Front. Comput. Sci., 2017, 11(2): 253–265
DOI 10.1007/s11704-017-6537-3

RESEARCH ARTICLE

Hierarchical deep hashing for image retrieval

Ge SONG[1,2], **Xiaoyang TAN** (✉)[1,2]

1 College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China

2 Collaborative Innovation Center of Novel Software Technology and Industrialization, Nanjing 211106, China

Abstract We present a new method to generate efficient multi-level hashing codes for image retrieval based on the deep siamese convolutional neural network (DSCNN). Conventional deep hashing methods trade off the capability of capturing highly complex and nonlinear semantic information of images against very compact hash codes, usually leading to high retrieval efficiency but with deteriorated accuracy. We alleviate the restrictive compactness requirement of hash codes by extending them to a two-level hierarchical coding scheme, in which the first level aims to capture the high-level semantic information extracted by the deep network using a rich encoding strategy, while the subsequent level squeezes them to more global and compact codes. At running time, we adopt an attention-based mechanism to select some of its most essential bits specific to each query image for retrieval instead of using the full hash codes of the first level. The attention-based mechanism is based on the guides of hash codes generated by the second level, taking advantage of both local and global properties of deep features. Experimental results on various popular datasets demonstrate the advantages of the proposed method compared to several state-of-the-art methods.

Keywords image retrieval, deep hashing, hierarchical deep hashing

1 Introduction

With the popularization of high-pixel camera phones and the explosive growth of the Internet, massive images flood our daily lives, which makes it more difficult to find relevant images that we are interested. Image retrieval, i.e., finding images containing the same object or scene as in a query image, has attracted much attention from both academia and industries in recent years. For this task, the semantic gap between low-level content and higher-level concepts remains the major challenge for all current approaches that rely on visual similarity for judging semantic similarity [1]. Traditional methods extracting visual content from images are mainly based on statistical information at the pixel level, e.g., scale-invariant feature transform (SIFT) and histogram of oriented gradient (HOG), which could not capture effective semantic-level representation and subject to various image quality deterioration such as blurring, low precision, various lighting conditions and so on.

Recently, many studies have shown that convolutional neural networks (CNNs) could learn representations with high-level semantic concepts and reported that it achieved the state of the art performance in many computer vision tasks such as image classification [2], object detection [3] or semantic segmentation [4]. Notably, in image retrieval field, Zheng et al. [5] discussed the progress of image retrieval over the past decade in detail and highlighted the popularity of CNN-based models in the community. Many works [6, 7] adopted solutions based on features extracted from fully-connected layers of a CNN pre-trained for the image classification task. While several works [8–11] paid more attention on features from the deep convolutional layers of CNNs and demonstrated that these features contain particular interpretations of local image regions which usually lead to better performance. The

Received November 13, 2016; accepted January 24, 2017

E-mail: sunge@nuaa.edu.cn; x.tan@nuaa.edu.cn

work of Ref. [12] integrated CNN features with the Bag-of-Words (BoW) model as contextual evidence to promote the match accuracy of keypoints, based on the observation that the automatically learnt deep feature sets are complementary to the hand-crafted SIFT feature sets. However, it remains some problems with adopting deep feature directly for retrieval or other visual tasks. The units of CNN are activated by different regions or objects of image [13], which indicates that the learn features include a variety of semantic concepts. Therefore, it is reasonable to localize the discriminative image regions and use particular localizable deep representations for retrieval task. On the other hand, the CNN features are faced with the problem of high computational cost in similarity calculation due to the high-dimensional representation produced, hence being not appropriate for rapid image retrieval. A practical solution to this issue is to use the technique of hashing based methods [14–17]. Actually, it has become one of the most popular approaches in large-scale vision problems [18].

The idea of hash methods is to design a nonlinear function which projects images into compact binary codes so that similar images are mapped into similar codes. Therefore, with millions of images being represented by compact binary codes, the storage cost can be greatly reduced, and the retrieval problem can be done very efficiently by explicitly computing Hamming distance among query and database images. Inspired by the success of deep learning in various visual tasks, a few deep hash models have been proposed [16,19–21] recently. The semantic space of image dataset can be approximated reasonably well by the learned Hamming space of hash codes, partly due to the impressive representation power of CNNs on which the hash codes are actually built.

Although these methods achieved improved performance, they have some limitations. Firstly, all these methods paid less attention on the modeling of pairwise semantic similarity between images, hence ignoring the learning of factors that explain the relevance between the pair of images that are related to each other but not necessarily in the same category. Secondly, these methods concentrate on hashing full-connect layer to obtain highly compact global-level binary codes. This may lead to two unwelcome consequences: 1) too compactness essentially conflicts with the expressiveness of the codes, and 2) the local information of images that is important for image retrieval [6] could be lost.

To deal with the aforementioned issues, we made several contributions as follows. Firstly, we proposed a deep siamese CNN (DSCNN) that learns semantic-preserved hash codes in both point-wise and pair-wise manners. Secondly, we enrich the expressiveness of hash codes by extending them to a two-level hierarchical coding scheme, in which the first level captures the local semantic information, while the subsequent level squeezes them to more global and compact codes. The outputs of both levels are combined in a hierarchical way to improve the retrieval efficiency. Last but not least, to improve the retrieval robustness and runtime efficiency, instead of using the full hash codes of the first level, we propose an attention-based mechanism that relies on activation maps and class activation map (CAM) [13] to select some of its most essential bits specific to each query image, hence taking advantage of both local and global properties of deep features. Experimental results on several popular benchmark data sets demonstrate the superiority of the proposed method over other methods. Figure 1 illustrates some retrieval results using the proposed method.

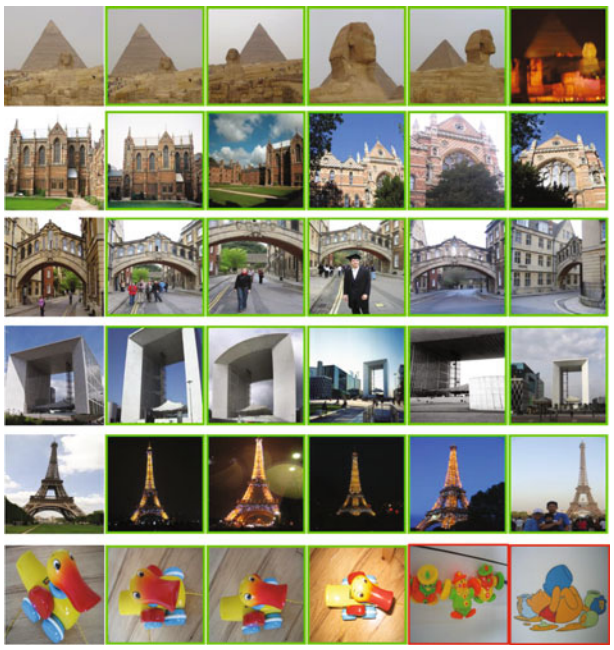
Fig. 1 Examples of the retrieval results by our proposed hierarchical hashing method for effective query images (left, without border) on four different datasets: Holidays, Oxford Buildings, Paris Buildings and UKB

In what follows, Section 2 discusses the related work to our method, the proposed method is detailed in Section 3 and its effectiveness is evaluated in Section 4. Finally, we conclude the paper in Section 5.

2 Related work

• **Deep descriptors for image retrieval** With the progress in supervised learning representation, some works [6, 8, 22] have considered the use of deep features for image retrieval. Babenko et al. [6] have demonstrated the suitability of features from fully-connected layers for image retrieval with or without fine-tuning on related datasets. They reported that the best performance is observed not on the very top of the network, but rather at the layer that is two levels below the outputs, i.e., convolutional descriptors. Zheng et al. [11] investigated the transfer learning capability of CNN features, and they found that combining pooled features from multiple convolution layers is even more useful. Kalantidis et al. [23] applied non-parametric spatial and channel-wise weighting strategies to the convolutional layers and achieved significant improvement. Ng et al. [10] adopted the VLAD method to encode activations of the convolutional layer. These works mainly focus on aggregating local descriptors into a global representation while ignoring the local semantic property.

In order to obtain deep features for local regions, Paulin et al. [22] introduced a feature representation for patches based on the kernel feature map of a convolutional kernel. The method is similar to SIFT, adopting Hessian-Affine method to detect interesting points and then extracting patch-level descriptors. However, traditional interest point detectors could not ensure that the detected points fall in discriminative objects or scene areas. Therefore, detecting object regions at a higher level becomes more important. Girshick et al. [3] presented R-CNN method which uses fully connected layers feature to train a bounding box regressor and classifier. Ren et al. introduced a region proposal network (RPN) to generate object proposals with R-CNN. Salvador et al. [24] took advantage of this end-to-end object detection architecture and proposed to extract both image and region features for instance search. Unlike previous methods, Zhou et al. [13] introduced the class activation map (CAM) model to localize the object in an unsupervised manner.

• **Deep learning for hashing** Deep learning methods have been popularly used for hashing recently. Xia et al. [19] learned hash functions using CNNs to fit the learned hash codes. Liong et al. [15] developed a multiple layer full-connect network to learn hash functions by maximizing the inter-class variations and minimizing the intra-class variation.

In those methods, the learning process of image representations is separated from learning hash functions. To deal with this issue, several deep architectures have been proposed to combine these two tasks within a single learning framework [16, 20, 25]. Lin et al. [25] proposed an approach to simultaneously learn domain image representations and hashing-like functions for image retrieval, through embedding a latent sigmoid full-connect layer at the top of AlexNet network.

However, at the training stage of these methods, only the label information is used while the similarity information between samples is completely ignored. Without similarity constraints, dissimilar images may be encoded to similar binary codes, which should be avoided in image retrieval. To address this issue, Lai et al. [26] designed a network-in-network deep model NINH to learn image representation and hash codes in one stage with a triple rank loss function. Zhao et al. [16] used ranking loss to train their model and their hash layer is connected with both two fully-connect layers to utilize diverse feature information. Liu et al. [21] proposed to take pairs of images as training inputs to learn similarity-preserving binary codes. One challenge of these methods, however, is that it is usually difficult to decide how many pairs or triplets to be selected for training. Another problem is that the hash codes based on the fully-connect layer help to capture the global-level information, but it is not easy to embed local semantic information in them.

3 The proposed method

In this section, we present DSCNN to produce global hash code and local hash code for image retrieval task.

3.1 Learning multi-level semantic-preserved hash code

We propose a two-level hierarchical coding scheme to hash convolutional features as local-level codes and hash fully-connect features as global-level codes. Besides, we use a siamese architecture to explore the point-wise (label) and pair-wise (relevance between two images) semantic information in the training set. The resulted DSCNN network is illustrated in Fig. 2, where the activation of the fully-connect hash layer and the conv7 layer are both tanh function so as to output better approximate hash code.

Assuming that the feature maps of Conv7 are $I_1, I_2, \ldots, I_C \in (-1, 1)^{W \times P}$, W, P are width and height, C is the number of filters, the output of Hash Layer are $a \in (-1, 1)^H$, H is the length of hash code. $\hat{y}$ is the output of softmax layer, y is expected output. To learn the two levels of hash codes (i.e., the first level hash codes H_l and the second level H_g, c.f., Fig. 2), we adopt two different loss functions respectively. Particularly, the loss function of the first local-level hash codes H_l is defined as the point-wise loss:

$$L_1 = -\sum_{j=1}^{N} y_i \log(\hat{y}_j), \tag{1}$$

and that of the second global-level hash codes H_g is defined

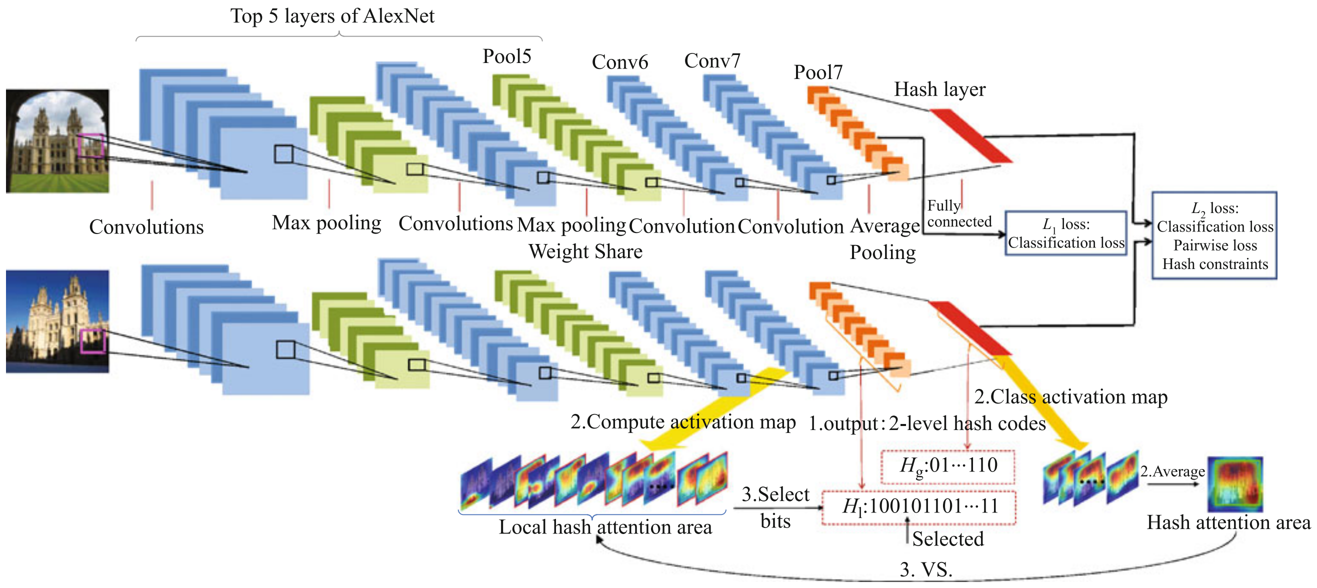

Fig. 2 The architecture of the proposed deep siamese CNN (DSCNN) for image retrieval (Firstly, the semantic-preserved global-level (H_g) and local-level hash codes (H_l) are learned in a joint point-wise and pair-wise manner. Secondly, we obtain activation maps (AM) of Covnv7 and class activation maps (CAM) of each bit of H_g, and average CAMs to acquire "Hash attention area", meanwhile we get "Local hash attention area" by AMs, because one AM corresponds to one bit of H$_l$, visually highlight bits (red colored) can be selected as compact local hash code through comparing these areas. Finally, both H_g and selected H$_l$ bits are combined to give the hash codes for a given query)

as the following pair-wise loss:

$$\begin{aligned}L_2 &= -L_1 + \alpha J_{11} + \alpha J_{12} + \beta J_2 + \gamma J_3\\ &= -\sum_{j=1}^{N} y_i \log(\hat{y}_j) + \alpha \sum_{j=1}^{N}\sum_{i=1}^{N} \delta(y_j = y_i)\|a_j - a_i\|_2^2\\ &\quad +\alpha \sum_{j=1}^{N}\sum_{i=1}^{N} \delta(y_j \neq y_i)\max(0, c - \|a_j - a_i\|_2^2)\\ &\quad +\beta \sum_{j=1}^{N}(\||a_j| - 1\|^2) + \gamma \sum_{j=1}^{N}(\|avg(a_j) - 0\|^2), \end{aligned} \tag{2}$$

where δ is an indicator function, avg is mean function, and c is a constant. The first L_1 and second item J_{1*} aim to embed semantic consistency and similarity to hash code respectively. The third term J_2 aims to minimize the quantization loss between the learned binary code and the original approximate code. The last term J_3 enforces evenly distribution of -1 and 1 in hash code. α, β, γ are parameters to balance the effect of different terms.

After training, we could obtain the hash code by quantization. That is, global-level hash codes H_g and local-level hash codes H_l are obtained respectively by

$$H_g = \delta(a > 0), H_l = \delta(f > 0), \tag{3}$$

where $f \in (-1, 1)^C, f_k = \frac{1}{W\times P}\sum_{i=1}^{W}\sum_{j=1}^{P} I_k(i, j), k = 1, 2, \ldots, C$.

3.2 Attention-based hash bits section

A number of works [27, 28] have attempted to visualize the internal representations and have shown that the deep convolutional feature maps are activated with different regions. Therefore, we can identify the image regions most relevant to the particular object by weighted summing these feature maps as done in the method of Class Activation Map (CAM) [13]. However, the CAM is originally used to obtain discriminative image region of identified category and is not suitable for image retrieval. In this work, we regard bits as categories and use CAM to locate the salient region for each global hash bits. For local-level codes, we adopt a similar method to get CAMs. Since some of feature maps are irrelevant to those salient areas of interest (c.f., Fig. 3), hash bits corresponding to them could be discarded as they do not contain useful information for image retrieval.

The first stage is to find out the attention region in the input image for each bit of the global hash code H_g. We treat every bit as a category and compute their CAMs as $M_1, M_2, \ldots, M_H$, and average them to obtain M_{avg}. The average map is then binarized as $B_{avg} = \delta(M_{avg} > \theta)$, where θ is a threshold. Finally, we obtain the attention region by finding the largest connected subgraph of B_{avg}. Figure 3 (middle column) gives some examples of the resulting average binarized map.

The next stage is to evaluate the salience score for each lo-

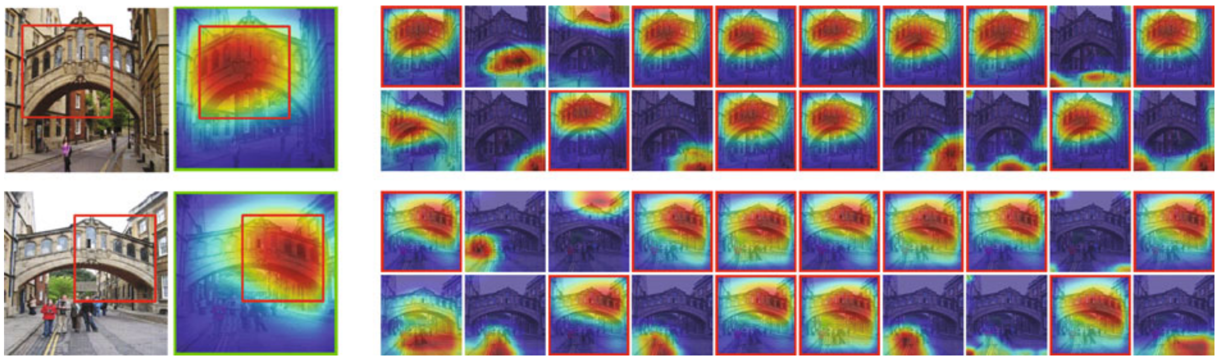
Fig. 3 The leftmost column contains a query image (top) and its top one retrieval result (bottom). The images at the rightmost are the illustration of 20 activation maps (AM) of feature maps from Conv7 (512 in total) corresponding to the leftmost images. Images in the middle are the weighted average activation map M_{avg} obtained from these AMs, which are then used for bit selection in the first layer hash bit selection (those selected are labeled with the red box). Note that not all activation maps focus on the same query-specific salient region as the average map

cal feature map. We converted all feature maps $I_1, I_2, \ldots, I_C$ of Conv7 to activation maps $AM_1, AM_2, \ldots, AM_C$ using the method of [13], and then calculate heat map for each of them, see the rightmost column for some illustration of these. Denote the resulted binary maps as $B_1, B_2, \ldots, B_C$. We then calculate the salience score of each feature map as the overlapping between its heat map with that of the average map

$$s_i(B_i, B_{avg}) = sum(B_i \wedge B_{avg}), \tag{4}$$

where $\wedge$ is a bit-by-bit AND operation, and *sum* represents summing over all elements of the matrix.

At the last stage, we rank the feature maps $I_1, I_2, \ldots, I_C$ according to their scores S and select top L as informative local features. The corresponding L bits of H_l (denoted as H_l') are then chosen for effective retrieval. The index of these L selected bit will serve as a filtering template for the first level hash codes of each of the images to be retrieved. That is, we only need to compare the selected hash bits of the query image with those at the same position for each image i. Formally, let $\Psi_q(*)$ denote the L bits of $*$ at the same L positions as those selected in the query hash codes H_q, we have $H_q' = \Psi_q(H_q)$. Then the Hamming distance between the query and each hash code H_i can be calculated as follows,

$$d_H(H_q', H_i) = d_H(H_q', \Psi_q(H_i)). \tag{5}$$

• **Fast implementation** One bottleneck of runtime complexity of the above bit selection scheme lies in the computation of CAMs and activation maps. Since feature maps focusing on the foreground attention area tend to have a stronger correlation with each other than those focusing on the background, we could exploit this heuristic to dig compact bits in a more efficient way. For this, a correlation analysis on the feature maps is performed. We first vectorize all features maps $\{I_i\}_{i=1}^N$ and construct data matrix F with each column representing a feature map. Next, we calculate the covariance matrix $COV \in \Re^{C\times C}$ of F, where $COV_{i,j} = cov(I_i, I_j)/\sqrt{cov(I_i, I_i)cov(I_j, I_j)}$. This covariance matrix is visualized in Fig. 4. Fortunately, we could find that feature maps activated by salient local regions almost share high correlation and their indexes are approximately consistent with what selected by CAMs. Finally, we rank those feature maps by their correlation coefficients and select top highest feature maps to construct the needed hash code. The time complexity of proposed method will be discussed in Section 4.6.

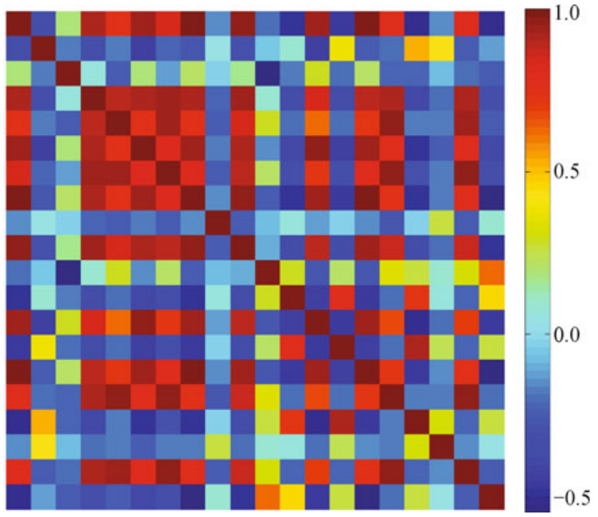

Fig. 4 The result of correlation analysis (CA) of feature maps (We only apply CA on 81st–100th feature maps of the image shown in Fig. 3. The index of the column with strong correlation is consistent with the index of the selected activation maps (with red box annotated) in Fig. 3)

3.3 Searching via multi-level hashing

To perform efficient retrieval, we proposed a hierarchical method that takes advantage of the low computational cost and high accuracy properties using the learned global and local level hashing codes. Particularly, we use the following baseline reranking strategy: first use global-level hash code H_q for efficient rough retrieval, select top K results as candidates, and then use local-level hash code H_L to rerank the results.

We also designed two weighted Hamming distances for similarity evaluation. Denote the query image as x_q, N images to be retrieved as $\{x_i\}_{i=1}^N$, and denote their corresponding global-level hash codes as $\left\{H_{gi}\right\}_{i=1}^N$ and local-hash codes as $\{H_{li}\}_{i=1}^N$, respectively. Then, the Hamming distance (HD) $d_H(\cdot,\cdot)$ between a query and an image is calculated as follows:

1) Weighted distance by linear combination (wLinCb-HD):

$$Sim(x_q, x_i) = \lambda d_H(H_{gq}, H_{gi}) + (1-\lambda) d_H(H'_{lq}, H_{li}), \qquad (6)$$

where λ is a linear combination coefficient set by the users (e.g., 0.5).

2) Weighted distance based on attention (wAtt-HD):

$$Sim(x_q, x_i) = d_{H\_W}(H'_{lq}, H_{li})^{\mathrm{T}} W, \qquad (7)$$

where $W^{\mathrm{T}} \in (0,1)^L$, and its element is computed as follows,

$$w_i = \frac{1}{z}\exp(\frac{s_i}{s_{\max}}), \qquad (8)$$

in which s_i is the salience score of bit i (c.f., Eq. (4)) and,

$$z = \sum_{j=1}^{L} \exp(\frac{s_i}{s_{\max}}),\ s_{\max} = \max(\{s_i\}_{i=1}^C). \qquad (9)$$

In our experiments, we first do the retrieval with global-level hash codes, and then rerank the result with the above two weighted strategies.

4 Experiments

4.1 Datasets and evaluation protocols

To evaluate the performance of proposed method, we use the following benchmark datasets in our experiments for image retrieval.

- **CIFAR10** [29] (CIFAR10) contains 60,000 32 × 32 color images correspond to 10 classes. For each class, it includes 1,000 query images and 5,000 training samples. Hence we have 10,000 queries and 50,000 training images in total.
- **INRIA Holidays dataset** [30] (Holidays) contains 1,491 vacation images, divided into 500 groups. Images in the same group have the same object or scene and the number of images for groups is not balanced. Choosing one image of each group as a query.
- **Oxford Buildings dataset** [31] (Oxford5k) contains 5,062 images, including 55 queries corresponding 11 landmark buildings, and the ground truth relevance of these 11 classes are provided. Similar to Holidays, its images do not evenly distribute in these 11 classes. Although the region of interest (ROI) of 55 queries are annotated, we did not use them in our experiments and simply took the whole image as the query.
- **Oxford Buildings dataset+100K** [31] (Oxford105k) includes Oxford Buildings dataset (5k) and extra 100K images from Flickr as the distractor.
- **Paris Buildings dataset** [32] (Paris6k) contains 6,412 images associated with Paris landmarks, among which 55 are queries corresponding to 11 buildings with ground truth relevance provided. We did not use the annotated ROI information of query in our experiments.
- **University of Kentucky Benchmark dataset** [33] (UKB) includes 10,200 indoor object images. Each object contains four images, and each image is used to query the rest. The performance is measured by the average number of same object images within the top four results. In our experiment, we select one image of each object as the query (2,550 images in total) and use the rest as training samples (7,650 images in total).

Especially, the images of Holidays dataset contain complex scene or object, and the training examples are too few to feed CNNs which easily lead to over-fit. The main challenge is that there exists only one training sample for some groups exclude query image resulting in no positive pairs for these groups. In order to address this problem, as Babenko et al. [6] done, we firstly collect extra relevant images for these groups by image search engine, for each query we eyeball the returned images and download most relevant photographs from the top of the response. We totally obtain 652 images, to preserve the distribution of original dataset as possible, we then manually select top eight or fewer images as added training samples for the group with single training data. However, we found that several groups still have no additional similar images through the first stage. Therefore, we adopt one near-duplicate transformation, e.g., crop or rotation, on the single images to generate positive pairs for these groups. We finally obtain 165 added images for training. The usefulness of these added images could be understood in two ways: 1) they encode our prior knowledge of our task by saying that introducing these sets of transformations onto our data should not alter the output of the network; and 2) these added images could introduce useful noise into training, acting as a regularizer. In other words, such noise could actually help our model to escape from over-fitting.

Except on the UKB dataset, we adopt the commonly-used mean average precision (*mAP*) as the performance metric, which is defined as follows:

$$\begin{aligned} P@n &= \frac{\#\{relevant\, images\, in\, top\, N\, results\}}{N}, \\ AP &= \frac{\sum_n P@n \times I\{image\, n\, is\, relevant\}}{\#\{retrieved\, relevant\, image\}}, \\ mAP &= \frac{1}{Q}\sum_i AP_i, \end{aligned} \tag{10}$$

where # is count function, I is an indicator function, and Q represents the total number of queries.

4.2 Implementation details

We implement the proposed DSCNN with the Caffe [34] package, in which we modify the AlexNet architecture by maintaining the top four layers, altering the number of filters at layer 5 to 384, adding conv6, conv7 layer (both with 512 filters, 3×3 size, 1 pad) and an average-pooling layer (11×11 size, 11 stride), and replacing the Fc6-Fc7 layers with hash layer (48 nodes). We also implement the Hash Constraints loss layer. The activate function of Conv7 is the tanh function (sigmoid was also considered but proven to be not so effective) while those of other convolution layers are ReLu. In this architecture, an extra average-pooling layer is added so as to approximate the binary codes for the first layer of hash code H_L and the learned weights from this layer to the second hash layer are used to calculate CAMs, as described in Section 3. We binarized CAMs with threshold $\theta = 0.6$.

Table 1 gives the details of our network. The SoftmaxLoss layer is usually connected with a fully-connect layer, and the number of nodes for this layer is set to be the number of object categories in the target dataset.

All images are resized to 256×256 before passing through the network. In training phase, we randomly select positive and negative pairs at a ratio of 5:2 approximately from the training set and the query images are not used for training. The number of positive pairs for each category is almost equal. If two images share the same label, then they could construct a positive pair, otherwise a negative pair. For CIFAR10 dataset, we sample 100,000 positive pairs and 50,000 negative pairs from 50,000 training examples. We use the same training set for Oxford5k and Oxford105k, which includes 65,422 pairs and the positive pair is defined by the provided relevance information. We also construct 53,235 positive and 11,088 negative pairs for Pairs6k. For Holidays, totally 49,550 positive and 23,784 negative pairs are constructed. Finally, we build 76,500 positive and 22,950 negative pairs with 7,650 training samples from UKB. For training effectively, we adopt fine-tune strategy over all six datasets and initialize the weights of top four layers (Conv1–Conv4) of DSCNN with pre-trained AlexNet model on ImageNet (provided by Caffe), Conv5–Conv7 and Fc8 with Gaussian distribution (std are 0.01 and 0.005 respectively). For parameter setting of the loss function, we penalize classification and pair-wise loss equally and discount the importance hash constraints loss by setting c to 1, α to 1, β, γ to both 0.1 in Eq. (2). We set the iteration number to 100,000 and decay the base learning rate by 90% every 10,000 step from 0.01 or 0.001. The batch size is set to 32, momentum to 0.9, and weight decay to 0.0005.

Table 1 Model description of the proposed DSCNN

Layer	Details
Data	256 × 256 RGB image cropped to 227 × 227
Conv1	11 × 11 96 ReLU. stride 4
Pool1	3 × 3 Max. stride 2
Conv2	5 × 5 256 ReLU. stride 1
Pool2	3 × 3 Max. stride 2
Conv3	3 × 3 384 ReLU. stride 1
Conv4	3 × 3 384 ReLU. stride 2
Conv5	3 × 3 384 ReLU. stride 2
Pool5	3 × 3 Max. stride 1
Conv6	3 × 3 512 ReLU. stride 1
Conv7	3 × 3 512 TanH. stride 1
Pool7	11 × 11 Ave. stride 11
Fc8 | L1	48 TanH | SoftmaxLoss
L2	Softmax/HashConstraints/ContrastiveLoss

In the test phase, we extract global hash code (48 bits) from the second hash layer and local hash code (512 bits) from the average-pooling layer (the first hash layer). Mention that, we need to modify the numbers of nodes of hash layers and retrain models to obtain global hash codes with variable length. As we discussed in Section 3, we only select compact local bits of query images. For each query, 256 bits of a hash mask is first learned and is applied to every image to be compared. All experiments were run on a workstation with Xeon E5 CPU 16G memory and a Nvidia Titan X GPU.

4.3 Comparison with state-of-the-art non-deep hashing methods

We firstly compare our proposed hierarchical deep hashing (HDH) method with several state-of-the-art non-deep hashing methods. For our method, we firstly do retrieval with corresponding bits H_g code, then rerank top 5,000 ranks with selected 256 bits H_l code by different strategies: HDH (baseline) without weighted strategies, HDH (wLinCb-HD) and HDH (wAtt-HD) with weighted strategies. We category non-deep hashing methods into unsupervised hashing methods and supervised hashing methods. The unsupervised hashing methods include one data-independent method, local sensitive hashing (LSH [14]), and four data-dependent hashing method, iterative quantization (ITQ [35]), spectral hashing (SH [36]), spherical hashing (SpH [37]) and scalable graph hashing (SGH [38]). The hash function of LSH is a set of random project hyperplane, while other four data-dependent methods learn hash function based on data similarity in feature space. The supervised hashing methods include fast supervised hashing (FastHash [39]), supervised discrete hashing (SDH [40]), supervised hashing with kernels (KSH [17]), supervised hashing with latent factor models (LFH [41]) and column sampling based discrete supervised hashing (COSDISH [42]). For the fairness of the experiment, all these non-deep methods are conducted with 4096-dim deep features, represented by "+CNN", and the features are extracted from the last fully-connected layer of AlexNet pre-trained on ImageNet. Since most of the results in these papers are on the CIFAR10 [29], for a fair comparison, we also conduct the experiments on this dataset.

Figure 5 shows the results for different unsupervised and supervised hash methods with 12, 24, 32 and 48 bits, respectively. As we can see, our HDH method outperforms the other compared non-deep hashing methods.

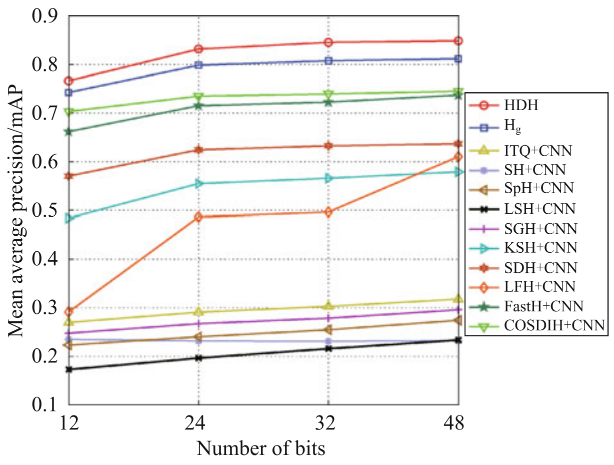

Fig. 5 Mean average precision on the CIFAR10 dataset (Five unsupervised and five supervised non-deep hashing methods with CNN feature are compared)

Figure 6 gives some illustration of the retrieval examples, where we used 48 bits hash codes for different methods. Compared with COSDISH and FastH methods, although both results are seemly good, e.g., top retrievals are relevant, our method seems to work the best. One explanation is that our HDH effectively embeds high-level semantic relevance into hash codes and exploits the local property of hash codes to further improve the retrieval performance.

4.4 Comparison with state-of-the-art deep hashing methods

To further assess the performance of the proposed hierarchi-

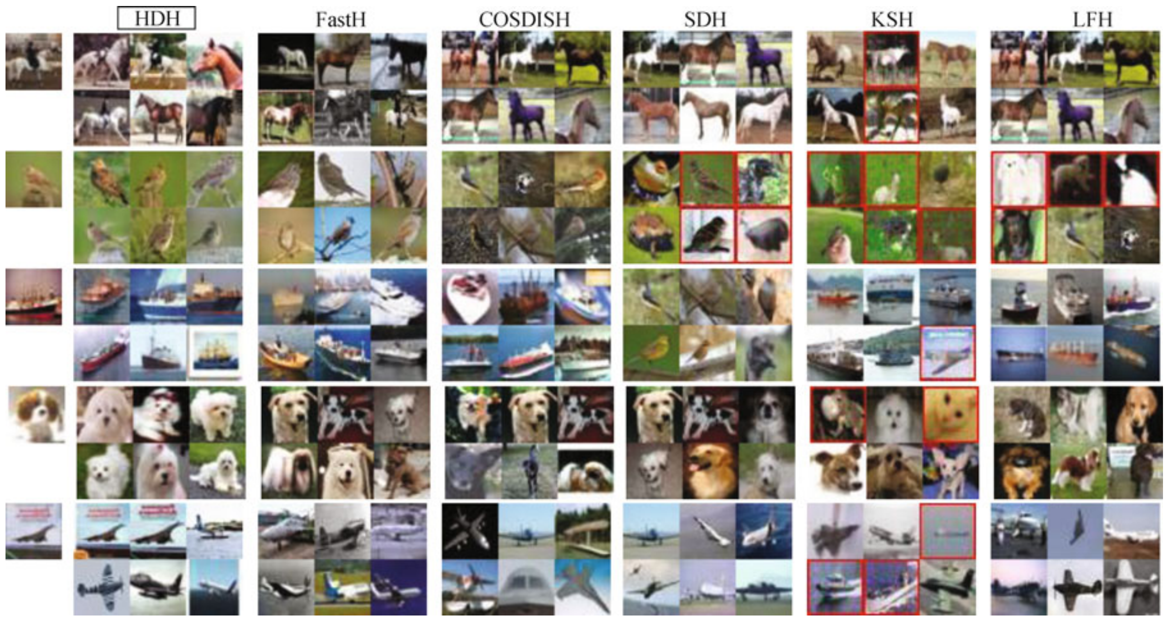

Fig. 6 Top six images of five queries are retrieved with different supervised hashing methods on the CIFAR10 dataset (The image on the first column is the query sample. From left to right are the retrieved results by HDH, FastH, COSDISH, SDH and LFH when 48-bit binary codes are used for search)

cal deep hashing (HDH) method, we compare it with several closely related state-of-the-art deep hashing methods, including convolutional neural network hashing (CNNH [19]) and its improved version CNNH+, deep pairwise-supervised hashing (DPSH [20]), network-in-network hashing (NINH [26]), deep supervised hashing (DSH [21]), deep regularized similarity comparison hashing (DRSCH [43]) and its simplified variant DSCH, and supervised deep hashing (SDH [15]). We also conduct the experiments on CIFAR10 dataset. Note that for a fair comparison, because above methods did not use the rerank strategy and we only use H_g hash code with varying lengths for retrieval. On the other hand, it is easy to incorporate our strategy into existing deep hashing structures. To verify this, we re-implement the DSH structure by modifying the activate function of the two fully-connect layers to tanh, and imposing L_1 and L_2 loss to them respectively. Then we trained the model on our pairwise training samples, using the same training setting of hyper-parameters (e.g., learning rate, decay rate) as that of DSH [21]. We take the output of the second fully-connect layer as the global hash code ($HDH\_shallow_{global}$) after quantization and take the output of the first fully-connect layer as the local hash code (500 bits), which is used for reranking ($HDH\_shallow_{rerank}$).

Table 2 gives results. One can see that the proposed hierarchical deep hashing with attention-based weighted hamming distance (i.e., HDH (wAtt-HD)) obtain the best performance, and it improves the closest competitor — deep pairwise-supervised hashing (DPSH [20]) by 10.0% to 15.0%. DRSCH [43] and NINH [26] both use the triplet loss to train their networks, which is similar to the siamese network adopted here in some sense, but the table reveals that our method yields 6% to 20% performance advantage over these two even with our most compact H_g code. Indeed, the performance gaps between deep hashing methods may be caused by their different loss functions, both DSH and DPSH use pairwise as objectives and perform superior to triplet-based methods DRSCH, NINH, which indicates pairwise constraints could depict similarity more powerful than triplet ones in some cases. While ours adopt extra classification constraint to capture supplementary semantic information to boost state performance.

The retrieval performance may have a strong connection with the underlying deep learning architectures. In particular, convolutional neural network hashing (CNNH [19]) uses several convolutional layers and two fully-connected layers to fit pre-designed binary codes. Supervised deep hashing (SDH [15]) uses three fully-connected layers without any convolutions and adopts hand-crafted features as inputs. Deep pairwise-supervised hashing (DPSH [20]) and network-in-network hashing (NINH [26]) share a similar loss function, but DRSCH is composed of three convolutional layers and three fully-connected layers while NINH only contains four convolutional layers. Comparing $HDH\_shallow_{global}$ and H_g, they give different retrieval performance although they are trained with the same loss function and with the same set of training samples. The re-ranking strategy is also useful, as verified by the fact that $HDH\_shallow_{rerank}$ improves the performance of $HDH\_shallow_{global}$ by 2.9%.

Table 2 Comparison with state-of-the-art deep hashing methods on CIFAR10

Method	12bits	24bits	32bits	48bits
CNNH [19]	43.9	51.1	50.9	52.2
CNNH+ [19]	46.5	52.1	52.1	53.2
SDH [15]	-	-	20.83	-
NINH [26]	55.2	56.6	55.8	58.1
DSCH [43]	-	61.3	61.7	61.9
DRSCH [43]	-	62.1	62.8	63.0
DSH [21]	61.5	65.1	66.7	67.5
DPSH [20]	68.2	68.6	72.5	73.3
H_g code	74.3	79.9	80.8	81.2
$HDH\_shallow_{global}$	-	-	-	37.2
$HDH\_shallow_{rerank}$	-	-	-	40.1
HDH (baseline)	77.2	83.2	84.6	84.9
HDH (wLinCb-HD)	75.2	82.3	84.0	**86.3**
HDH (wAtt-HD)	**77.8**	**83.9**	**84.9**	85.9

Note that in our HDH method with weighted Hamming distance, the actual code length used is the corresponding length of H_g code (for rough search) plus the 256 bits of local hash code (for result refining). Among the three HDH variants compared here, the HDH(wAtt-HD) strategy works the best, while the linear combination method HDH improves the performance of baseline method HDH(baseline) by 1.6% when the code-length of H_g is 48. All the three hierarchical search strategies are beneficial to the performance compared to the retrieval scheme that is directly based on the H_g code, revealing that they could fuse local and global information and make the resulting retrieval scheme more discriminative and more robust while being as efficient as possible. Figure 7 gives some examples on how the hierarchical search improves the ranking accuracy over the naive H_g code.

4.5 Comparison with state-of-the-art non-hashing methods

We next conduct experiments on other five datasets, including Oxford5k [31], Oxford105k [31], Paris6k [32], Holidays [30], and UKB [33]. Specifically, besides the proposed hierarchical deep hashing method (HD), we evaluate several variants of the proposed method, including the global hash

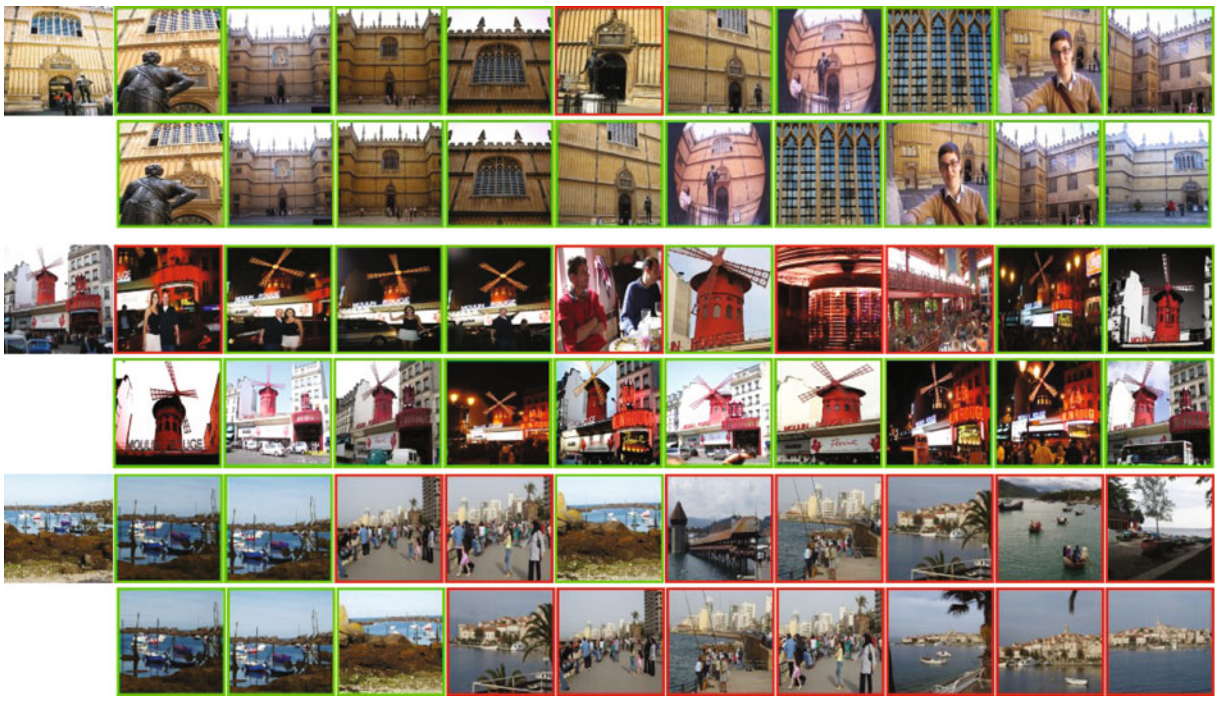

Fig. 7 Illustration of using local-level hash for reranking (For each query image (the first column), images in the first row are the top ten results retrieved with global-level H_g hash code, while the images below are results adjusted by using the proposed multi-level hash search method)

code (H_g), local hash codes (H_l) without pruning ($H_l full$), with attention-based bit selection ($H_l selected$), and fast implementation ($H_l fast$). For comparison, we also test a PCA-based dimensionality reduction method for the Conv7 layer, denoted as Conv7-PCA. We compare our method with several states-of-the-art non-hashing CNN methods popularly used for image retrieval. Particularly these methods can be roughly divided into two categories: the first category are three sophisticated aggregation methods, including those based on Fisher vectors [8], VLAD [44] and Triangulation embedding of traditional local descriptors (T-embed [45]). The second category mainly includes methods that are based on various variants of CNN features, such as CNN-based VLAD (C-VLAD [10]), OxfordNet [46], CNN spatial search (CNN-ss [7]), regional maximum activation of convolution (R-MAC [9]), cross-dimensional weighting (CDW [23]), and Faster R-CNN [24].

Table 3 gives the results (Note that the number appeared in the parentheses of the first column are dimensionality of the corresponding method). From the table, several observations can be made: First, one can see that methods using current state-of-the-art sophisticated aggregation feature descriptors [8,45] yield similar performance to some of the state-of-the-art deep feature descriptors network [10,46,8] on the task of image retrieval. However, their performance is inferior to that of more advanced CNN variants, such as R-MAC [9], CDW [23], and Faster R-CNN [24], while our hierarchical deep hashing method (HD) yields better or competitive results compared to these state-of-the-art non-hashing methods. Particularly on the challenging large-scale Oxford105k dataset, our method with attention-based weighted distance yields mAP of 63.5%, outperforming the previous best performer R-MAC [9] by 1.9% without using extra complex retrieval strategies such as spatial reranking or query expansion as the latter does. For Holidays dataset, images are composed of complex scenes. CDW encodes prior knowledge into the aggregate process and C-VLAD adopts VLAD on complete feature maps with less loss of scene information, while our average strategy may introduce more noise for this situation. Therefore, our HDH performs inferior to C-VLAD and CDW.

To evaluate the impact of binarization, in Table 3 we also give the retrieval performance without hashing, these are respectively denoted as F_l *full* (using the outputs of pool7), F_g *baseline* (using the output of hash layer but not performing binarization), and F_l *selected* (using the features before hashing to H_l *selected*). We use cosine distance to measure the similarity of these continuous feature sets. Comparing these results with their binarized versions, we see that as expected, binarization reduces the retrieval performance, but the degrees of influence are different from dataset to dataset. It seems that the performance is less sensitive to binarization on the datasets of Holidays and UKB, while their behaviors are much different on the Paris6k and Oxford105k. This highlights the needs of further research on narrowing the performance gap due to feature binarization.

Table 3 Comparison with state-of-the-art non-hashing methods (mAP)

Method	Holidays	Oxford5k	Oxford105k	Paris6k	UKB
Fisher Vec. [8] (256)	74.7	54.0	-	-	-
T-embed. [45] (1,024)	72.0	56.0	50.2	-	3.51
VLAD [8] (128)	62.5	44.8	37.4	55.5	-
C-VLAD [10] (128)	**81.6**	55.8	-	58.3	-
OxfordNet [46] (256)	71.6	53.3	48.9	67.0	3.36
SPC [8] (256)	80.2	58.9	57.8	-	3.65
Neural Codes [6] (256)	78.9	55.7	52.4	-	3.42
CNN-SS [7] (32k)	-	55.6	-	69.7	3.47
R-MAC [9] (512)	-	66.9	61.6	83.0	-
CDW [23] (256)	81.5	65.4	59.3	77.9	-
FasterRCNN [24] (4,096)	-	67.8	-	78.4	-
F_g *baseline* (48)	65.1	61.7	62.3	74.1	2.92
F_l *selected* (256)	78.9	70.1	67.8	88.8	3.68
F_l *full* (512)	81.0	**71.2**	**70.4**	**89.3**	**3.74**
Conv7+PCA (256)	62.9	58.6	55.7	68.6	3.1
H_g *baseline* (48)	64.4	59.3	58.2	69.2	2.84
H_l *fast* (256)	71.5	67.3	60.1	77.8	3.52
H_l *selected* (256)	78.2	69.7	63.9	85.2	3.67
H_l *full* (512)	80.8	70.5	65.1	87.3	3.71
HDH(baseline) (294)	77.2	67.1	63.3	83.7	3.62
HDH(wLinCb-HD) (294)	78.1	67.2	62.8	83.4	3.42
HDH(wAtt-HD) (294)	78.2	67.7	63.5	84.1	3.62

One advantage of our method compared to these CNN-based non-hashing methods is that we use binarized hash codes instead of continuous features for image representation, hence being much more efficient in terms of computational and storage cost. Under the situation when the retrieval efficiency is not so critical, using our hash descriptors (e.g., $H_l(full)$) directly without performing retrieval first using H_g yields the best performance among the methods compared. Particularly, on the large-scale Oxford105k dataset, the mAP performance of our $H_l(full)$ descriptor is 65.1%, 14.9% higher than that of the best traditional features [45] and 7.3% higher than that of the best CNN descriptors [45]. Similar observations can be made on other datasets.

Another interesting point revealed by the table is that, although the three variants of the proposed method achieve similar performance on all the datasets tested, the attention-based weighted Hamming distance seems to work the best among them. One possible reason for this is that the local salience weighting factors help to highlight the attention region of the image to be compared, hence being more robust against the noise of the irrelevant region.

4.6 Analysis of time cost

In this section, we investigate the running time complexity of our method empirically. In particular, we report the time for feature extraction (denoted as “GPU” and “CPU”), bits selection (denoted as “CAM”, “AM”, and “Fast”), and retrieval (denoted as “Search”, “Rerank”), respectively. The feature extraction stage is tested on both the CPU mode and the GPU mode. The experiments are conducted on the Paris6k dataset with 24-bit and 48-bit codes respectively.

Figure 8 gives the logarithmic time (in microseconds, base 10) of different components, where the results were averaged over the whole test set. As we can see, varying the length of codes take almost the same amount of time on the feature extraction stage, because of using the common convolution layers. The GPU-accelerated mode is 10x faster than the CPU mode as excepted. Although the stage of calculating CAM and AM consumes more time than the stage of feature extraction, the fast version shortens the time cost by nearly two orders of magnitudes, which makes it possible to real-world applications. Since not all candidates are involved in the reranking stage, it is much faster than rough search and hence the query time with or without rerank strategy is at the same order of magnitude.

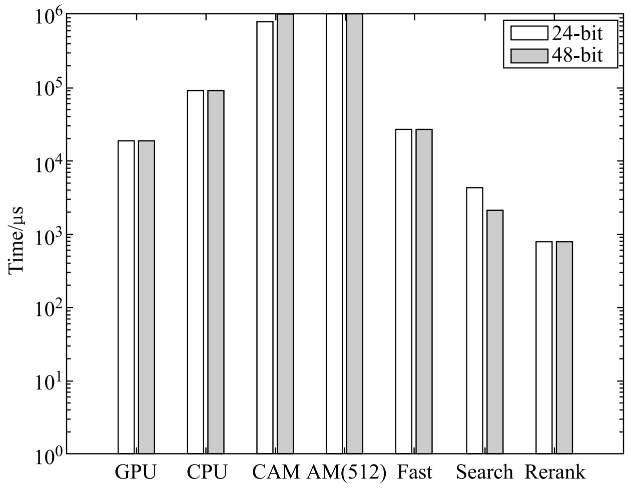

Fig. 8 Time cost to query (microseconds) on Paris6k

5 Conclusion

This paper presents a deep siamese CNN to produce global-level and local-level semantic-preserved hash codes for image retrieval. Meanwhile, in order to reduce the dimension of local-level hash code while maintaining its discriminative capability, we propose a method that selects hash bits most relevant to a specific query image based on the attention mechanism. We also give a fast implementation of this using correlation analysis. Finally, we present an efficient hierarchical search that combines the advantages of both local and global hash code. Extensive results over several benchmark datasets verify the effectiveness of the proposed method.

Acknowledgements This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 61373060 and 61672280) and Qing Lan Project.

References

1. Smeulders A W M, Worring M, Santini S, Gupta A, Jain R. Content-based image retrieval at the end of the early years. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2000, 22(12): 1349–1380
2. Krizhevsky A, Sutskever I, Hinton G E. ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 2012, 25(2): 2012
3. Girshick R, Donahue J, Darrell T, Malik J. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014, 580–587
4. Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 1337–1342
5. Zheng L, Yang Y, Tian Q. SIFT Meets CNN: a decade survey of instance retrieval. 2016, arXiv preprint arXiv:1608.01807
6. Babenko A, Slesarev A, Chigorin A, Lempitsky V. Neural codes for image retrieval. In: Proceedings of European Conference on Computer Vision. 2014, 584–599
7. Razavian A S, Azizpour H, Sullivan J, Carlsson S. CNN features off-the-shelf: an astounding baseline for recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2014, 512–519
8. Babenko A, Lempitsky V. Aggregating deep convolutional features for image retrieval. In: Proceedings of the IEEE Conference on Computer Vision. 2015, 1269–1277
9. Tolias G, Sicre R, Jégou H. Particular object retrieval with integral max-pooling of CNN activations. Computer Science, 2015
10. Ng Y H, Yang F, Davis L S. Exploiting local features from deep networks for image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 53–61
11. Zheng L, Zhao Y L, Wang S J, Wang J D, Tian Q. Good practice in CNN feature transfer. 2016, arXiv preprint arXiv:1604, 00133
12. Zheng L, Wang S J, Wang J D, Tian Q. Accurate image search with multi-scale contextual evidences. International Journal of Computer Vision, 2016(1): 1–13
13. Zhou B L, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning deep features for discriminative localization. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, 2921–2929
14. Andoni A, Indyk P. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science. 2006, 459–468
15. Liong V E, Lu J W, Wang G, Moulin P, Zhou J. Deep hashing for compact binary codes learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 2475–2483
16. Zhao F, Huang Y Z, Wang L, Tan T N. Deep semantic ranking based hashing for multi-label image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 1556-1564
17. Chang S F. Supervised hashing with kernels. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2012, 2074–2081
18. Gong Y C, Pawlowski M, Yang F, Brandy L, Bourdev L, Fergus R. Web scale photo hash clustering on a single machine. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 19–27
19. Xia R K, Pan Y, Lai H J, Liu C, Yan S C. Supervised Hashing for Image Retrieval via Image Representation Learning. In: Proceedings of the 28th AAAI Conference on Artificial Intelligence. 2014
20. Li W J, Wang S, Kang W C. Feature learning based deep supervised hashing with pairwise labels. Computer Science, 2015
21. Liu H M, Wang R P, Shan S G, Chen X L. Deep supervised hashing for fast image retrieval. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2016, 2064–2072
22. Paulin M, Douze M, Harchaoui Z, Mairal J, Perronin F, Schmid C. Local convolutional features with unsupervised training for image retrieval. In: Proceedings of the IEEE International Conference on Computer Vision. 2015, 91–99
23. Kalantidis Y, Mellina C, Osindero S. Cross-dimensional weighting for aggregated deep convolutional features. In: Proceedings of European Conference on Computer Vision. 2016, 685–701
24. Salvador A, Giroinieto X, Marques F, Satoh S I. Faster R-CNN features for instance search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2016, 9–16
25. Lin K, Yang H F, Hsiao J H, Chen C S. Deep learning of binary hash codes for fast image retrieval. In: Proceeding of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. 2015, 27–35
26. Lai H J, Pan Y, Liu Y, Yan S C. Simultaneous feature learning and hash coding with deep neural networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 3270–3278
27. Zeiler M D, Fergus R. Visualizing and understanding convolutional networks. In: Proceedings of European Conference on Computer Vision. 2013, 818–833
28. Mahendran A, Vedaldi A. Understanding deep image representations by inverting them. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 5188–5196
29. Krizhevsky A. Learning multiple layers of features from tiny images. Technical Report. 2012
30. Jegou H, Douze M, Schmid C. Hamming embedding and weak geometric consistency for large scale image search. In: Proceedings of European conference on computer vision. 2008, 304–317
31. Philbin J, Chum O, Isard M, Sivic J, Zisserman A. Object retrieval with large vocabularies and fast spatial matching. In: Proceedings of the IEEE International Conference on Computer Vision. 2007, 1–8
32. Philbin J, Chum O, Isard M, Sivic J, Zisserman A. Lost in quantization: improving particular object retrieval in large scale image databases. In:

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2008, 1–8

33. Nister D, Stewenius H. Scalable recognition with a vocabulary tree. In: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2006, 2161–2168

34. Jia Y Q, Shelhamer E, Donahue J, Karayev S, Long J, Girshick R, Guadarrama S, Darrell T. Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM international conference on Multimedia. 2014, 675–678

35. Gong Y C, Lazebnik S. Iterative quantization: A procrustean approach to learning binary codes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2011, 817–824

36. Weiss Y, Torralba A, Fergus R. Spectral hashing. In: Proceedings of the Neural Information Processing Systems Conference. 2008, 1753–1760

37. Heo J P, Lee Y, He J, Chang S F, Yoon S E. Spherical hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2012, 2957–2964

38. Jiang Q Y, Li W J. Scalable graph hashing with feature transformation. In: Proceedings of the International Conference on Artificial Intelligence. 2015, 331–337

39. Lin G S, Shen C H, Shi Q F, van den Hengel A, Suter D. Fast supervised hashing with decision trees for high-dimensional data. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014, 1971–1978

40. Shen F M, Shen C H, Liu W, Shen H T. Supervised discrete hashing. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2015, 37–45

41. Zhang P C, Zhang W, Li W J, Guo M Y. Supervised hashing with latent factor models. In: Proceedings of the 37th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2014, 173–182

42. Kang W C, Li W J, Zhou Z H. Column sampling based discrete supervised hashing. In: Proceedings of the 30th AAAI Conference on Artificial Intelligence. 2016

43. Zhang R M, Lin L, Zhang R, Zuo W M, Zhang L. Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. IEEE Transactions on Image Processing, 2015, 24(12): 4766–4779

44. Arandjelovic R, Zisserman A. All about VLAD. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2013, 1578–1585

45. Jégou H, Zisserman A. Triangulation embedding and democratic aggregation for image search. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2014, 3310–3317

46. Razavian A S, Sullivan J, Carlsson S, Maki A. Visual instance retrieval with deep convolutional networks. 2014, arXiv preprint arXiv:1412.6574

Ge Song received his BS degree in computer science and technology from Zhengzhou University, China in 2014. Now he is a PhD student in Nanjing University of Aeronautics and Astronautics, China. His research interests are in image retrieval, machine learning, pattern recognition, and computer vision.

Xiaoyang Tan received his BS and MS degrees in computer applications from Nanjing University of Aeronautics and Astronautics (NUAA), China in 1993 and 1996, respectively. Then he worked at NUAA in June 1996 as an assistant lecturer. He received a PhD degree from Department of Computer Science and Technology of Nanjing University, China in 2005. From September 2006 to October 2007, he worked as a postdoctoral researcher in the LEAR (Learning and Recognition in Vision) team at INRIA Rhone- Alpes in Grenoble, France. His research interests are in face recognition, machine learning, pattern recognition, and computer vision. In these fields, he has authored or coauthored over 40 scientific papers.