Deep High-order Asymmetric Supervised Hashing for Image Retrieval

Yongchao Yang*, Jianxin Zhang† *, Qian Wang*, Bin Liu‡
*Key Lab of Advanced Design and Intelligent Computing (Ministry of Education), Dalian University, Dalian, China
†School of Computer Science and Engineering, Dalian Minzu University, Dalian, China
‡International School of Information Science and Engineering (DUT-RUISE), Dalian University of Technology, Dalian, China
Email address:jxzhang0411@163.com, liubin_dlut@163.com

Abstract—**Deep hashing has recently been attracting more and more attentions for large-scale image retrieval task owing to its superior performance of search efficiency and less storage space requirements. Among deep hashing models, asymmetric deep hashing performs feature learning on query dataset and directly generates hash code on database images, significantly improving the retrieval performance of deep hashing models. Meanwhile, recently works also establish that high-order statistic of deep features are helpful to obtain more discriminant representations of images. Therefore, to boost the retrieval capability of deep hashing, this work tries to integrate merits of the high-order statistic module and the asymmetric deep hashing architecture, and it further proposes a novel deep high-order asymmetric supervised hashing (DHoASH) for image retrieval. More specifically, we utilize a powerful global covariance pooling module based on matrix power normalization to compute the second-order statistic features of input images, which is fluently embedded into an asymmetric hashing architecture in an end-to-end manner, leading to the generation of more discriminant binary hashing code. Experiment results on two benchmarks illuminates the effectiveness of the proposed DHoASH, which also achieves very competitive retrieval accuracy compared to the state-of-the-art methods.**

Keywords—**Deep supervised hashing, high-order statistics, covariance pooling, asymmetric, image retrieval**

I. Introduction

Hashing, representative Approximately Nearest Neighbor (ANN) [1] algorithm, has been attracting wide attentions in practical application for a long time due to its advantages of search efficiency and less storage space requirements. Current hashing methods can be divided into data-dependent and data-independent methods according to whether it depends on the data for generating hash code. Locality Sensitive Hashing (LSH) [2] is a typical data-independent method proposed in the early stage, which achieves binary hash code through a random hash function. Compared to data-independent hashing, data-dependent hashing commonly gains superior performance and has been addressed by more researchers in recent years. Besides, data-dependent hashing can be further classified into unsupervised hashing [3] [4] [5] and supervised hashing [6] [7] according to the label information used for the learning, in which supervised hashing generally encourage better retrieval performance than unsupervised hashing.

With the breakthrough progress of convolutional neural networks (CNNs) in various computer vision tasks, deep hashing, the integration of hashing and CNN, has also achieved great success for image retrieval application. Representations of deep hashing methods consist of convolutional neural network hashing (CNNH) [8], deep learning of binary hash codes (DLBHC) [9], deep supervised hashing with pairwise labels (DPSH) [10], deep supervised hashing with triplet labels (DTSH) [11], etc. To be specific, CNNH and DLBHC are two early works of pointwise deep hashing. CNNH [8] employs an adjustable coordinate descent algorithm to decompose the pairwise similarity matrix to generate approximate hash code, which automatically learns the hash function and feature representation through a deep neural network. On the contrary, DLBHC [9] adds a hidden layer to the last fully connected layer of the network structure, learning the hash function and feature representation of input image directly in an end-to-end manner. Different from pointwise deep hashing methods, DPSH [10], a representative pairwise deep hashing method, simultaneously learns hash function and feature expression based on pairwise label information of two input images. Besides, DTSH [11] simultaneously performs feature representations and hash code learning through supervised label information of image triples. All of the above methods learn the hash function for query images in the same way as the database images. More recently, Jiang and Li propose a novel asymmetric deep supervised hashing (ADSH) architecture, which learns two asymmetric hash functions in an asymmetric data stream, significantly improves the retrieval performance of deep hashing models [12]. In ADSH, the hash functions of query points are obtained through the adopted network, while the hash codes of dataset points are directly learned, achieving the new state-of-the-art performance. Following the stream, some researchers further put forward the deep asymmetric pairwise hashing (DAPH) [13] and dual asymmetric deep hashing (DADH) [14], which also achieves promising performance for image retrieval task.

Besides, recent works have established that global high-order statistics of deep features provide more discriminant representations of natural images, which has illustrated wide potentiality on a number of visual recognition tasks [23], [27], [29], [30]. Deep high-order models capture more complex relationships of deep convolutional features through second-order or higher-order functions inserted into convolutional

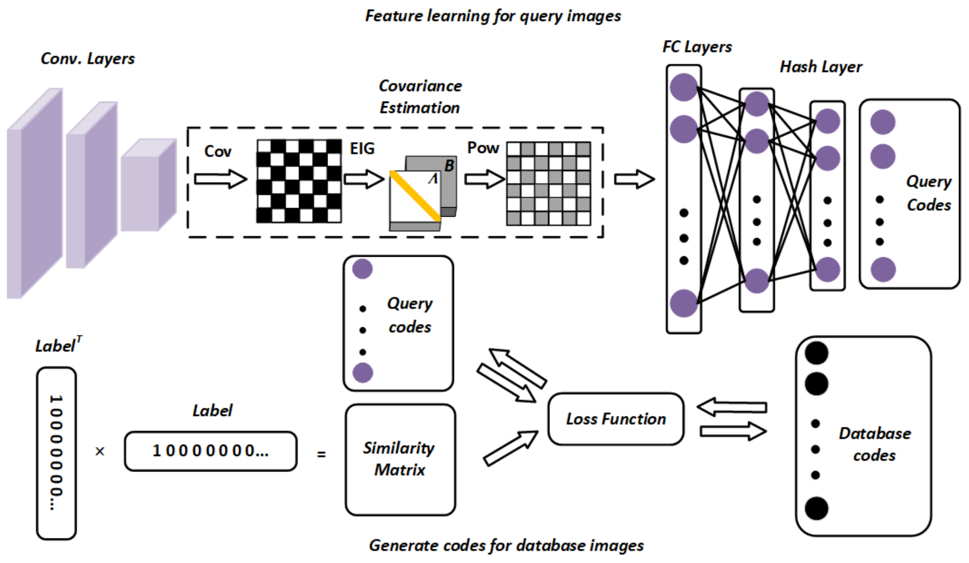

Fig. 1. The overall framework of the proposed DHoASH mainly consists of feature learning part and loss function part. The feature learning part performs extractions of both first-order and high-order features. We adopt the classic CNN-F network as the backbone architecture to extract first-order features, which will be regarded as the input of the covariance pooling layer to obtain high-order features of images by constructing covariance matrix and matrix power normalization. During this process, COV, EIG and Pow represent the covariance matrix, eigenvalue decomposition, and matrix power normalization, respectively. For the loss function part, the similarity matrix is generated by the label, provided by the given dataset. The loss function optimizes the parameters through approximate hash code of query images (query codes) and the supervised information (similarity matrix). Ultimately, data codes are generated relying on supervised information, query codes, and loss functions.

layers, leading to the classification performance improvement of CNNs. Representative deep high-order statistical models mainly consist of deep second-order pooling (DeepO2P), bilinear-CNN (B-CNN), matrix power normalized covariance (MPN-COV), etc. [23], [27], [29]. Among them, DeepO2P [27] integrates a second-order pooling layer into existing first-order convolutional network in an end-to-end learning manner. By contrast, B-CNN [29] computes second-order statistic features by applying an outer product of convolutional activations from two convolutional network streams, and experiment results illustrate its competitive performance compared to the current state-of-the-art on fine-grained visual recognition task. Then, Li et al. [23] embeds covariance square root normalization to capture deep second-order statistic features of input images and obtains excellent results on large-scale visual recognition tasks. To further improve the performance, GSoP-Net [30] inserts global covariance pooling in-between network to provide more distinguishable second-order feature representations. Therefore, this work tries to combine merits of the high-order statistics and the asymmetric deep hashing method to boost the retrieval performance of existing deep hashing methods. More specifically, it proposes a novel deep high-order asymmetric supervised hashing (DHoASH) for image retrieval, in which a powerful global covariance pooling module based on matrix power normalization (MPN) is embedded into an asymmetric hashing network to capture deep second-order covariance features, and this potentially leads to generating more discriminant binary hashing code. The overall framework of DHoASH can be demonstrated in Figure 1. The main contributions of our work are summarized as follows: (1) A novel deep high-order asymmetric supervised hashing (DHoASH) is presented for image retrieval application, which is the first attempt to combine high-order statistic model and asymmetric supervised hashing architecture. (2) DHoASH utilizes global covariance pooling based on matrix power normalization to capture the second-order statistic features of input images, which is naturally embedded into an asymmetric hashing network to generate more discriminant binary hash codes in an end-to-end learning manner. (3) Experiment results on two benchmarks illuminates the effectiveness of DHoASH, and it achieves very competitive retrieval accuracy compared to the state-of-the-art methods.

II. Method

A. Problem definition

Given m query images and n database images respectively denoted as $X = \{x_i\}_{i=1}^{m}$ and $Y = \{y_j\}_{j=1}^{n}$, the pairwise supervised information (i.e., the similarity matrix) is derived as $S \in \{-1, +1\}^{m \times n}$.

$S_{ij} = 1$ indicates that the two images belong to the same class and share common class labels. On the contrary, $S_{ij} = -1$ means that they belong to different classes and have different labels. Therefore, the supervised information matrix makes it convenient to judge whether the corresponding images are similar or not, which can guide the binary hash codes learning of query images and database images. These hash codes conserve the feature statistics of query images and database images. More specifically, for two similar images, the hamming distance of the corresponding hash codes should be relatively small while the hamming distance should be as far as possible if they are dissimilar. In this work, the learned binary code for query images is denoted as $U = \{u_i\}_{i=1}^{m} \in \{-1, +1\}^{m \times c}$ and $V = \{v_j\}_{j=1}^{n} \in \{-1, +1\}^{n \times c}$ is set to signify the directly learned binary hash codes of database images, where c represents the length of binary codes. Features of query images are mapped to a K-bit continuous representation in the process of approximate hash learning, which can be quantified to binary code through the function $sign(\cdot)$. Ultimately, the hash code of query images U can be obtained for image retrieval.

B. Network structure

As shown in Figure 1, DHoASH roughly includes two parts, the feature learning part and the loss function part. The principal work of DHoASH mainly focus on extracting high-order features of query images. Due to the pairwise similarity measurement module based parameter optimization strategy, our loss function should take the pairwise similarity information into account. The classic CNN-F [22] network is applied for first-order characteristic learning of images, which can be replaced with other deeper networks certainly, such as the ResNet. Through the operation of covariance matrix power normalization, we extract higher-order features from the input image, which carries sufficient global image features and grasps the similarity relationship between image pairs. The comprehensive configuration of the proposed network is listed in Table I. As aforementioned, we apply the designed loss function to ensure binary codes preserve the relationships among query images, database images, and similarity matrix.

TABLE I
CONFIGURATION OF NETWORK

Layers	Configuration
conv1	filter 64×11×11, stride 4×4, pad 0, LRN, pool 2×2
conv2	filter 256×5×5, stride 1×1, pad 2, LRN, pool 2×2
conv3	filter 256×3×3, stride 1×1, pad 1
conv4	filter 256×3×3, stride 1×1, pad 1, pool 2×2
conv5	filter 256×3×3, stride 1×1, pad 1
high-order	covariance_pooling
fc	32896
fc	4096
latent	K bit

Please note that DHoASH integrates the feature learning part and the loss function part into an end-to-end network framework, both of which can give feedback when an image pair passes through the network. It should be noted that the process of feature learning is only performed on the query images, while the hashing code of database images can be directly learned through the learned query image features and the existing similarity matrix S.

C. Feature learning

We employ the fundamental CNN-F [22] network as the extractor of the first-order features for the query images, keeping the top five conv. layers unmodified but abandoning the Pooling and Relu operations of the fifth layer. Features extracted from the first thirteen layers of the CNN-F network are regarded as the input of the covariance pooling layer, which can be expressed as Eq. (1).

$$G = \phi(x_i; \Theta) \tag{1}$$

where ϕ and Θ represent the process of first-order feature learning and the parameters of the backbone network, respectively. $G \in \mathbb{R}^{d \times n}$ is a matrix with n conv. features of d-dimension.

Embedding the covariance pooling layer behind the fifth convolutional layer, we can obtain a robust covariance matrix by extracting image features and performing matrix operations, such as eigenvalue decomposition (EIG), on the first-order representations, which models the holistic correlation of images. The covariance matrix can be established as follows:

$$O = G\bar{I}G^T \tag{2}$$

The $\bar{I}$ appeared in the equation can be calculated by $\bar{I} = \frac{1}{N}(I - AA^T)$, where I denotes an $n \times n$ identity matrix and A represents an n-dimensional vector, all elements of which are set to 1. T represents the transpose of the matrix.

The maximum likelihood estimation is adopted to assess the efficiency of the constructed covariance matrix, whose performance might be dissatisfying if the feature matrix develops into a small size but high dimensions. Generally speaking, the output features of the last layer of deep neural networks are higher dimensions with fewer feature vectors. As we all know, the eigenvalues of a non-full rank matrix can not match the dimensions, which means that some useful features extracted before might be missed, thus potentially limiting the capacity of the covariance matrix to represent features. It is worth noticing that the geometric structure established by the covariance matrix, a more detailed and plentiful feature representation, is a Riemannian manifold. Additionally, matrix power normalization has been proven to handle the above problems properly and evaluate the covariance matrix comprehensively for taking its manifold space into account. Therefore, after obtaining the robust covariance matrix through the matrix power normalization(MPN) operation on the sampled covariance matrix, the preliminary global features of the holistic image can be captured by utilizing the geometry of the Riemannian manifold, which is beyond the ability of the first-order feature representations [23].

Considering that the covariance matrix characterizes symmetric semi-definite properties, the eigenvalue decomposition operation can be performed on it, which can be written as follows:

$$O = B\Lambda B^T \tag{3}$$

where $\Lambda = diag(\lambda_1, ..., \lambda_d)$ represents the diagonal matrix corresponding to each eigenvalue, B is the orthogonal matrix composed of eigenvectors. Thereafter, the matrix power normalization of O can be obtained by performing a power operation on matrix Λ.

$$E \triangleq O^\alpha = B\Lambda^\alpha B^T \tag{4}$$

In Eq. (4), $\Lambda^{\alpha} = diag(\lambda_1^{\alpha}, ..., \lambda_d^{\alpha})$, where the exponent α refers to a positive real number less than 1. During this course, the brief but effectual MPN operation produces a robust covariance matrix and utilizes its Riemann geometric features, for which the robust covariance matrix E can model complex global relationships, outperforming the commonly used first-order feature statistics. Empirically, the performance of the covariance matrix E achieves the best when α equals 0.5.

From the above formula description, it can be observed that the forward propagation process of the covariance pooling layer is concise and explicit. In the process of back-propagation, following [23], [27], the derivative of the loss function with respect to O, G can be derived by the principle of matrix back-propagation, respectively. We employ the spectral norm (L_2 normalization) after the operation of MPN, which is denoted as $\sqrt{\lambda_{max}(EE^T)}$, enhancing the generalization ability of the extracted feature matrix. In order to convert the deep covariance representation into a compressed binary code, we add a hidden hash layer for the hash function learning, mapping the resulting feature matrix of the above operations to a fixed K-bit code. By doing so, binary-like code carrying global statistical characteristics can be obtained. We set $F(x_i; \Theta)$ as the final output of the network. Thereafter, the hash code of query images U can be learned through the threshold function $sign(\cdot)$ and written as below:

$$U = sign(F(x_i; \Theta)) \tag{5}$$

It is worth mentioning that the entire network is optimized in an end-to-end pattern. During the test process, to retain and integrate the inherited global attributes of coding features naturally, an out-of-sample extension need to be conducted to generate the K-dimensional binary vectors and detect the capability of our network, which will be elaborated in the next section.

D. Loss optimization and learning

In order to generate the hash code that stores pairwise similarity, the L_2 loss between the similarity matrix (supervised information) and the inner product matrix of query-database image pairs should be required as small as possible, which is defined as follows:

$$\begin{aligned} \min_{U,V} = L(U,V) = \sum_{i=1}^{m}\sum_{j=1}^{n}(u_i^T v_j - cS_{ij})^2 \\ s.t.\ U \in \{-1,+1\}^{m\times c}, V \in \{-1,+1\}^{n\times c}. \end{aligned} \tag{6}$$

Here, $u_i = sign(F(x_i; \Theta))$, where $F(x_i; \Theta) \in \mathbb{R}^c$. During back-propagation, we use $tanh(\cdot)$ to approximate the $sign(\cdot)$ for avoiding the problem of vanishing gradients.

In practical applications, our obtained images generally cover the whole database $Y = \{y_j\}_{j=1}^n$, instead of the query set that have been partitioned. In this case, we can randomly sample m data images from the database images as the query set. More precisely, we set $X = Y^{\Omega}$, where Y^{Ω} indicates the database images indexed by Ω. We use $\Gamma = \{1, 2, ..., n\}$ and $\Omega = \{i_1, i_2, ..., i_m\} \in \Gamma$ to indicate the indices of the database images and the sampled query images, respectively. Thereafter, we set $S = S^{\Omega}$ to represent the similarity information, where $S^{\Omega} \in \{-1,+1\}^{m\times n}$ indicates sub-matrix of S rows indexed by Ω. Please note the affiliation of $\Omega \in \Gamma$. Consequently, y_i can be expressed from two perspective: One is the binary hash code of database v_i, and the other is the query expression $tanh(F(y_i; \Theta))$. We add an extra consistency constraint to make v_i and $tanh(F(y_i; \Theta))$ as close as possible, and regard hyper-parameter γ as the coefficient. The final loss function can be written as below:

$$\begin{aligned} \min_{\Theta,V} = L(\Theta,V) = \sum_{i\in\Omega}\sum_{j\in\Gamma}[tanh(F(y_i;\Theta))^T v_j - cS_{ij}]^2 \\ +\gamma\sum_{i\in\Omega}[v_i - tanh(F(y_i;\Theta))]^2 \\ s.t. V \in \{-1,+1\}^{n\times c} \end{aligned} \tag{7}$$

Next, in order to directly represent the process of network back-propagation, we calculate the partial derivatives from the hash layer to the covariance pooling layer of the original first-order network. For the purpose of convenience, Hadamard product can be represented by $\odot$ and $\widetilde{u_i} = tanh(F(y_i; \Theta))$. According to the loss function formula and the back-propagation algorithm, the derivative can be calculated:

$$\begin{aligned} \frac{\partial L}{\partial H} = \{2\sum_{j=1}^{n}[(\widetilde{u_i}^T v_j - cS_{ij})v_j] + 2\gamma(\widetilde{u_i} - v_j) \\ \odot\ (1 - \widetilde{u_i}^2)\} \end{aligned} \tag{8}$$

Given the partial derivatives propagated from the hash layer, we can easily obtain the partial derivatives related to the eigenvalue decomposition through the back-propagation chain rule $\frac{\partial L}{\partial H} \rightarrow \frac{\partial L}{\partial O} \rightarrow \frac{\partial L}{\partial G}$, this calculation process can be expressed in matrix form:

$$\frac{\partial L}{\partial O} = B((Q^T \odot (B^T\frac{\partial L}{\partial B})) + (\frac{\partial L}{\partial \Lambda})_{diag})B^T \tag{9}$$

where the symbol $diag$ represents matrix diagonalization and $Q = Q_{ij}$ denotes a square matrix when $Q_{ij} = 1/(\lambda_i - \lambda_j)\ if\ i \neq j$ and $Q_{ij} = 0$ otherwise. Readers can refer to [23], [27] for details.

Ultimately, given $\frac{\partial L}{\partial O}$, the partial derivatives of loss function and feature matrix G can be written as below:

$$\frac{\partial L}{\partial G} = \overline{I}G(\frac{\partial L}{\partial O} + (\frac{\partial L}{\partial O})^T) \tag{10}$$

For the entire network, the supervised information will be used as the optimization specification to learn the hash function, re-updating and optimizing the parameters of the network gradually to obtain better results.

Since our proposed network has the property of asymmetric structure, the details of directly learning database image hash codes will be introduced next. By extracting the query image features, the parameters in the network are optimized. After training, all parameters of the network are fixed. Then, we update V column by column, keeping the other columns fixed when any column of V is being updated. The procedure is defined as follows:

$$\min_{V} L(V) = ||\widetilde{U} V^T||_F^2 + tr(VZ^T) + const \tag{11}$$

where $Z = -2cS^T\widetilde{U} - 2\gamma\overline{U}$, and $const$ represents a constant value unrelated to V. Here, $\overline{U} = \{\overline{u}_j\}_{j=1}^n$ and $\overline{u}_j = \begin{cases}\overline{u}_j & if\ j \in \Omega \\ 0 & otherwise\end{cases}$. Naturally, expand Eq. (11):

$$\begin{gathered}\min_{V_{*p}} L(V_{*p}) = tr(V_{*p}[2\widetilde{U}_{*p}^T\widehat{U}_p\widehat{V}_p^T + Z_{*p}^T]) + const \\ s.t.\ V_{*p} \in \{-1, +1\}^n\end{gathered} \tag{12}$$

V_{*p} indicates the pth column of V and $\widehat{V}_p$ denotes the matrix of V precluding V_{*p}. $\widetilde{U}_{*p}$ expresses the pth column of $\widetilde{U}$ and $\widehat{U}_p$ indicates the matrix of $\widetilde{U}$ precluding $\widetilde{U}_{*p}$. Similarly, Z_{*p} denotes the pth column of Z. Finally, hash codes of the database image can be learned through Eq. (13):

$$V_{*p} = -sign(2\widehat{V}_p\widehat{U}_p^T\widetilde{U}_{*p} + Z_{*p}) \tag{13}$$

Ultimately, hash code of the database images will be learned. Readers can read [12] for more details on asymmetric strategies.

III. Experiment

We conduct experiments to evaluate the effectiveness of the proposed DHoASH. All experiments are performed on a server with 3.30GHz CPU, 64GB RAM, and NVIDIA GTX 1080GPU using MatConvNet [21] toolkit in MATLAB R2014a.

A. Datasets and settings

Two benchmarks, i.e., CIFAR-10 [19] and NUS-WIDE [20], are used to verify the performance of our DHoASH. CIFAR-10 dataset is a single-label dataset containing 60,000 RGB images with a resolution of 32×32 in ten categories in total. If the labels of the selected two images are consistent, we consider the two images belong to the same category and completely similar. NUS-WIDE dataset contains 269,648 images collected from Flickr which contains 5018 unique tags. It is a multi-label dataset where each picture individually uses one or more label concepts from the given 81 concepts. Following [12], we select 195,834 images associated with the 21 most widely used concepts in our experiments, each of which contains at least 5,000 RGB images.

We adopt the mean average precision (MAP) of different bits $\{12, 24, 32, 48\}$ to evaluate the quality of the retrieved data images. Considering that our loss function is based on asymmetric pairwise similarity relations, the pairwise similarity matrix S is established based on the category labels. In CIFAR-10, two images with a semantic label are shared by the semantically similar pair relationship $S_{ij} = 1$. As for multi-label dataset NUS-WIDE, we define similar pairs as sharing at least one semantic label. In addition, we employ different learning rate strategies for training on different datasets. To be specific, the learning rate we used on the CIFAR-10 dataset is set to $[10^{-4}, 10^{-6}]$. As for the NUS-WIDE, the learning rate is $[10^{-4.5}, 10^{-6}]$. Besides, we set the same hyper-parameter γ value as 200 for both benchmarks.

B. Experiment results

The compared experiment results on CIFAR-10 are reported in Table II. We compare our DHoASH with some representative deep supervised hash methods including deep supervised hashing (DSH) [15], deep hashing network(DHN) [16], deep quantization network(DQN) [17], deep supervised hashing with pairwise labels(DPSH) [10], deep supervised discrete hashing(DSDH) [18], deep forest hashing(DFH) [24], deep policy hashing network(DPHN) [25], and deep saliency hashing(DSaH) [26]. Among these methods, DSH, DHN, DPSH, DSaH, DFH and DPHN belong to symmetric hashing models while the remaining three methods are based on asymmetric hashing architectures.

TABLE II
Compared MAP results on CIFAR-10

Method	**Year**	**12 bits**	**24 bits**	**32 bits**	**48 bits**
DSH [15]	2016	0.6441	0.7421	0.7703	0.7792
DHN [16]	2016	0.6805	0.7213	0.7233	0.7332
DPSH [10]	2016	0.6818	0.7204	0.7341	0.7464
DSaH [26]	2018	0.8003	0.8457	0.8476	0.8478
DFH [24]	2019	0.4570	0.5130	0.5240	0.5590
DPHN [25]	2019	0.8440	0.8620	0.8680	0.8780
DAPH [13]	2017	0.7569	0.8213	0.8307	0.8448
ADMH [28]	2019	0.9226	0.9379	**0.9477**	0.9433
ADSH [12]	2018	0.8898	0.9280	0.9310	0.9390
DHoASH	–	**0.9293**	**0.9394**	0.9409	**0.9472**

As illustrated in Table II, DPHN is the optimal one among the six symmetric hashing methods. However, our DHoASH still outperforms it by a large margin. DHoASH obtains 8.53%, 7.74%, 7.29% and 6.92% gains over DPHN on 12, 24, 32 and 48-bit results, respectively. Compared with asymmetric hashing models, DHoASH achieves the best retrieval performance on three cases (12, 24 and 48 bits), only inferior to ADMH on 32 bits. Additionally, DHoASH gains 3.95%, 1.14%, 0.99%, 0.82% performance improvement over its first-order baseline of ADSH. These compared experiment results powerfully illuminate the effectiveness of the proposed DHoASH, which mainly are contributed to the robust high-order statistic features employed in deep hashing.

The compared MAP results with four various bits on NUS-WIDE dataset are listed in Table III. Here, we also compare DHoASH with seven symmetric hashing models [10], [16]–[18], [24]–[26] and three asymmetric hashing networks [12], [13], [28], which are regarded as the representative state-of-the-art deep hashing methods proposed in recent four years. As illuminated in Table III, our DHoASH achieves the optimal retrieval results on all of the four cases. It significantly outperforms DSaH by 1.16%, 3.84%, 3.27% and 3.56% in cases of 12, 24, 32 and 48 bits, respectively. Note that DSaH gains the

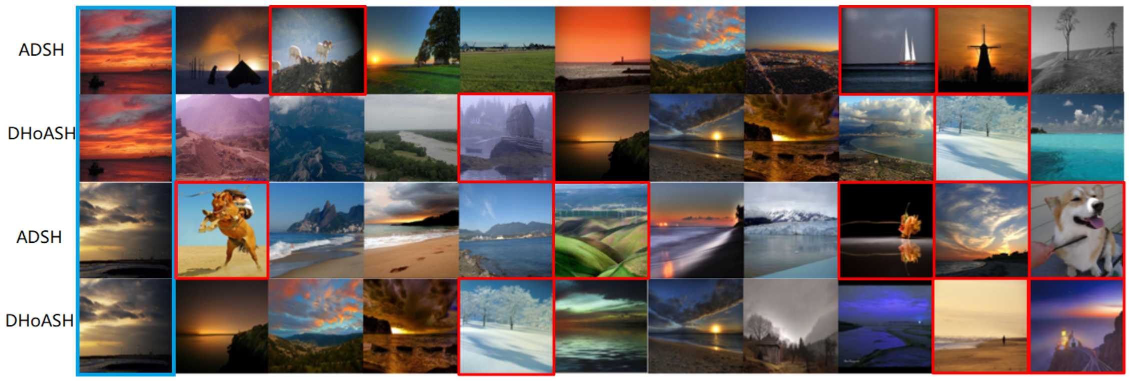

Fig. 2. Examples of visualization express the top-10 images retrieved from the the NUS-WIDE dataset by different methods of 24-bit. The first column for each sub-figure enclosed in a blue bounding box indicates the query samples and the following ten columns represent the top-10 returned samples. Note that the red box marks denote the "negative samples".

best performance on this dataset among the seven symmetric models, and slightly outperforms DPHN. Since NUS-WIDE has more categories and multiple semantic labels for each image, it is more challenging to improve retrieval accuracy on this benchmark. However, compared with ADSH model, DHoASH still outperforms it by 0.96%, 1.40%, 0.16%, and 0.31% on 12, 24, 32 and 48 code bits. These compared results once again demonstrate the well performance of DHoASH for image retreival.

TABLE III
COMPARED MAP RESULTS ON NUS-WIDE

Method	Year	12 bits	24 bits	32 bits	48 bits
DHN [16]	2016	0.7719	0.8013	0.8051	0.8146
DQN [17]	2016	0.7680	0.7760	0.7830	0.7920
DPSH [10]	2016	0.7941	0.8249	0.8351	0.8442
DSDH [18]	2017	0.7760	0.8080	0.8200	0.8290
DSaH [26]	2018	0.8380	0.8540	0.8640	0.8730
DFH [24]	2019	0.6220	0.6590	0.6740	0.6950
DPHN [25]	2019	0.8340	0.8490	0.8500	0.8540
DAPH [13]	2017	0.7167	0.7706	0.7870	0.8164
ADMH [28]	2019	0.8405	0.8835	0.8851	0.9069
ADSH [12]	2018	0.8400	0.8784	0.8951	0.9055
DHoASH	–	**0.8496**	**0.8924**	**0.8967**	**0.9086**

Taking the more challenging NUS-WIDE dataset as an instance, we further conduct the retrieval performance comparison of 24-bit hash code based on CNN-F [22] network. Randomly selecting two images as query samples, we can obtain an image sequence sorted by the hamming distance between the query image and database image, and then exhibit the first 10 retrieved images. Specifically, for each given query sample, we calculate the hamming distance based on the hash code, which is determined by a deep hashing paradigm combined with covariance estimation. Negative samples suggest the query image and retrieved images have no shared label, which can be confirmed based on the ground-truth label of the returned 10 samples. In particular, for the multi-label attributes of NUS-WIDE, the retrieval accuracy refers to whether the retrieved sample and the query sample share at least one semantic label. As shown in Figure 2, the first column in the blue borders represents the query samples, and the next 10 columns represent the search results corresponding to each query, in which the "negative samples" are encapsulated in red borders. Here, we present the comparison result of our method DHoASH and ADSH for the NUS-WIDE dataset, through which the retrieval performance can be evaluated roughly based on the top-10 ground-truth neighbors and the number of red boxes. Obviously, the number of "negative samples" contained in the red border of our DHoASH is less than that of ADSH, demonstrating the better performance achieved by DHoASH and the effectiveness of the covariance merged into deep hashing. Compared with ADSH, our proposed DHoASH achieves superior hash code representations in the mode of embedding robust covariance estimation, so that the retrieved images are more similar to the semantics of query images.

IV. CONCLUSION

In this work, we propose a novel deep high-order asymmetric supervised hashing (DHoASH) for image retrieval application. DHoASH embeds a robust covariance pooling layer into asymmetric hashing network in an end-to-end manner, providing a new perspective for learning asymmetric deep hash code via exploiting high-order statistic feature interactions. Experiment results on two benchmarks illuminates the effectiveness of DHoASH. Our future work is to recover more powerful high-order global statistics descriptor for deep hashing models. We will also vigorously explore the influence of time complexity on deep high-order hashing, and strive to improve the retrieval accuracy with less time complexity.

ACKNOWLEDGMENT

This work was partially supported by the National Key R&D Program of China (2018YFC0910506), the National Natural Science Foundation of China (61972062 and 61603066), the Natural Science Foundation of Liaoning Province (2019-MS-011), the Key R&D Program of Liaoning Province (2019 JH2/10100030) and the Liaoning BaiQianWan Talents Program.

References

[1] A. Andoni and I. Razenshteyn, "Optimal data-dependent hashing for approximate near neighbors," in *Proceedings of the forty-seventh annual ACM symposium on Theory of computing (STOC-15)*, ACM, 2015, pp. 793-801.

[2] B. Kulis and K. Grauman, "Kernelized locality-sensitive hashing for scalable image search," in *Proceedings of the 12th IEEE International Conference on Computer Vision (ICCV-09)*, IEEE, 2009, pp. 2130—2137.

[3] Y. Weiss and A. Torralba and R. Fergus, "Spectral hashing," in *Proceedings of the 21st International Conference on Neural Information Processing Systems (NIPS-08)*, 2008, pp. 1753-1760.

[4] W. Liu, J. Wang, S. Kumar, and S. F. Chang, "Hashing with graphs," in *Proceedings of the 28th international conference on machine learning (ICML-11)*. Citeseer, 2011, pp. 1-8.

[5] Y. C. Gong, and 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 (TPAMI)*, vol.35, no.12, pp. 2916-2929, 2012.

[6] W. Liu, J. Wang, R. R. Ji, Y. G. Jiang and S. F. Chang, "Supervised hashing with kernels," in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR-12)*, IEEE, 2012, pp. 2074-2081.

[7] X. S. Shi, F. Y. Xing, K. D. Xu, and M. Sapkota, and L. Yang, "Asymmetric discrete graph hashing," in *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)*, AAAI, 2017, pp. 2541-2547.

[8] R. K. Xia, Y. Pan, H. J. Lai, C. Liu, and S. C. Yan, "Supervised hashing for image retrieval via image representation learning," in *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence (AAAI-14)*, AAAI, 2014, pp.2156-2162.

[9] 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-2015)*, IEEE, 2015, pp. 27-35.

[10] W. J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI-16)*, AAAI, 2016, pp. 1711-1717.

[11] X. F. Wang, Y. Shi, and K. M. Kitani, "Deep supervised hashing with triplet labels," in *Asian conference on computer vision (ACCV-16)*, 2016, pp. 70-84.

[12] Q. Y. Jiang, and W. J. Li, "Asymmetric deep supervised hashing," in *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018.

[13] F. M. Shen, X. Gao, L. Liu, Y. Yang, and H.T. Shen, "Deep asymmetric pairwise hashing," in *Proceedings of the 25th ACM international conference on Multimedia (ACM MM-17)*, ACM, 2017, pp. 1522-1530.

[14] J. X. Li, B. Zhang, G. M. Lu, and D. Zhang, "Dual Asymmetric Deep Hashing Learning," *IEEE Access*, vol.7, pp. 113372-113384. 2019.

[15] H. M. Liu, R. P. Wang, S. G. Shan, and X. L. Chen, "Deep Supervised Hashing for Fast Image Retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-16)*, IEEE, 2016, pp. 2064-2072.

[16] H. Zhu, M. S. Long, J. M. Wang, and Y. Cao, "Deep Hashing Network for efficient similarity retrieval," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, AAAI, 2016, pp. 2415-2421.

[17] Y. Cao, M. S. Long, J. M. Wang, H. Zhu, and Q. F. Wen, "Deep Quantization Network for efficient image retrieval," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*, AAAI, 2016, pp. 3457-3463.

[18] Q. Li, Z. N. Sun, R. He, and T.N. Tan, "Deep supervised discrete hashing," in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS-17)*, 2017, pp. 2479-2488.

[19] A. Krizhevsky, "Learning Multiple Layers of Features from Tiny Images," *Masterś thesis, University of Tront*, Citeseer 2009.

[20] T. S. Chua, J. H. Tang, R. C. Hong, H.J. Li, Z. P. Luo, and Y.T. Zheng, "NUS-WIDE: a real-world web image database from National University of Singapore," in *Proceedings of the ACM international conference on image and video retrieval (ACM CIVR-09)*, ACM, 2009, pp. 1-9.

[21] A. Vedaldi,and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proceedings of the 23rd ACM international conference on Multimedia (ACM MM-15)*, ACM, 2015, pp. 689-692.

[22] K. Chatfield, and K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," *arXiv preprint arXiv:1405.3531* (2014).

[23] P. H. Li, J. T. Xie, Q. L. Wang, and W. M. Zuo, "Is Second-Order Information Helpful for Large-Scale Visual Recognition?" in *2017 IEEE International Conference on Computer Vision (ICCV-17)*, IEEE, 2017, pp. 2089-2097.

[24] M. Zhou, X. H. Zeng, and A. Z. Chen, "Deep Forest Hashing for Image Retrieval," *Pattern Recognition*, 2019.

[25] S. Y. Wang, H. J. Lai, and Y. F. Yang, and J. Yin, "Deep Policy Hashing Network with Listwise Supervision," in *Proceedings of the 2019 on International Conference on Multimedia Retrieval (ICMR-19)*, ACM, 2019, pp. 123-131.

[26] S. Jin, H. X. Yao, and X. S. Sun, S. C. Zhou, L. Zhang, and X. S. Hua, "Deep saliency hashing," *arXiv preprint arXiv:1807.01459* (2018).

[27] C. Ionescu, O. Vantzos, and C. Sminchisescu, "Matrix Backpropagation for Deep Networks with Structured Layers," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV-15)*, IEEE, 2015, pp. 2965-2973.

[28] L. Ma, H. L. Li, F. M. Meng, Q. B. Wu, and K.N. Ngan, "Discriminative deep metric learning for asymmetric discrete hashing," *Neurocomputing*.

[29] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear cnn models for fine-grained visual recognition," in *Proceedings of the IEEE international conference on computer vision (ICCV-2015)*, IEEE, 2015, pp. 1449-1457.

[30] Z. L. Gao, J. T. Xie, Q. L. Wang, and P. H. Li, "Global second-order pooling convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR-19)*, IEEE, 2019, pp. 3024-3033.