International Journal of Electrical and Computer Engineering (IJECE)
Vol. 12, No. 3, June 2022, pp. 2526~2538
ISSN: 2088-8708, DOI: 10.11591/ijece.v12i3.pp2526-2538

A deep locality-sensitive hashing approach for achieving optimal image retrieval satisfaction

Hanen Karamti[1], Hadil Shaiba[1], Abeer M. Mahmoud[2]
[1]Computer Sciences Department, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia
[2]Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt

Article Info

Article history:

Received Feb 23, 2021
Revised Dec 27, 2021
Accepted Jan 10, 2022

Keywords:

Convolutional neural network
Deep learning
Feature extraction
Image retrieval
Locality-sensitive hashing

ABSTRACT

Efficient methods that enable high and rapid image retrieval are continuously needed, especially with the large mass of images that are generated from different sectors and domains like business, communication media, and entertainment. Recently, deep neural networks are extensively proved higher-performing models compared to other traditional models. Besides, combining hashing methods with a deep learning architecture improves the image retrieval time and accuracy. In this paper, we propose a novel image retrieval method that employs locality-sensitive hashing with convolutional neural networks (CNN) to extract different types of features from different model layers. The aim of this hybrid framework is focusing on both the high-level information that provides semantic content and the low-level information that provides visual content of the images. Hash tables are constructed from the extracted features and trained to achieve fast image retrieval. To verify the effectiveness of the proposed framework, a variety of experiments and computational performance analysis are carried out on the CIFRA-10 and NUS-WIDE datasets. The experimental results show that the proposed method surpasses most existing hash-based image retrieval methods.

This is an open access article under the CC BY-SA license.

Corresponding Author:

Hadil Shaiba
Computer Sciences Department, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University
P.O. Box 84428, Riyadh, 11671, Saudi Arabia
Email: HAShaiba@pnu.edu.sa

1. INTRODUCTION

For the last two decades, the continuous improvements in emerging technologies and the role of artificial intelligence in many domains like education, Bioinformatics, medical-informatics, biomedicine, and web crawling, caused an incredible increase in the amount of audio, images and videos. As a result of this massive amount of data, researchers are faced with a new challenge of developing accurate methods with greater efficiency and effectiveness in media indexing, retrieval, recognition, classification, as well as other areas [1]–[3]. For instance, in the banking sectors' domain; due to the outbreak of the novel virus named COVID-19; an urgent need for applying artificial intelligence techniques towards mining customers' data for authentication and verification burdens arise. In addition, the demand for decision making on daily transactions, bank customer services, front desk services and online banking, which involve a huge amount of images, increase rapidly. Accordingly, these examples of frequently needed tasks in one domain (banking sector) enrich the role of the intelligent information retrieval in general and image retrieval in focus. The content-based image retrieval (CBIR) approach performs search in large image databases, where it maps the

content of a query image to a similar query image. To sum-up, such applications are significant in various fields and specific tasks like facial recognition, visualization, authentication, and verification. [4], [5]. The color, shape or texture of an image represents the term content in CBIR. Enormous efforts based on the visual descriptors of an image were proposed in the literature to index and retrieve images. The main idea is to extract features from images to measure their similarity by calculating the mean, standard deviation, Euclidean distance [5] or other similarity measures. However, the retrieving process suffers from a key problem, which is the poor understanding of the high-level content of images. This occurs when the images retrieved do not meet the user's expectations due to the extraction of low-level features used by most similarity measures formula; this literately known as the semantic gap problem [6]. Recently, a recommended approach to eliminate the semantic gap problem is to use efficient methods for feature extraction such as deep learning techniques [7]–[9] that improve the retrieval performance by extracting the deep features of images.

In fact, many deep learning techniques were proposed, like convolutional neural networks (CNNs) that were introduced in several models of image retrieval and have reported promising results: serving as a generic descriptor in image retrieval [7], [9]–[11]. CNNs are used for deep feature extraction, where a basic CNN network [12], or a fine-tuned CNN that employs principal component analysis (PCA) whitening-based 3D model [13] is used for obtaining discriminative features. Many existing works such [8], [12], use a network with several convolutional layers as descriptors followed by fully connected (FC) layers. Although, CNNs are used to obtain the most important features from images; where these features usually are considered high-level descriptors; holding their semantic information. However, they are missing their finer-grain descriptors. In the other hand, the low-level features hold the spatial resolution of images, missing their semantic details.

As an attempt to improve the performance of CBIR systems, a hashing-based [14] technique, with either a traditional method or a deep learning architecture, was added to speed up the image retrieval process and performance and to increase its accuracy. The high-dimensional feature vectors are transferred to low-dimensional binary codes (hash codes), and the Hamming distance between hash codes is calculated to indicate the relationship between images. Accordingly, the image with the shortest distance is returned. In the literature, numerous hashing-based retrieval methods with various frameworks have been reported [14], [15]. However, there is a limitation in the reported retrieval systems such as ineffective feature extraction, weak handling of complex queries, long execution time and low accuracy, which challenge researchers to compete to propose enhanced models. One of the successful satisfied performance hashing algorithms is the local sensitive hashing, due to recognition of any small change between images [1].

Accordingly, this paper proposes a new CBIR-based system that is motivated by the literate reported efficiency of both CNN models and the locality-sensitive hashing (LSH) algorithm. Moreover, it entails different support benefits of both techniques; we focus on extracting low-level and high-level features from various network layers to overcome the drawbacks of using CNNs. Obviously, each layer concentrates on certain type of valuable features. LSH sorts images according to their similarity, which means similar images are clustered close to each other. The main contribution is to transfer learning from the pre-trained model named VGG-16 in order to extract low-level and high-level features to create hash tables that are trained. The results are then merged to improve the performance of the proposed system and reduce the retrieval computational time. The remainder of this article is laid out as follows: section 2 is a review of the literature on hashing algorithms in image retrieval. The section 3 explains the paper's contribution as well as the methodology. The acquired results are presented and discussed in section 4. Finally, section 5 brings the paper to a conclusion.

2. RELATED WORK

Hashing [16], [17] is a widespread technique in image retrieval. It is based on the conversion of high-dimensional feature vectors to low-dimensional hash codes (binary codes), which then uses the Hamming distance [16] to measure the distance between the hash codes of images. Hashing methods for CBIR were extensively studied in the literature [14], [15]; their deployment depends on the type of the used architecture which can be a traditional (using local or global descriptors) or a deep learning architecture.

Traditional hashing-based methods use handcrafted features extracted by global or local descriptors. Global features focus on color [18], texture or shape [19] to extract low-level features, whereas local descriptors focus on a particular section of an image to present more details about its visual content to extract high-level features. Histogram of color, edge histogram, color layout, Gabor filter and wavelets are examples of popular global descriptors. Examples of local descriptors include speeded up robust features (SURF), scale-invariant feature transform (SIFT) [20], points of interest (POI) detectors, Harris corner detectors, Shi-Tomasi and features from accelerated segment test (FAST) [20].

Unsupervised, semi-supervised, and supervised learning are the three types of traditional hashing algorithms. Several approaches, such as LSH [1], which randomly transfers data from high-dimensional

feature space to low-dimensional space, are presented in the first type. Improvements to this method include kernelized LSH [3] and locality-sensitive binary codes from shift-variant kernels (SKLSH) [2] that use a kernel function that improves the structure of images while disregarding their semantics. We can also include in this category the spectral hashing (SH) [8] which is a way of handling hash codes that utilizes various hash functions to reduce correlations. In addition to the asymmetric cyclical hashing (ACH) [21] which handles hash codes and reduces the storage cost.

In the semi-supervised hashing (SSH) category, several methods are proposed that not only utilize images with a few labels, but also utilize images with a set of labels. These methods generate hash codes and minimize the empirical errors between pairwise data in order to avoid over-fitting [22]. We can refer to the bootstrap sequential projection learning (BSPLH) technique as an example of SSH methods [23]. Supervised hashing is the third category and requires data to be labeled. Support vector machine (SVM), as an example, is applied to generate hash codes [9] and kernel hashing (KH) [10] is applied to generate the similarity between codes. To reduce between-data errors in the original and Hamming spaces, binary reconstructive embedding (BRE) [11] is employed. Traditional hashing-based methods have achieved good retrieval performance. However, this achievement is limited to the handcrafted features that fail to capture the semantic information from images. Deep learning networks are used as visual descriptors to extract deep features to overcome the semantic gap problem. Compared to handcrafted features, deep features are more relevant and informative and have recently achieved a significant enhancement in retrieval performance.

Hashing methods are proposed in the literature to exploit the high performance of deep neural networks. For example, Xia *et al.* [24] proposed a two-stage supervised hashing method for image retrieval. First, they proposed a scalable coordinate descent method to divide the pairwise similarity matrix into a product of two matrices and mapped each row to a hash code associated with a training image. Then, their model learns a set of hash functions, using deep convolutional network. Kang *et al.* [25] introduced a traditional supervised hashing-based model that directly learns the discrete hashing code from the semantic information. First, they constructed several columns from the semantic similarity matrix and then built the optimized hashing code. They proved in their paper that the supervised hashing recorded better accuracy than the unsupervised hashing technique. Wu *et al.* [26] presented a semi-supervised hashing method with regularized hashing and bootstrap sequential projection learning to reduce errors. The authors used a nonlinear hashing to capture the relationship among data points and reduce the dimensionality, which reduced the computational overhead. They proved the effectiveness of their experiments over six data sets.

The aforementioned techniques apply one type of feature extraction (mostly high-level features). To create a more thorough description, multiple types of features should be extracted. Several approaches for retrieving multi-level images are proposed. Zhao *et al.* [27] implemented a deep semantic ranking method for learning hash functions that hold multi-level semantic similarity between multi-label images. In their model, they mapped the deep convolutional neural network to hash functions then to hash codes. This result in a ranking list guides the learning process. They used a surrogate loss function for optimization and proved their superiority results. Lai *et al*. 2015 [28] designed a deep architecture for supervised hashing, and deep neural networks. The authors presented their model in three phases. First, they built a sub-network with a stack of CNNs for intermediate features. Second, they applied a divide-and-encode module to divide these features into several hash branches. Finally, a triplet ranking loss was designed for optimization.

Lin *et al.* 2017 [29] presented a new discriminative deep hashing (DDH) network for image retrieval. The authors unified the end-to-end, the divide-and-encode and the desired discrete code learning modules. Then they benefited from the stack of CNN-pooling layers to obtain multi-scale features. Prior to that, they merged the results of layers three and four. They finally optimized their results using a suitable loss function. Ng *et al.* 2020 [30] introduced a new multi-level supervised hashing algorithm for image retrieval systems that is integrated with the CNN deep framework. The authors instead of generating a complementarity multi-level hash tables for feature extraction from different layers of the CNN deep network; they constructed and trained these tables individually using different levels of features (semantic and structural). They reported improved performance on three databases.

The main challenges while developing a CBIR-based system are reducing the semantic gap, achieving higher accuracy, minimizing the computation complexity, and subsequently the time to train and obtain results from testing as well as evaluating the proposed model. Accordingly, the proposed work in this paper focuses on these challenges and was motivated by solving optimality issues of performance. The LSH algorithm is embedded with aim of optimization the unsupervised learning efficiency of CNNs. First, seven blocks of CNNs are used to extract low level and high level features corresponding to logical and global contents. These separately extracted features space were flatten and converted to hash codes for simplicity, reducing the search space burden and speed up of the retrieval task. Actually, the idea of hashing algorithm proved efficiency long time ago in replacing difficulties in searching by data itself rather than assuming codes for each data element and instead search such codes for reducing linear complexity. Local sensitive

hashing is used as it is a very sensitive to tiny difference in the input data; even a single bit will change the hash value and this accordingly increases reliability.

Hamming distance were used as an efficient measure that accurately reflect images difference from each other, where two images are identical or perceptually similar if the distance between both equal zero otherwise both are different relative to the value obtained. Late fusion is used to merge such obtained hashed features optimally with eliminating redundancy and reducing the state space. The same query image is tested for various combinations of layers and transformations hashing. The proposed technique is invariant and showed significant performance as depicted later in this paper. The main contribution of this study is summarized in the following points:

- Achieving high performance in image retrieval with low execution time. We will show that our model is capable of achieving high performance on different databases.
- Our proposed method achieves high performance by extracting low-level and high-level features. Further improvement is achieved by using fusion on high-level features extracted from pre-trained models.
- Low execution time is achieved through the use of LSH method, which allows a fast retrieval of images in a very large search space. With LSH, we were able to extract more features (low-level and high-level features) in less time.

3. METHOD

The proposed method is presented in Figure 1 that displays all the steps from feature extraction to calculating the similarity and displaying the results. The CNN builds the layers of the proposed network in order to extract features from different prospective. We extract low-level and high-level features from images to preserve their local and global properties. To do that, the CNN model is divided into L blocks, and the last CNN-code is flattened to give a feature vector.

The low-level features are extracted from the first convolutional blocks, and the high-level features are extracted from the last convolutional blocks. We denote the features extracted in the middle blocks by medium-level features. Then, the LSH is applied on each features set to generate different hash codes. The hashing representation is used in several works from the literature [14], [15] to fasten the image retrieval process as each feature is recognized by a binary representation. Then the hamming distance metric is applied to create the result list that contains the similarity score between the query and the images in the database. This result list from measuring hamming distance is then sorted to reflect rank of the images where small values are sorted on top of the list that intensively reflects the closer to the image query. Finally, the obtained lists from each block are merged using the late fusion technique according to their rank to enhance the retrieval performance.

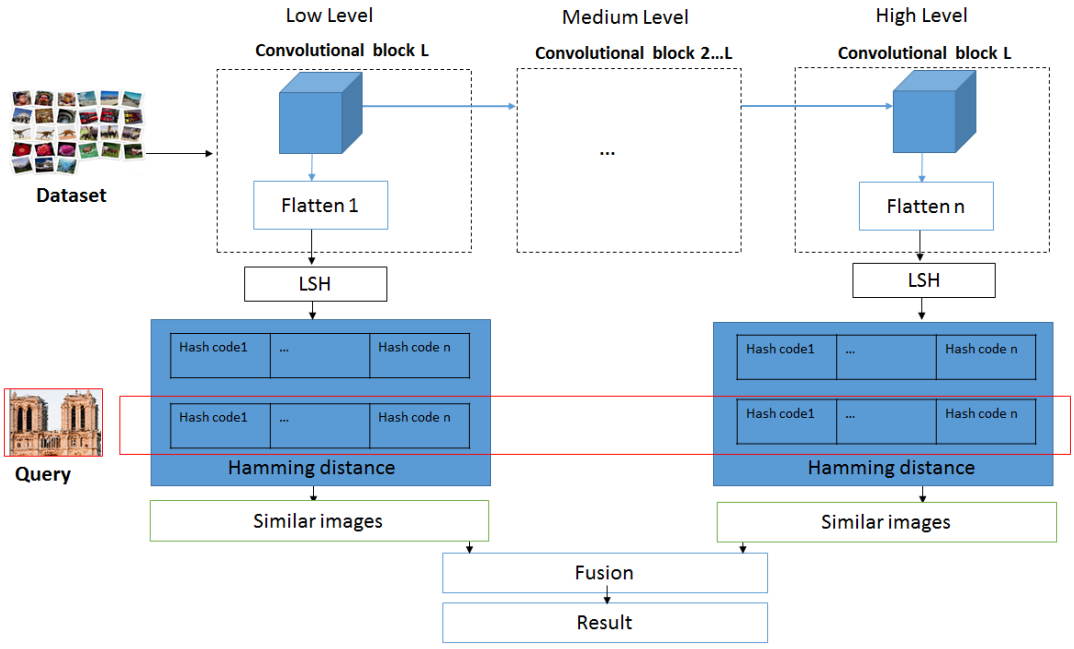


Figure 1. The proposed model diagram

3.1. CNN construction and feature extraction

The proposed model as shown in Figure 2, for the model architecture and Table 1 for CNN setting parameters) is divided into L blocks, where each one contains a set of convolutional layers, and the two last blocks represent the first and second fully connected layers, where each one contains 4096 units. The last fully connected layer is reserved to classify the units into C classes. All the fully connected layers adopt the rectified linear activation function (1).

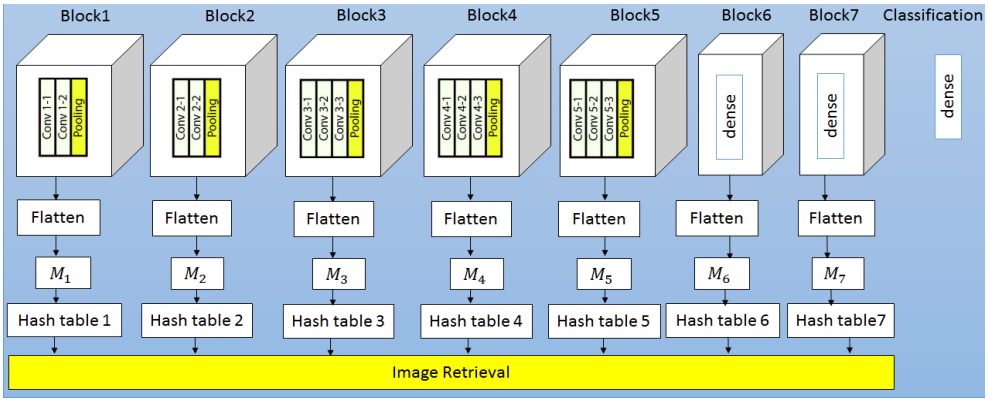


Figure 2. The proposed CNN model

Table 1. VGG-16 pre-trained convolutional neural network architecture.

Layer Type	Output Size	Layer Type	Output Size
Input image	224x224x3	Convolutional layer	28x28x512
Convolutional layer	224x224x64	Convolutional layer	28x28x512
Convolutional layer	224x224x64	Max pooling	14x14x512
Max pooling	112x112x64	Convolutional layer	14x14x512
Convolutional layer	112x112x128	Convolutional layer	14x14x512
Convolutional layer	112x112x128	Convolutional layer	14x14x512
Max pooling	56x56x128	Max pooling	7x7x512
Convolutional layer	56x56x256	Fully connected layer	1x1x4096
Convolutional layer	56x56x256	Fully connected layer	1x1x4096
Convolutional layer	56x56x256	Fully connected layer	1x1x1000
Max pooling layer	28x28x256	SoftMax layer	1x1x1000
Convolutional layer		28x28x512	

$$f(x) = \max(0, x) \quad (1)$$

Let *I* be the set of *N* images of training from *C* classes, the proposed model extracts features selected from various blocks and constructs a set of features. Assume a convolutional block produces A feature maps as its output each with a height *H* and a width *W*. Thus, an image is represented as an H×W×A-dimensional vector, allowing the model to contain a flatten layer to convert the data into a 1-dimensional feature map. Algorithm 1 represents more details about the feature extraction process, where N and |L| represent respectively the number of images and blocks and d_j is the dimension of features for each block j∈L.

Algorithm 1: Pseudo-code for low-level and high-level feature extraction from VGG-16 CNN

```
Initialize: I: set of images, N: number of images, L: set of blocks, d_j: the feature dimension
For each j∈{1,2,…,|L|} do
    For each i∈{1,2,…,N} do
        For each e∈{1,2,…,d_j} do
F_j^{i,e} <− Extract features from the convolutional block j for image i
        End For
M_j <− F_j^i //Construction of the feature matrix for each block
End For
    M<−M_j
End For
Return M
```

Let $M=\{M_1,M_2,\ldots M_j,\ldots,M_L\}$, the feature matrix, where each M_j represents the set of features F_j^iextracted for each image i in the block j, and d represents the number of features for each matrix feature F_j^{i,d_j}, where $j\epsilon[1,|L|]$ and $i\epsilon[1,N]$.

$$M_j = \begin{pmatrix} F_j^{1,1} & \ldots & F_j^{1,d_j} \\ F_j^{2,1} & \ldots & F_j^{2,d_j} \\ . & \ldots & . \\ F_j^{N,1} & \ldots & F_j^{N,d_j} \end{pmatrix} \quad (2)$$

In our model, we have seven blocks; thus, the feature matrices M=$\{M_1,M_2,\ldots,M_7\}$, are referred to set of $\{M_1,M_2,\ldots,M_5\}$ represents the feature matrices extracted from the first to the fifth convolutional blocks and $M_6$and M_7 represent the feature matrices obtained from the first and second fully-connected layers. Features of each block serve as the input to an unsupervised LSH algorithm to train the corresponding hash table, which encodes each feature matrix M_j on k-binary code b_j, where k is the dimension of the binary code, to build the Hamming space.

3.2. Locality sensitive hashing for retrieving similar images

Locality-sensitive hashing (LSH) idea is that instances that are similar and close to each other will be located in the same bucket. LSH maps the points from a high-dimensional space into a low-dimensional space, which in turn creates hash codes for each vector in the search space. For each block j, and each feature matrix row, we apply the method LSH. The locality-sensitive function indicates that two images are close if they have a high probability that indicates the similarity of their hash code, and they are distant if they have a low probability. LSH family includes a set of hash functions H {h: S→U} (where U represents the universe and S represents a set of elements from U) that is sensitive of type (r1, r2, p1, p2) with r1<r2 and p1>p2. If we have the following priorities:

$$\forall\, p \in B(q,r_1), then\ Pr_{h\in H}[h(q)=h(p)] \geq p1 \quad (3)$$

$$\forall\, p \in B(q,r_2), then\ Pr_{h\in H}[h(q)=h(p)] \geq p2 \quad (4)$$

where, *B(q, r)* is the bucket of center *q* and radius *r*. Multiple hash tables hi are proposed according to L hashing functions. So, all the points are stocked in L different hashing tables. The algorithm is parameterized with k number of dimensions that are hashed. Each function h_i is defined by two vectors:

$$D_i =< D_0^i, D_1^i, \ldots, D_{k-1}^i > \quad (5)$$

$$T_i =< t_0^i, t_1^i, \ldots, t_{k-1}^i > \quad (6)$$

The values $D_i \epsilon [0, z_j - 1]$ are randomly chosen, where z_j is the space dimension and the values of $T_i \epsilon[0, C]$represent the thresholds where C is the largest coordinate of all the points. Each function h_i projects p point from $[0,C]^{d,z_j}$ in $[0,2^k-1]$ in order that $h_i(p)$ will be calculated as a linked list of k-bits named $b_0^i, b_2^i,\ldots b_a^i,\ldots\ b_{k-1}^i$ where b_a^i is defined by (7):

$$b_a^i = \begin{cases} 0 & if\ (P_{D_a^i} < t_a^i) \\ 1 & else \end{cases} \quad (7)$$

We denote $P_{D_a^i}$ the coordinate of p by the dimension of D_a^i. The list k bits are the hash key in the i^{th} case in the hashing table. To search for the nearest-neighbor points to a query q, we calculate its hash key for each table. Then, we apply a linear search on the points of the corresponding cases. The parameters L and k allow choosing between rapidity and precision. k can obtain a large value (e.g., k=32); thus, the hashing functions' space can be very large and expensive in memory. In addition, to avoid the collision problem, we add a second hashing function to project the result of function h_i in a small domain calculated on each list of k-bits. Before its application, we reduce the number of k to W. To do so, we calculate the most distinctive W dimensions, where W is less than the feature dimensioning (W<z_j). The vector of these W dimensions is used to generate the hash keys of these points. The main idea is that two points that share the same distinctive dimensions have a high probability of being close. Intuitively, a point is distinctive along one dimension if it

is far from the mean value according to this dimension. Another idea is that the dimensions with high variance are the most distinctive.

Let M_j^i, the feature vector of image i that represents the row I in the matrix M_j of block j be defined by: $M_j^i=\{F_j^{i,1}, F_j^{i,1}, \dots, F_j^{i,z_j}\}$, and β the function that measures the distinctiveness of the feature matrices by (8):

$$\beta(x_a^i) = \left|\overline{x_a} - x_a^i\right|\sigma_a^\alpha \tag{8}$$

where, $\overline{x_a}$ is the average value of features according to the dimension a, σ_a is the deviation and α=0.5 represents the weight of the deviation. For point x^i, we denote $D(x^i) =< D_1(x^i), D_2(x^i), \dots, D_{z_j}(\boldsymbol{x^i}) >$ the vector of dimensions (from 1 to $\boldsymbol{z_j}$) sorted in descending order of distinctiveness:

$$\beta\left(x^i_{D_1(x^i)}\right) > \beta\left(x^i_{D_2(x^i)}\right) > \cdots > \beta\left(x^i_{D_{z_j}(x^i)}\right) \tag{9}$$

Thus, $D_1(x^i)$ is the dimension along in, and the point x^i is the most distinctive. $D_{D_{z_j}}(x^i)$ is the dimension that recognizes that the point is the least distinctive. The idea behind the proposed structure is that if two points q and $x^i$are close, the W first values of their vectors D(q) and D(x^i) will be identical (or almost identical, e.g., order may vary). Therefore, the final hash table denoted by H contains W-dimensions where each dimension is indexed by an integer between 1 to z_j. For a point x^i, the W first values of D(x^i), sorted in ascending order, form a vector are denoted by (10):

$$D'(x^i) =< a_1, a_2, \dots, a_W > where\ 1 \le a_1 < a_2 < \cdots < a_W \le z_j \tag{10}$$

so, the point x^i is saved in the case $H[a_1][a_2] \dots [a_W]$. In this level we apply the second hashing function h' that projects the w-dimension interval of [0...c] and is defined as (11):

$$h'\left(D'(x^i)\right) = \left(\left(\sum_{i=1}^{W} r_i a_i\right) mod\ P\right) mod\ c \tag{11}$$

where, P is a prime number and r_i are random integers. In fact, the hash table has only one dimension. Therefore, the point x^i is saved in a new hash table $H'[h'(D_{x_i})]$. The objective of the use of h' is to introduce new collisions, have not taken place with table H. Algorithm 2 represents the proposed hashing methods.

Algorithm 2: Pseudo-code for the proposed LSH-for block j

```
Initialize: M_j:matrix of feature, N:training image, j block, z_j:block dimension, K:code,
length K
For each i∈{1,2,…,z_j } do
    For each a∈{1,2,…,N} do
         For each point x_i=F_j^{i,a} do
          h(x_i)←Compute hash code for F_j^{i,a} //Compute Hash code of each point (x_i)
            H(x_i) ←h(x_i)//the hash tables
W ←determinate the distinctive dimension
            Determinate (D'(x_i))
             h'(D'(x_i))←Compute hash code for D'(x_i)/Compute Hash code of each dimension
D'(x_i)
H'(D'(x_i)) ←h' (D'(x_i))
        End For
End For
End For
```

3.3. Retrieving similar images

In the retrieval phase, when a new query image q arrives, the hash code for q is computed. As a result, the query will have a set of hash codes related to the CNN block's hash functions. Once the hash codes are generated, the similarity measure that is based on the Hamming distance is calculated between the query hash code and every database's image hash code found in their hash tables. Then, similar images are retrieved and saved in a result list. This step is repeated for all blocks. Finally, the retrieved images are combined using fusion by rank where images that are repeated the most are more likely to be selected and images that are less repeated are less likely to be selected.

We employ late fusion as per rank to integrate the obtained results, which calculates the average position of every image within the result list [31]. In our case, we have seven scored lists comprising k related images for the query.

$$Rank(img) = (Weight * nbBlock - Score(img) + \frac{RankC(img)}{Score(img)}) \quad (12)$$

where, *Weight* is a weight defined by *W=(2∗k)+1*. *k* is the count of chosen closest neighbors. *nbBlock* is the number of the convolutional block that is equal to seven. *RankC(Img)* is the integration of the image's rank *Img*. *freq(Img)* is the rate of image's occurrence *Img*.

4. EXPERIMENTS AND RESULTS

4.1. Data collection

We ran numerous experiments on two datasets to evaluate the suggested technique named: National University of Singapore-Web Image Dataset (NUS-WIDE) as shown in Figure 3(a) and Canadian Institute for Advanced Research (CIFAR-10) as shown in Figure 3(b). We implemented the proposed method using Keras and TensorFlow, and our workspace has the Intel Core i7 CPU and 32 GB memory. NUS-WIDE dataset contains 269,648 images where each one is presented by a size of 64×64 grouped in 81 categories. Each image is associated to one or more groups. Following some previous works [14], in this paper, we use the 21 most categories, with approximately 5000 images in each category. Therefore, there are 157, 465 images in total. The input of the proposed model is the pixel-based images, and the input of the traditional hashing methods is the GIST features with 512 dimensions. The GIST descriptor was initially proposed in [32] and its idea is to create a local low-level representation without segmentation.

CIFAR-10 database is composed of 10 categories where each contains 60,000 (50,000 training and 10,000 testing) images with a single label. Each image is represented by a 32×32 color image. The input of the proposed model is the raw pixel-based images, and for the traditional hashing methods, the inputs are GIST features with 512 dimensions.

For evaluation, we used the mean average precision (MAP). For comparison, we compared the proposed method with two sets of state-of-the-art works. The first set includes the following eight non-deep hashing methods: iterative quantization (ITQ) [33], principal component analysis hashing (PCAH) [34], locality sensitive hashing (LSH) [17], density sensitive hashing (DSH) [16], spherical hashing (SPH) [35], spectral hashing (SH) [36], discrete graph hashing (AGH) [37], and sparse embedding and least variance encoding (SELVE) [38]. The other set includes the following five deep-hashing methods: deep hashing (DH) [39], Deepbit [40], unsupervised hashing with binary deep neural network (UH-BDNN) [41], semantic structure-based unsupervised deep hashing (SSDH) [42], and stochastic generative hashing (SGH) [43]. All these techniques are unsupervised image retrieval methods.

(a) (b)

Figure 3. Examples of (a) NUS-WIDE and (b) CIFAR-10 datasets

4.2. Results

To evaluate the performance of the proposed hashing method, we have adopted the evaluation of the number of hash tables. Therefore, we implement our method using 1, 5, 10, 50,100,150, and 200 hash tables. For each version, We used hash codes of 8, 16, 24, 32, and 64 bits. For the NUS-WIDE and CIFAR-10 datasets, we employed MAP@100 to assess our model's performance. We have included the query execution time results alongside the MAP results. The MAP results are shown for NUS-WIDE dataset in Figure 4 and

for CIFAR-10 dataset in Figure 5. The query execution time results are shown in Tables 2 and 3 for NUS-WIDE and CIFAR-10 datasets respectively.

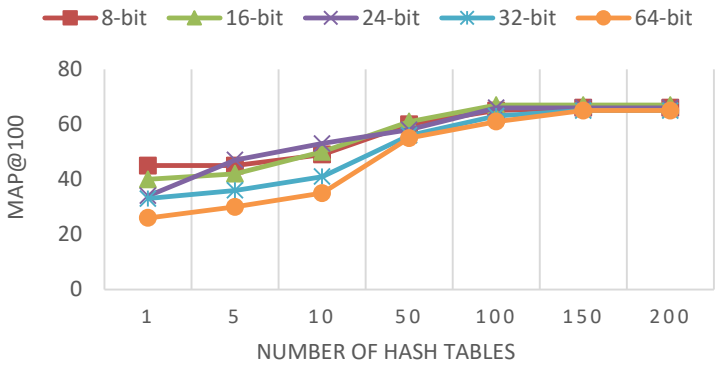


Figure 4. MAP of different versions of the proposed hashing method with different number of hash bits and different number of hash tables on NUS-WIDE dataset

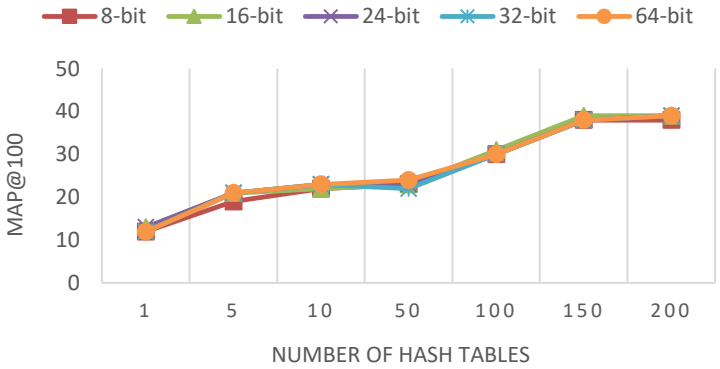


Figure 5. MAP of different versions of the proposed hashing method with different number of hash bits and different number of hash tables on CIFAR-10 dataset

Table 2. Query time (milliseconds) result on NUS-WIDE dataset

Number of hash tables	8-bit	16-bit	24-bit	32-bit	64-bit
50	498.32	25.44	11.16	1	1.65
100	975.24	41.29	7.22	3.27	4.44
150	1424.11	50.28	14.13	7.21	10.24
200	1967.17	88.18	43.57	10.59	12

Table 3. Query time (milliseconds) result on CIFAR-10 dataset

Number of hash tables	8-bit	16-bit	24-bit	32-bit	64-bit
50	513.19	31.47	25.45	2.41	3.05
100	1065.23	55.23	12.22	6.27	8.13
150	1624.45	78.09	16.5	10.54	13.22
200	2134.33	100.01	61.11	12.59	14.14

The proposed method has provided better query time when it uses 64-bits compared to when it uses 12, 16, 24 and 32 bits. The query time varies with the increasing number of bits where we have gained about 10 times faster querying samples when the number of bits increases. This remark is applicable for both datasets. Based on the MAP results, when applying the proposed hashing method to retrieve similar images, we obtained better MAP scores where the number of bits is 16 and the number of hash tables is 150 for

NUS-WIDE (MAP=67%) and CIFAR-10 datasets (MAP=39%). The query time increased where the number of bits increased, so we got better efficiency when the number of bits is equal to 32-bits. Indeed, the query time, using 150 hash tables and 32-bits, is 7.21 and 10.54 for NUS-WIDE and CIFAR-10 respectively, and by using 150 hash tables and 16-bits, the query time is 50.28 and 78.09 for NUS-WIDE and CIFAR-10 respectively. Comparing the 16-bit hash code with the 32-bit MAP results, we got 65% and 38% for NUS-WIDE and CIFAR-10 respectively. Choosing the number of bits is challenging as we want to choose the method that gives a better MAP result and enhance the query retrieval time. In our case, putting in mind that our main objective is to retrieve similar images and given that the difference between different methods' retrieval times is <50 ms, we decided to trade-off and use a 16-bit hash code.

We now compare the fusion of the features extracted from the seven convolutional blocks. Tables 4 and 5 display the results obtained by each block using 150 hash tables from 8 to 64-bits using both datasets. The integration of different types of features extracted from several levels of VGG16 delivers superior results when compared to a single feature representation. Different systems from the literature [10], [11] used only block seven or block six as a high-level feature to retrieve images as it represents the last layer from the CNN prior to the classification layer. The first blocks (from block 1 to block 3) represent many feature-maps and are considered as low-level features because they display more information about the color. Block 4 and 5 are considered as a middle-level feature as they represent an intermediate level between two levels of features. So, blocks 6 and 7 represent the MAP values of features extracted from the first and second fully connected layers with different hash bits respectively. Similarly, blocks 1, 2, 3, 4 and 5 denote the first, second, third, fourth, and fifth convolutional blocks respectively. In Tables 5 and 6, the MAP values of using features from the corresponding block are displayed as a comparison, and the MAP of the proposed method that is noted fusion in the tables represent the late fusion by rank between the result lists of each block. The results show that the best performance is achieved by using fusion (the proposed method) regardless of the number of bits used in both datasets.

Table 4. MAP of hashing with different number of hash bits and different convolutional blocks on NUS-WIDE

Convolutional Block	8-bit	16-bit	24-bit	32-bit	64-bit
1	24%	53%	49%	45%	50%
2	33%	54%	47%	40%	32%
3	51%	57%	56%	58%	37%
4	53%	57%	55%	53%	39%
5	64%	65%	59%	54%	41%
6	59%	59%	61%	54%	46%
7	69%	61%	66%	55%	43%
Fusion	66%	67%	66%	65%	65%

Table 5. MAP of hashing with different number of hash bits and different convolutional blocks on CIFAR-10

Convolutional Block	8-bit	16-bit	24-bit	32-bit	64-bit
1	13%	17%	10%	12%	14%
2	16%	12%	14%	15%	17%
3	21%	26%	29%	29%	31%
4	25%	21%	28%	27%	27%
5	23%	24%	28%	26%	24%
6	32%	38%	31%	31%	32%
7	31%	34%	37%	35%	31%
Fusion	38%	39%	38%	38%	38%

4.3. Comparison with state-of-the-art hashing methods

To reveal the efficiency of the proposed method, a comparison between several state-of-the-art hashing methods is given in Tables 6 and 7 for NUS-WIDE, and CIFAR10 respectively, with numbers in the hash code that range from 16 to 64 bits. When we compare the suggested hashing method to existing hashing methods, we can find that in most scenarios, our method surpasses the others. This may be caused by the quality of the extracted features. Our method surpasses the other deep hashing methods, see Tables 6 and 7.

When we compare traditional hashing methods to deep-hashing methods, we notice that deep-hashing techniques outperform the first type of approaches in terms of MAP scores. That might be because typical hashing approaches don't completely use the representation ability of deep networks and may achieve unsatisfactory performance by over-fitting due to bad local minima. While, our deep hashing method generates promising outcomes by utilizing local and global structures.

Table 6. MAP of hashing with different number of hash bits on NUS-WIDE

Methods	16-bit	32-bit	48-bit	64-bit	Methods	16-bit	32-bit	48-bit	64-bit
ITQ [33]	0.51	0.51	0.51	0.52	SELVE [38]	0.46	0.46	0.44	0.43
PCAH [34]	0.41	0.39	0.38	0.37	DH [39]	0.56	0.52	0.51	0.45
LSH [17]	0.41	0.4	0.42	0.41	Deepbit [40]	0.4	0.4	0.43	0.46
DSH [16]	0.5	0.49	0.49	0.51	UH-BDNN [41]	0.47	0.47	0.47	0.48
SPH [35]	0.41	0.45	0.47	0.47	SSDH [42]	0.66	0.66	0.67	0.67
SF [36]	0.34	0.35	0.36	0.36	SGH [43]	0.49	0.49	0.48	0.48
AGH [37]	0.56	0.52	0.51	0.47	Our method	0.67	0.66	0.65	0.65

Table 7. MAP of hashing with different number of hash bits on CIFAR

Methods	16-bit	32-bit	48-bit	64-bit	Methods	16-bit	32-bit	48-bit	64-bit
ITQ [33]	0.31	0.32	0.33	0.34	SELVE [38]	0.3	0.28	0.26	0.23
PCAH [34]	0.21	0.18	0.17	0.16	DH [39]	0.19	0.19	0.19	0.18
LSH [17]	0.17	0.21	0.21	0.24	Deepbit [40]	0.2	0.2	0.22	0.24
DSH [16]	0.24	0.26	0.28	0.29	UH-BDNN [41]	0.26	0.28	0.28	0.29
SPH [35]	0.2	0.26	0.28	0.29	SSDH [42]	0.24	0.25	0.25	0.25
SF [36]	0.18	0.18	0.17	0.16	SGH [43]	0.16	0.17	0.18	0.18
AGH [37]	0.3	0.26	0.25	0.23	Our method	0.39	0.38	0.38	0.38

5. CONCLUSION

This paper presented a new unsupervised deep hashing method for image retrieval based on the use of LSH and the local and global features that are extracted from CNN architecture. Firstly, we calibrate our network using the VGG16 model that is divided into seven convolutional blocks. Then, we extract the features from each block where the first block corresponds to the low-level features and the two last blocks correspond to the fully-connected layers. Secondly, we created the hash tables using the LSH method. The hash tables are created for each point feature and for each distinctive dimension corresponding to this point. Thirdly, when a query arrives, the similarity is calculated between the query hash tables and the images' hash tables using the hamming distance, where all the hash tables are a binary representation according to a 16-bit hash code that speed up the retrieval process. All the previous steps are repeated for each convolutional block to obtain a result list for m each block. Finally, we combined the obtained result lists using the late fusion method which depends in its calculation on the rank and the score of each image result. The experimental results were performed out on two benchmark datasets CIFAR-10 and NUS-WIDE. It demonstrated that the proposed method surpasses other state-of-the-art hashing methods. Using 16 bits, the proposed method achieves mean average precisions equal to 0.39 and 0.67 respectively on CIFAR-10 and NUS-WIDE. In future work, we intend investigating a new supervised hashing method.

ACKNOWLEDGEMENTS

This research was funded by the Deanship of Scientific Research at Princess Nourah bint Abdulrahman University through the Fast-track Research Funding Program.

REFERENCES

[1] J. Y.-H. Ng, F. Yang, and L. S. Davis, "Exploiting local features from deep networks for image retrieval," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2015, pp. 53–61, doi: 10.1109/CVPRW.2015.7301272.

[2] D. Giveki, M. A. Soltanshahi, and G. A. Montazer, "A new image feature descriptor for content based image retrieval using scale invariant feature transform and local derivative pattern," *Optik*, vol. 131, pp. 242–254, Feb. 2017, doi: 10.1016/j.ijleo.2016.11.046.

[3] J.-M. Guo, H. Prasetyo, and J.-H. Chen, "Content-based image retrieval using error diffusion block truncation coding features," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 3, pp. 466–481, Mar. 2015, doi: 10.1109/TCSVT.2014.2358011.

[4] J. Zhou, X. Liu, W. Liu, and J. Gan, "Image retrieval based on effective feature extraction and diffusion process," *Multimedia Tools and Applications*, vol. 78, no. 5, pp. 6163–6190, Mar. 2019, doi: 10.1007/s11042-018-6192-1.

[5] A. M. Mahmoud, H. Karamti, and M. Hadjouni, "A hybrid late fusion-genetic algorithm approach for enhancing CBIR performance," *Multimedia Tools and Applications*, vol. 79, no. 27–28, pp. 20281–20298, Jul. 2020, doi: 10.1007/s11042-020-08825-6.

[6] Y. Liu, D. Zhang, G. Lu, and W.-Y. Ma, "A survey of content-based image retrieval with high-level semantics," *Pattern Recognition*, vol. 40, no. 1, pp. 262–282, Jan. 2007, doi: 10.1016/j.patcog.2006.04.045.

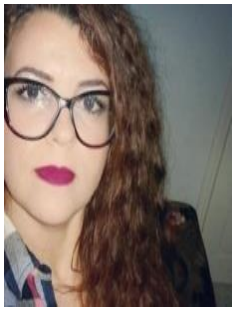
[7] L. Deng, "A tutorial survey of architectures, algorithms, and applications for deep learning," *APSIPA Transactions on Signal and Information Processing*, vol. 3, Jan. 2014, doi: 10.1017/atsip.2013.9.

[8] A. Gordo, J. Almazán, J. Revaud, and D. Larlus, "Deep image retrieval: learning global representations for image search," in *Computer Vision textendash ECCV 2016*, Springer International Publishing, 2016, pp. 241–257.

[9] W. Huang and Q. Wu, "Image retrieval algorithm based on convolutional neural network," in *Current Trends in Computer Science and Mechanical Automation Vol.1*, De Gruyter Open, 2017, pp. 304–314.
[10] P. Wu, S. C. H. Hoi, H. Xia, P. Zhao, D. Wang, and C. Miao, "Online multimodal deep similarity learning with application to image retrieval," in *Proceedings of the 21st ACM international conference on Multimedia-MM '13*, 2013, pp. 153–162, doi: 10.1145/2502081.2502112.
[11] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural codes for image retrieval," in *Lecture Notes in Computer Science*, Springer International Publishing, 2014, pp. 584–599.
[12] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," in *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, Jun. 2014, pp. 512–519, doi: 10.1109/CVPRW.2014.131.
[13] F. Radenovic, G. Tolias, and O. Chum, "Fine-tuning CNN image retrieval with no human annotation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 7, pp. 1655–1668, Jul. 2019, doi: 10.1109/TPAMI.2018.2846566.
[14] F. Sabahi, M. O. Ahmad, and M. N. S. Swamy, "Content-based image retrieval using perceptual image hashing and hopfield neural network," in *2018 IEEE 61st International Midwest Symposium on Circuits and Systems (MWSCAS)*, Aug. 2018, pp. 352–355, doi: 10.1109/MWSCAS.2018.8623902.
[15] M. Zareapoor, J. Yang, D. K. Jain, P. Shamsolmoali, N. Jain, and S. Kant, "Deep semantic preserving hashing for large scale image retrieval," *Multimedia Tools and Applications*, vol. 78, no. 17, pp. 23831–23846, Sep. 2019, doi: 10.1007/s11042-018-5970-0.
[16] Z. Jin, C. Li, Y. Lin, and D. Cai, "Density sensitive hashing," *IEEE Transactions on Cybernetics*, vol. 44, no. 8, pp. 1362–1371, Aug. 2014, doi: 10.1109/TCYB.2013.2283497.
[17] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," *Communications of the ACM*, vol. 51, no. 1, pp. 117–122, Jan. 2008, doi: 10.1145/1327452.1327494.
[18] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2015, pp. 27–35, doi: 10.1109/CVPRW.2015.7301269.
[19] Y. Wu and Y. Wu, "Shape-based image retrieval using combining global and local shape features," in *2009 2nd International Congress on Image and Signal Processing*, Oct. 2009, pp. 1–5, doi: 10.1109/CISP.2009.5304693.
[20] X. Yuan, J. Yu, Z. Qin, and T. Wan, "A SIFT-LBP image retrieval model based on bag-of-features," *International conference on image processing*, pp. 1061-1164, 2011.
[21] G. Ciocca, S. Corchs, and F. Gasparini, "Genetic programming approach to evaluate complexity of texture images," *Journal of Electronic Imaging*, vol. 25, no. 6, Jul. 2016, doi: 10.1117/1.JEI.25.6.061408.
[22] A.-B. M. Salem and A. M. Mahmoud, "A hybrid genetic algorithm-decision tree classifier," in *Intelligent Information Processing and Web Mining*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 221–232.
[23] A. B. Yandex and V. Lempitsky, "Aggregating local deep features for image retrieval," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1269–1277, doi: 10.1109/ICCV.2015.150.
[24] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan, "Supervised hashing for image retrieval via image representation learning," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, 2014.
[25] W.-C. Kang, W.-J. Li, and Z.-H. Zhou, "Column sampling based discrete supervised hashing," in *AAAI'16: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, 2016, pp. 1230–1236.
[26] C. Wu, J. Zhu, D. Cai, C. Chen, and J. Bu, "Semi-supervised nonlinear hashing using bootstrap sequential projection learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1380–1393, Jun. 2013, doi: 10.1109/TKDE.2012.76.
[27] Fang Zhao, Y. Huang, L. Wang, and Tieniu Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1556–1564, doi: 10.1109/CVPR.2015.7298763.
[28] H. Lai, Y. Pan, Ye Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 3270–3278, doi: 10.1109/CVPR.2015.7298947.
[29] J. Lin, Z. Li, and J. Tang, "Discriminative deep hashing for scalable face image retrieval," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Aug. 2017, pp. 2266–2272, doi: 10.24963/ijcai.2017/315.
[30] W. W. Y. Ng, J. Li, X. Tian, H. Wang, S. Kwong, and J. Wallace, "Multi-level supervised hashing with deep features for efficient image retrieval," *Neurocomputing*, vol. 399, pp. 171–182, Jul. 2020, doi: 10.1016/j.neucom.2020.02.046.
[31] H. Karamti, M. Tmar, M. Visani, T. Urruty, and F. Gargouri, "Vector space model adaptation and pseudo relevance feedback for content-based image retrieval," *Multimedia Tools and Applications*, vol. 77, no. 5, pp. 5475–5501, Mar. 2018, doi: 10.1007/s11042-017-4463-x.
[32] M. Douze, H. Jégou, H. Sandhawalia, L. Amsaleg, and C. Schmid, "Evaluation of GIST descriptors for web-scale image search," 2009, doi: 10.1145/1646396.1646421.
[33] 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, Dec. 2013, doi: 10.1109/TPAMI.2012.193.
[34] X.-J. Wang, L. Zhang, F. Jing, and W.-Y. Ma, "AnnoSearch: image auto-annotation by search," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)*, vol. 2, pp. 1483–1490, doi: 10.1109/CVPR.2006.58.
[35] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-E. Yoon, "Spherical hashing," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 2957–2964, doi: 10.1109/CVPR.2012.6248024.
[36] Y. Weiss, A. Torralba, and R. Fergus, "Spectral hashing," in *Advances in Neural Information Processing Systems*, 2009, vol. 21.
[37] W. Liu, C. Mu, S. Kumar, and S.-F. Chang, "Discrete graph hashing," in *Advances in Neural Information Processing Systems*, 2014, vol. 27.
[38] X. Zhu, L. Zhang, and Z. Huang, "A sparse embedding and least variance encoding approach to hashing," *IEEE Transactions on Image Processing*, vol. 23, no. 9, pp. 3737–3750, Sep. 2014, doi: 10.1109/TIP.2014.2332764.
[39] V. E. Liong, Jiwen Lu, G. Wang, P. Moulin, and J. Zhou, "Deep hashing for compact binary codes learning," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 2475–2483, doi: 10.1109/CVPR.2015.7298862.
[40] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 1183–1192, doi: 10.1109/CVPR.2016.133.
[41] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *Computer Vision textendash ECCV 2016*, Springer International Publishing, 2016, pp. 219–234.

[42] E. Yang, C. Deng, T. Liu, W. Liu, and D. Tao, "Semantic structure-based unsupervised deep hashing," in *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, Jul. 2018, pp. 1064–1070, doi: 10.24963/ijcai.2018/148.
[43] B. Dai, R. Guo, S. Kumar, Ni. He, and L. Song, "Stochastic generative hashing," in *ICML'17: Proceedings of the 34th International Conference on Machine Learning*, 2017, vol. 70, pp. 913–922.

BIOGRAPHIES OF AUTHORS



Hanen Karamti completed a Bachelor of Computer Science and Multimedia at ISIMS (High Institute of Computer Science and Multimedia of Sfax) University, Tunisia. She completed a Master of Computer Science and Multimedia at the same University. She obtained her Ph.D. degree from the Computer Science from the National Engineering School of Sfax (University of Sfax, Tunisia), in cooperation with the university of La Rochelle (France) and the university of Hanoi (Vietnam). Karamti is now an assistant professor at Princess Norah bint Abdulrahman University, Riyadh, Saudi Arabia. Her areas of interest are information retrieval, multimedia systems, Image retrieval, health informatics, big data, and data analytics. She can be contacted at email: HMkaramti@pnu.edu.sa.

Hadil Shaiba holds a Ph. D and M.S. in Computer Science from Southern Methods University, USA, and a B.S. in Information Technology from King Saud University, KSA. Hadil is an Assistant Professor in the College of Computer and Information Sciences at Princess Nourah bint Abdulrahman University, KSA. Her research interests include data mining and machine learning methods for applications in meteorology and medicine. She can be contacted at email: HAShaiba@pnu.edu.sa.

Abeer M. Mahmoud received her Ph.D. (2010) in Computer science from Niigata University, Japan, her M. Sc (2004) B.Sc. (2000) in computer science from Ain Shams University, Egypt. Her work experience: Lecturer Assistant, Assistant professor, and Associate Professor, faculty, of Computer and Information Sciences, Ain. Shams University. Cairo, Egypt. Her research areas include Artificial Intelligence, Medical Data Mining, Machine Learning, Big Data and Robotic Simulation Systems. She can be contacted at email: abeer.mahmoud@cis.asu.edu.eg, abeer_f13@yahoo.com.