of the 22nd Annual Conference on Neural Information Processing Systems (NIPS '08)*, pp. 1753–1760, Vancouver, Canada, December 2008.

[20] W. Liu, J. Wang, S. Kumar, and S. F. Chang, "Hashing with graphs," in *The 28th international conference on machine learning (ICML11)*, 2011.

[21] F. Shen, X. Zhou, Y. Yang, J. Song, H. T. Shen, and D. Tao, "A fast optimization method for general binary code learning," *IEEE Transactions on Image Processing*, vol. 25, no. 12, pp. 5610–5621, 2016.

[22] F. Shen, W. Liu, S. Zhang, Y. Yang, and H. T. Shen, "Learning binary codes for maximum inner product search," in *Proceedings of the 15th IEEE International Conference on Computer Vision, ICCV 2015*, pp. 4148–4156, Chile, December 2015.

[23] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS '12)*, pp. 1097–1105, Lake Tahoe, Nev, USA, December 2012.

[24] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *The American Association for the Advancement of Science: Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[25] A. Torralba, R. Fergus, and Y. Weiss, "Small codes and large image databases for recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, 2008.

[26] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," *Journal of Machine Learning Research*, vol. 2, pp. 412–419, 2007.

[27] V. E. Liong, J. Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 2475–2483, USA, June 2015.

[28] Y. Gong, S. Lazebnik, A. Gordo, and F. Perronnin, "Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 12, pp. 2916–2929, 2013.

[29] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR '12)*, pp. 2074–2081, Providence, RI, USA, June 2012.

[30] J. Masci, A. Bronstein, M. Bronstein, and P. Sprechmann, "Sparse similarity-preserving hashing," in *Int. Conf. Learn. Represent*, pp. 1–13, 2014.

[31] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 6, pp. 1092–1104, 2012.

[32] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015*, pp. 1556–1564, June 2015.

[33] G. Cheng, C. Yang, X. Yao, L. Guo, and J. Han, "When Deep Learning Meets Metric Learning: Remote Sensing Image Scene Classification via Learning Discriminative CNNs," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–11.

[34] J. He, J. Feng, X. Liu et al., "Mobile product search with Bag of Hash Bits and boundary reranking," in *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012*, pp. 3005–3012, USA, June 2012.

[35] F. Shen, Y. Mu, Y. Yang et al., "Classification by retrieval: Binarizing data and classifiers," in *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2017*, pp. 595–604, Japan, August 2017.

[36] P. Li, S. Zhao, and R. Zhang, "A cluster analysis selection strategy for supersaturated designs," *Computational Statistics & Data Analysis*, vol. 54, no. 6, pp. 1605–1612, 2010.

[37] A. Pradeep, S. Mridula, and P. Mohanan, "High security identity tags using spiral resonators," *Cmc-Computers Materials & Continua*, vol. 52, no. 3, pp. 187–196, 2016.

[38] Y. Cao, Z. Zhou, X. Sun, and C. Gao, "Coverless information hiding based on the molecular structure images of material," *Computers, Materials and Continua*, vol. 54, no. 2, pp. 197–207, 2018.

[39] Y. Liu, H. Peng, and J. Wang, "Verifiable diversity ranking search over encrypted outsourced data," *Cmc-Computers Materials & Continua*, vol. 55, no. 1, pp. 037–057, 2018.

[40] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016*, pp. 1183–1192, USA, July 2016.

[41] T. Do, A. Doan, and N. Cheung, "Learning to Hash with Binary Deep Neural Network," in *Computer Vision – ECCV 2016*, vol. 9909 of *Lecture Notes in Computer Science*, pp. 219–234, Springer International Publishing, Cham, 2016.

[42] Rui Zhang, Di Xiao, and Yanting Chang, "A Novel Image Authentication with Tamper Localization and Self-Recovery in Encrypted Domain Based on Compressive Sensing," *Security and Communication Networks*, vol. 2018, Article ID 1591206, 15 pages, 2018.

[43] Xia ShuangKui and Jianbin Wu, "A Modification-Free Steganography Method Based on Image Information Entropy," *Security and Communication Networks*, vol. 2018, Article ID 6256872, 8 pages, 2018.

[44] J. Zhang, B. Qu, and N. Xiu, "Some projection-like methods for the generalized Nash equilibria," *Computational Optimization and Applications*, vol. 45, no. 1, pp. 89–109, 2010.

[45] Biao Qu and Jing Zhao, "Methods for Solving Generalized Nash Equilibrium," *Journal of Applied Mathematics*, vol. 2013, Article ID 762165, 6 pages, 2013.

[46] C. Wang, C. Ma, and J. Zhou, "A new class of exact penalty functions and penalty algorithms," *Journal of Global Optimization*, vol. 58, no. 1, pp. 51–73, 2014.

[47] Y. Wang, X. Sun, and F. Meng, "On the conditional and partial trade credit policy with capital constraints: A Stackelberg Model," *Applied Mathematical Modelling*, vol. 40, no. 1, pp. 1–18, 2016.

[48] S. Lian and Y. Duan, "Smoothing of the lower-order exact penalty function for inequality constrained optimization," *Journal of Inequalities and Applications*, Paper No. 185, 12 pages, 2016.