International Journal of Computer Vision
https://doi.org/10.1007/s11263-020-01315-0

Learning Multifunctional Binary Codes for Personalized Image Retrieval

Haomiao Liu[1,2,3] · **Ruiping Wang**[1,2] · **Shiguang Shan**[1,2] · **Xilin Chen**[1,2]

Received: 15 April 2019 / Accepted: 21 February 2020

Abstract
Due to the highly complex semantic information of images, even with the same query image, the expected content-based image retrieval results could be very different and personalized in different scenarios. However, most existing hashing methods only preserve one single type of semantic similarity, making them incapable of addressing such realistic retrieval tasks. To deal with this problem, we propose a unified hashing framework to encode multiple types of information into the binary codes by exploiting convolutional networks (CNNs). Specifically, we assume that typical retrieval tasks are generally defined in two aspects, i.e. high-level semantics (e.g. object categories) and visual attributes (e.g. object shape and color). To this end, our Dual Purpose Hashing model is trained to jointly preserve two kinds of similarities characterizing the two aspects respectively. Moreover, since images with both category and attribute labels are scarce, our model is carefully designed to leverage the abundant partially labelled data as training inputs to alleviate the risk of overfitting. With such a framework, the binary codes of new-coming images can be readily obtained by quantizing the outputs of a specific CNN layer, and different retrieval tasks can be achieved by using the binary codes in different ways. Experiments on two large-scale datasets show that our method achieves comparable or even better performance than those state-of-the-art methods specifically designed for each individual retrieval task while being more compact than the compared methods.

Keywords Image retrieval · Multi-task learning · Hashing

1 Introduction

In recent years, more and more images are available on the Internet, posing great challenges to retrieving images that are relevant to a given query image. Due to the high efficiency in storage and computation, hashing has become increasingly popular in such large-scale image retrieval tasks (Gionis et al. 1999; Gong and Lazebnik 2011; Xia et al. 2014; Liu et al. 2016a; Yang et al. 2015). The general idea of hashing methods is to project images into binary codes, where the similarity relationships between images are preserved in the Hamming space. However, in real-world applications, the expected retrieval results vary in different situations. For instance, three typical personalized retrieval tasks are shown in Fig. 1. Given the same query image on the left, one might want to search for (**i**) photos of the same person (Liu et al. 2016a; Yang et al. 2015) (category retrieval); (**ii**) pictures that are visually similar to the query image in terms of some specific visual attributes (Siddiquie et al. 2011; Scheirer et al. 2012) (attribute retrieval); (**iii**) photos of the same person with additional specified visual attributes, e.g. smiling (combined retrieval). Therefore, existing hashing methods, which only preserve one specific kind of similarity, cannot satisfy the numerous possible personalized expectations simultaneously.

Communicated by Li Liu, Matti Pietikäinen, Jie Qin, Jie Chen, Wanli Ouyang, Luc Van Gool.

✉ Ruiping Wang
wangruiping@ict.ac.cn

Haomiao Liu
haomiao.liu@vipl.ict.ac.cn

Shiguang Shan
sgshan@ict.ac.cn

Xilin Chen
xlchen@ict.ac.cn

[1] Key Laboratory of Intelligent Information Processing, Chinese Academy of Sciences (CAS), Institute of Computing Technology, CAS, Beijing 100190, China

[2] University of Chinese Academy of Sciences, Beijing 100049, China

[3] Huawei EI Innovation Lab, Beijing 100085, China

Published online: 17 March 2020

Fig. 1 A real example showing the three retrieval tasks on a face dataset. The top-ranked feedbacks of each task are shown here. In the first two rows, exactly matched images are bounded by green boxes, and red otherwise. In the last two rows, images of the same/different identity are bounded by green/red boxes respectively, and the blue bars indicate the confidence level of the corresponding attribute. Best viewed in color (Color figure online)

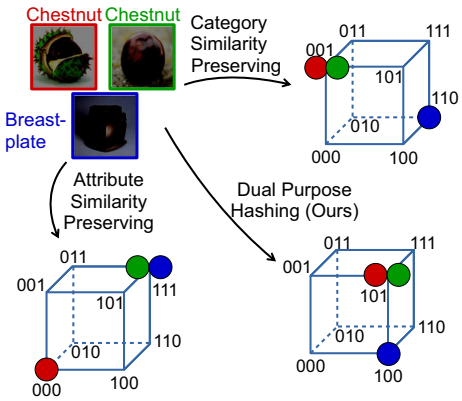


Fig. 2 Illustration of the idea of our Dual Purpose Hashing method. Unlike existing hashing methods that only preserve one kind of similarity (e.g. category similarity or attribute similarity), our method simultaneously preserves multiple types of similarity, making it applicable for multiple realistic retrieval tasks

One straightforward solution is to use multiple models, where each model preserves one type of similarity. However, such a solution could be very time-consuming in practice when the retrieval task at hand involves multiple types of information (e.g. the last two rows in Fig. 1), and the redundancies between different tasks might harm the storage efficiency. To address this problem, we propose a unified hashing framework, named Dual Purpose Hashing (DPH), to simultaneously encode multiple types of information into the binary codes. Specifically, we can see from Fig. 1 that the expected retrieval results are typically defined by two aspects of the image, i.e. high-level semantics (e.g. person identity or object category) and mid-/low-level visual attributes (e.g. gender or object shape and color), which can be abstracted as "category" and "attribute" respectively. Therefore, unlike existing hashing methods that only preserve one specific similarity relationship, the proposed DPH framework encodes both category and attribute information into the binary codes, as illustrated in Fig. 2.

Our basic idea comes from a very natural intuition that category and attributes, as objects' descriptions at different abstraction levels, should share some common low-level visual features. The successful applications of CNN models in many computer vision tasks (He et al. 2016; Liu et al. 2019, 2020) have promoted more research works to investigate the working mechanism of such models. Experimental studies in some recent works (Escorcia et al. 2015; Zhong et al. 2016; Bau et al. 2017) show that some nodes in the CNN model trained for classification tasks are highly correlated with visual attributes, which could partly confirm the correctness of our basic idea. Such observations also suggest that deep CNN model is a good choice to hierarchically capture the correlations between category and attributes. This motivates us to adopt CNN models to learn unified binary codes that can preserve both similarities simultaneously.

The framework of our DPH method is illustrated in Fig. 3. To be specific, since directly learning binary codes in Hamming space is NP-hard, our network adopts a binary-like layer to approximate the real binary codes. By jointly optimizing a category information encoding loss and an attribute prediction loss, our method can encode both similarities into the unified binary codes. Specifically, our method is compatible with multiple different forms of losses, and thus two widely studied forms for each loss are considered in our framework, which will be detailed in the following sections. Moreover, since most images available on the Internet do not have complete category and attribute labels and labelling them could be very costly, our loss function is properly designed to take into account such practical scenarios. Namely, each loss only takes images with the corresponding labels as training inputs, and thus even images with only one label can contribute to the model learning. By doing so, an additional benefit is that the network has the capacity to see a large amount of partially labelled data in the training stage, and thus greatly reduces the risk of overfitting.

Once the model training is done, images can be indexed by quantizing the outputs of the binary-like layer to compact hash codes. In the category retrieval task, retrieval can be done similarly to existing hashing methods by utilizing Hamming distance ranking or hash table lookup. For the other two tasks (attribute retrieval and combined retrieval), the attribute information of each image can be efficiently recovered from the binary codes and is utilized to perform the retrieval tasks accordingly. Compared with directly storing the attribute information, our method only incurs a little increase in computational cost, while dramatically reduces the storage space cost.

The main contributions of this work are threefold: **First**, we present a unified framework to learn hash functions that

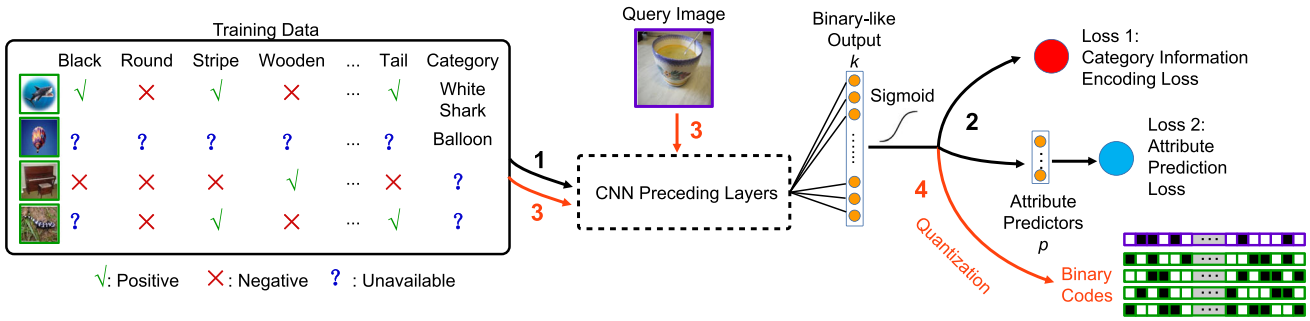


Fig. 3 The framework of our DPH method. To simultaneously encode category and visual attributes of images into binary codes, we devise a CNN model that can take partially labelled images as training input (step 1), and train the model on classification/metric learning and attribute prediction tasks (step 2). The binary-like output layer, which has k (the code length) nodes, is connected to the two task layers as input. To produce real binary codes, images are propagated through the network (step 3), and the binary-like network output is quantized (step 4)

simultaneously preserve category and attribute similarities for addressing multiple retrieval tasks. **Second**, we show that by jointly preserving multiple types of similarities, the model can make use of the rich relationships between different tasks to suppress the redundancies and learn more compact binary codes than learning multiple models separately. **Third**, we propose a new training scheme for the CNN models that can take partially labelled data as training inputs to improve the performance and alleviate overfitting.

Preliminary results of the proposed method have been published in Liu et al. (2017). Compared with the conference version, this paper has made four major extensions: **First**, we generalize our framework to be compatible with more loss functions (e.g. triplet ranking loss and hinge loss) that have been widely adopted in hashing methods, and extensively evaluate their performances on our tasks. **Second**, we adjust the evaluation metrics on the attribute-related tasks to better take into account the sample imbalance problem. **Third**, more experiments are conducted to give more insight into our proposed framework, including modifying the network architecture to better cope with the multiple retrieval tasks at different abstraction levels, more detailed analysis of the functionality of each binary bit in different retrieval tasks, and comparison with more state-of-the-art methods. **Fourth**, we extend the implementation details of our method, including the exact process of different retrieval tasks, to make it easier for re-implementation.

The rest of the paper is organised as follows: Sect. 2 discusses the works related to our method. Section 3 describes our DPH method in detail. Section 4 extensively evaluates the proposed method on two large-scale datasets, followed by conclusions in Sect. 5.

2 Related Works

In this paper, we aim at learning multifunctional binary codes for multiple realistic image retrieval tasks. Therefore, our work is naturally related to the multi-task learning problem. Specifically, some previous works, e.g. (Sun et al. 2014; Liu et al. 2016b; Kokkinos 2017; He et al. 2017; Zamir et al. 2018; Cao et al. 2018a), have adopted CNN models to simultaneously deal with multiple different tasks, and have achieved some successes. The benefits of such multi-task learning methods are twofold: First, by making use of the relations between tasks, one might be able to improve the performance of one or more task(s). Second, by sharing the feature extraction pipeline of multiple tasks, such multi-task learning methods consume less computation and memory than using one model for each individual task, and thus have their advantages in resource-limited situations, e.g. on smartphones or self-driving cars. Considering that large-scale image retrieval tasks have very strict requirements on time and storage, we borrow the idea from multi-task learning methods to learn compact binary codes for simultaneously dealing with multiple realistic retrieval tasks, which can significantly reduce the number of models and thus decrease retrieval time. Compared with the above methods, the key difference in representations (real-valued vs. binary) enables our method to better satisfy the strict time and storage requirements of large-scale image retrieval tasks.

In the past few years, hashing (Gionis et al. 1999; Kulis and Darrell 2009; Wang et al. 2012; Gong and Lazebnik 2011; Norouzi and Fleet 2011; Liu et al. 2012; Xia et al. 2014; Lai et al. 2015; Liu et al. 2016a; Zhang et al. 2016; Cao et al. 2017; Yang et al. 2015) has been widely studied for large-scale retrieval tasks for its low time and space complexity. The pioneering data-independent hashing method Locality Sensitive Hashing (LSH) (Gionis et al. 1999) uses random projections to produce binary bits, and thus LSH usually requires long codes to achieve satisfactory retrieval perfor-

mance. To reduce the storage cost, data-dependent hashing methods are proposed to learn more compact binary codes by utilizing a given training set. Such methods can be further divided into unsupervised and (semi-)supervised. Unsupervised methods, e.g. Spectral Hashing (SH) (Weiss et al. 2008) and Iterative Quantization (ITQ) (Gong and Lazebnik 2011), only make use of unlabelled training data to learn hash functions, while supervised methods are proposed to deal with the more complicated semantic similarities by taking advantage of semantic labels. Some representative supervised hashing methods are CCA-ITQ (Gong and Lazebnik 2011), Minimal Loss Hashing (MLH) (Norouzi and Fleet 2011), Semi-Supervised Hashing (SSH) (Wang et al. 2012), Binary Reconstructive Embedding (BRE) (Kulis and Darrell 2009), and Supervised Hashing with Kernels (KSH) (Liu et al. 2012). Although the aforementioned hashing methods have achieved successes in some applications, they use the readily extracted features, which are not specifically designed for the task at hand, and thus might lose some task-specific information. To tackle this issue, most recently, several hashing methods (Zhao et al. 2015; Lai et al. 2015; Xia et al. 2014; Zhang et al. 2015; Liu et al. 2016a; Zhang et al. 2016; Cao et al. 2017; Yang et al. 2015; Cao et al. 2018b, c; Jiang et al. 2018; Cakir et al. 2018) significantly improve the state of the arts by jointly learning the image representations and the hash functions using CNN models.

While the above supervised hashing methods mainly aim at category-oriented retrieval task, attributes have also been widely adopted in image retrieval (Sadovnik et al. 2013; Kovashka and Grauman 2013; Scheirer et al. 2012; Kumar et al. 2008; Tao et al. 2015; Turakhia and Parikh 2013; Rastegari et al. 2013; Yu et al. 2012; Siddiquie et al. 2011; Hu et al. 2016; Long et al. 2018). Our work is most related to the works that use nameable attributes (Parikh and Grauman 2011) as queries. Among these methods, (Kumar et al. 2008) predicts the probability of attributes with SVM classifiers, and uses the joint probabilities of query attributes to rank the database images. Follow-up works investigate the usage of attribute correlation (Siddiquie et al. 2011), fusion strategy (Scheirer et al. 2012; Rastegari et al. 2013), relative attributes (Sadovnik et al. 2013), natural language (Hu et al. 2016), dominate attributes (Long et al. 2018), and other techniques (Kovashka and Grauman 2013; Turakhia and Parikh 2013) to improve the retrieval performance. In this paper, we adopt the retrieval strategy in Kumar et al. (2008) for simplicity, while those more complicated ones (Siddiquie et al. 2011; Scheirer et al. 2012; Rastegari et al. 2013) are also compatible with our framework. A major issue of these attribute-oriented image retrieval methods is the usage of real-valued features, which limits the scalability and efficiency of such methods. Another line of research learns cross-modal binary codes to align samples of different modalities, e.g. images and texts (Jiang and Li 2017; Liu et al. 2015; Deng et al. 2018; Liu et al. 2014; Yang et al. 2017). Such methods are able to learn the correspondences between images and attributes. However, such methods usually treat the different modalities as a whole (i.e. all attributes form an attribute modality), and it is difficult to retrieve images based on a user-specified small part of the other modality (e.g. retrieve by a single attribute).

Some recent works (Rastegari et al. 2012; Liu et al. 2013; Li et al. 2015; Huang et al. 2016; Veit et al. 2017; Al-Halah et al. 2018) have made early attempts to learn image representations that are capable of dealing with multiple retrieval tasks. Rastegari et al. (2012) and Huang et al. (2016) discover attributes from learned binary codes by visualizing the images with the highest and lowest scores at each bit. This "post-processing" manner, however, hinders the method to learn the desired nameable attributes, thus making (Rastegari et al. 2012; Huang et al. 2016) unsuitable to be used for attribute-oriented retrieval tasks. Li et al. (2015) improves Rastegari et al. (2012) by explicitly modelling the connection between hash bits and attributes in the binary code learning stage. Nevertheless, the simple linear transformation based on the manually selected image representations in Li et al. (2015) is inadequate to capture the complex correlation between category and attributes. Veit et al. (2017) and Al-Halah et al. (2018) only learn real-valued representations that encode both types of information, which have high demands for both storage and computation cost. Moreover, the bit selection method proposed in Liu et al. (2013) could select the most useful hash bits for a selected retrieval task. However, this method could only select from the readily available hash bits, and there is no guarantee that the desired information is encoded in the available bits. As a result, these methods are suboptimal for dealing with large-scale image retrieval tasks. To address the shortcomings of previous works, we propose to exploit the CNN models to hierarchically capture the correlation between these two semantic descriptions, and encode them into compact binary codes in an end-to-end manner.

3 Approach

Our goal is to encode image information at different abstraction levels into unified binary codes, such that the learned binary codes could be adopted to achieve different retrieval tasks according to different users' expectations.

To achieve this goal, we present a Dual Purpose Hashing framework as illustrated in Fig. 3. The backbone network consists of multiple convolution-pooling layers, and optionally followed by several fully connected layers. The structure of these layers is very flexible, thus various successful models (Krizhevsky et al. 2012; Szegedy et al. 2015; He et al. 2016) can be adopted in our method. Since directly optimizing the discrete binary codes in CNN models is difficult, the penultimate layer in our network is designed to produce

binary-like outputs (a fully connected layer with $sigmoid$ activation) to approximate the actual binary codes. During the training stage, the whole network is jointly trained on a category-related task and an attribute-related one to encode both kinds of semantic information into binary codes. Moreover, the loss functions are specifically designed to make use of the abundant partially labelled data on the Internet, which can meanwhile improve the generalization ability of the models, as shown in Sect. 4.2.2. The detailed designs of each module will be introduced in the following subsections.

3.1 Problem Setup

Let Ω be the space of RGB images, we want to train an end-to-end model that maps images from Ω to k-bit binary codes $\mathscr{F} : \Omega \rightarrow \{0, 1\}^k$, where the binary codes could be used to perform multiple realistic retrieval tasks. In the following, we will use boldface uppercase letters (e.g. $\mathbf{A}$) and boldface lowercase letters (e.g. $\mathbf{a}$) to represent matrices and vectors respectively, where $\mathbf{A}_i$, $\mathbf{A}_{ij}$, and $\mathbf{a}_i$ denote the i-th row of $\mathbf{A}$, j-th element of $\mathbf{A}_i$, and i-th element of $\mathbf{a}$. Suppose that there are C disjoint categories and p visual attributes defined on the images. Let $S_{tr} = \{(X_i^{tr}, y_i, \mathbf{A}_i)|i = 1, \cdots, N\}$ denote the training set consisting of N images, where $X_i^{tr} \in \Omega$, $y_i \in \{1, \cdots, C, C + 1\}$ denotes the category of the i-th image, and $\mathbf{A}_i \in \{0, 1, 2\}^{1\times p}$ indicates the p corresponding visual attributes. More concretely, $y_i = C + 1$ means the category label of the i-th image is unknown, $\mathbf{A}_{ij} = 1$ and 0 indicates the j-th attribute is present/absent in the i-th image respectively, and $\mathbf{A}_{ij} = 2$ denotes that the j-th attribute label of the i-th image is not given. To avoid uninformative samples, we only consider the case that at least one label (either the category or one attribute) is given for each training image. For the i-th training image X_i^{tr}, we use $\mathbf{B}_i \in \{0, 1\}^k$ to denote the corresponding k-bit binary codes and $\mathbf{B}_i^r \in [0, 1]^k$ is the binary-like counterpart before quantization, where $\mathbf{B}_i = 0.5 * sign(\mathbf{B}_i^r - 0.5) + 0.5$. In practice, the $sign$ function is non-differentiable, which makes it intractable to optimize the CNN model via back propagation method. To deal with this problem, we use $\mathbf{B}_i^r$ in the training stage as an alternative. Moreover, assume that the d-dimensional feature representation of X_i^{tr} extracted by the CNN model is $\phi(X_i^{tr}) \in \mathbb{R}^d$, the binary-like codes $\mathbf{B}_i^r$ is obtained by:

$$\mathbf{B}_i^r = \sigma(\mathbf{W}^{hash}\phi(X_i^{tr}) + \mathbf{b}^{hash}), \tag{1}$$

where $\mathbf{W}^{hash} \in \mathbb{R}^{k\times d}$ and $\mathbf{b}^{hash} \in \mathbb{R}^k$ denote the weight matrix and bias terms that transform the feature representations to the binary-like codes, and $\sigma(\cdot)$ denotes the sigmoid function.

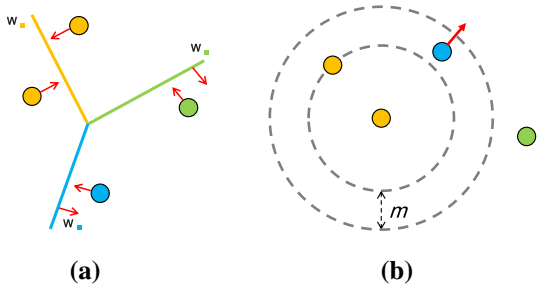


Fig. 4 Illustration of the two kinds of loss functions for encoding category information. **a** The softmax loss, where image samples are required to be close to their corresponding classifier weight vector in terms of inner product. **b** Triplet ranking loss, where the distance between similar samples (yellow circles) should be smaller than the distances between dissimilar samples (the center yellow circle, the blue circle, and the green circle) by a margin m (Color figure online)

3.2 Category Information Encoding

Existing mainstream hashing methods preserve category similarity mainly by two kinds of approaches, i.e. classification and metric learning. In our method, both choices are considered and will be illustrated in the following subsection.

3.2.1 Classification Based Encoding

As a commonly adopted approach (Shen et al. 2015; Yang et al. 2015), the basic idea of classification-based encoding is that if a simple transformation (e.g. softmax classifier or linear regression, as shown in Fig. 4a) can recover the category label from the binary codes, the category information would have been encoded into the binary codes. In our method, we borrow the above idea from previous works and use classification as a means of encoding category information into binary codes. Note that in our situation, the category labels of some training images might be missing, to avoid the risk of misclassification of such images, we choose to simply ignore them in the classification task. Thus we define the classification loss of a single training image X_i^{tr} based on the widely adopted softmax classification loss as follows:

$$L_i^{cls} = -\sum_{c=1}^{C} \mathbb{I}\{y_i = c\} log \frac{exp(\mathbf{W}_c^{cls}\mathbf{B}_i^r + \mathbf{b}_c^{cls})}{\sum_{l=1}^{C} exp(\mathbf{W}_l^{cls}\mathbf{B}_i^r + \mathbf{b}_l^{cls})}, \tag{2}$$

where the superscript cls indicates classification, $\mathbb{I}\{cond.\}$ is 1 when $cond.$ is true and 0 otherwise, $\mathbf{W}^{cls} \in \mathbb{R}^{C\times k}$ and $\mathbf{b}^{cls} \in \mathbb{R}^C$ denote the projection matrix and bias terms of the softmax classifier respectively. For the case when $y_i = C+1$, namely, the category label of the i-th image is missing, for all $c \in \{1, \cdots, C\}$ we have $\mathbb{I}\{y_i = c\} = 0$, thus the loss and gradient are both zeros, and those images without category labels are naturally ignored in the classification loss.

3.2.2 Metric Learning Based Encoding

Another widely studied approach for encoding category information is to project the images to a feature space, in which similar images are close to each other while dissimilar images are far away (Lai et al. 2015; Liu et al. 2016a; Cao et al. 2017). Specifically, images with same known category labels are considered as similar and vice versa (i.e. images with $y_i = C + 1$ are ignored similarly as in Sect. 3.2.1). For the i-th training image, a feasible way of defining the metric learning loss based on image triplets (as shown in Fig. 4b) is as follows:

$$L_i^{ml} = \sum_{y_k \neq y_i} \sum_{y_j = y_i} max(D_H(\mathbf{B}_i, \mathbf{B}_j) + m - D_H(\mathbf{B}_i, \mathbf{B}_k), 0), \tag{3}$$

where the superscript ml stands for metric learning, m is a margin parameter to control the strictness of the loss function, and $D_H(\cdot, \cdot)$ denotes the Hamming distance between the corresponding binary codes. As discussed in Sect. 3.1, the discrete $\mathbf{B}_i$ and Hamming distance could raise a lot of difficulty in optimization. Therefore, we relax Eq. (4) as follows:

$$L_i^{mlr} = \sum_{y_k \neq y_i} \sum_{y_j = y_i} max(D_E(\mathbf{B}_i^r, \mathbf{B}_j^r) + m - D_E(\mathbf{B}_i^r, \mathbf{B}_k^r), 0), \tag{4}$$

where the superscript mlr denotes metric learning relaxed, and $D_E(\cdot, \cdot)$ denotes the Euclidean distance between samples.

Note that here we adopt the triplet rank loss as in Lai et al. (2015) for preserving the category similarity. Besides, other metric learning based loss functions (e.g. Liu et al. 2016a; Cao et al. 2017) are also compatible with our framework, and can be simply implemented by replacing the current loss function with the aforementioned ones.

3.3 Attribute Encoding

To preserve attribute similarity, we adopt the similar idea to Sect. 3.2.1, i.e. the attributes of images are encoded into the binary codes by applying a transformation that can recover the visual attributes from binary codes. Since the attributes are binary in this work, for each of the p attributes, we define the loss as a binary classification task. Taking into account the imbalanced distribution of positive and negative samples on each attribute, we consider two choices of the loss function as described in the following, while other multi-label classification losses are also compatible with our framework.

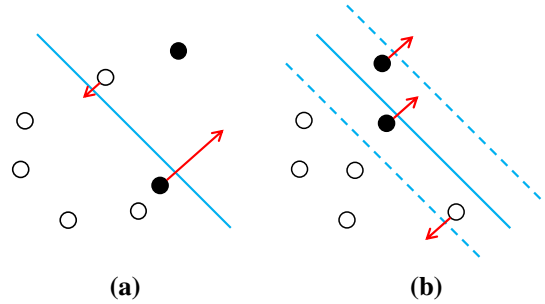


Fig. 5 Illustration of the two kinds of loss functions for encoding attribute information. **a** The weighted sigmoid cross entropy loss, where misclassified samples of the smaller category (black circles) will be severely punished, while misclassified samples of the larger category (white circles) will only be slightly punished. **b** Hinge loss, where only the samples that are not well-classified by the current model will be punished (the rightmost white circle and the black circles)

3.3.1 Weighted Sigmoid Cross Entropy Loss

One choice is to formulate the binary classification task as a binary logistic regression problem. To handle the missing label case, the standard formulation of logistic regression is modified to suit in our problem. Specifically, for $\mathbf{A}_{ij} \neq 2$, the j-th ($j \in \{1, 2, \cdots, p\}$) attribute prediction loss of a single training image X_i^{tr} is defined as a modified cross entropy loss:

$$L_{ij}^{sce} = -\mathbf{A}_{ij} log(\sigma(\mathbf{W}_j^{attr} \mathbf{B}_i^r + \mathbf{b}_j^{attr})) + (1 - \mathbf{A}_{ij}) log(1 - \sigma(\mathbf{W}_j^{attr} \mathbf{B}_i^r + \mathbf{b}_j^{attr})), \tag{5}$$

where the superscript sce denotes sigmoid cross entropy, $\mathbf{W}^{attr} \in \mathbb{R}^{p \times k}$ is the weights of the attribute predictor, and $\mathbf{b}^{attr} \in \mathbb{R}^p$ are the corresponding bias terms. While for $\mathbf{A}_{ij} = 2$, we simply define $L_{ij}^{sce} = 0$ to ignore such unlabelled samples.

In practice, directly optimizing Eq. (5) would lead to biased solution, since the distribution of some attributes are highly imbalanced (i.e. for some attributes, only a tiny portion of images have/do not have these attributes), even predicting all images as negative/positive would result in a relatively small loss. To alleviate the impact of sample imbalance, for $\mathbf{A}_{ij} \neq 2$, we propose a weighted version of Eq. (5) instead (as shown in Fig. 5a):

$$L_{ij}^{wsce} = -w_j^{(p)} \mathbf{A}_{ij} log(\sigma(\mathbf{W}_j^{attr} \mathbf{B}_i^r + \mathbf{b}_j^{attr})) + w_j^{(n)} (1 - \mathbf{A}_{ij}) log(1 - \sigma(\mathbf{W}_j^{attr} \mathbf{B}_i^r + \mathbf{b}_j^{attr})), \tag{6}$$

where the superscript $wsce$ stands for weighted sigmoid cross entropy, and $w_j^{(p)}$ and $w_j^{(n)}$ are used to balance the loss terms on positive and negative training samples. Specifically, the weights $w_j^{(p)} = \frac{N_j^{(n)}}{N_j^{(n)} + N_j^{(p)}}$ and $w_j^{(n)} = \frac{N_j^{(p)}}{N_j^{(n)} + N_j^{(p)}}$,

where $N_j^{(p)}$ and $N_j^{(n)}$ are the numbers of samples that have $\mathbf{A}_{ij} = 1$ and $\mathbf{A}_{ij} = 0$ respectively.

3.3.2 Hinge Loss

Another popular choice for dealing with the imbalanced binary classification task is the hinge loss, which only focuses on the samples that have not been correctly classified with a large margin, and thus can somehow avoid predicting all samples as a single category (as shown in Fig. 5b). For $\mathbf{A}_{ij} \neq 2$, the corresponding hinge loss of the i-th sample on the j-th attribute is defined as follows:

$$L_{ij}^{hinge} = max(1 - (2 * \mathbf{A}_{ij} - 1)(\mathbf{W}_j^{attr}\mathbf{B}_i^r + \mathbf{b}_j^{attr}), 0), \quad (7)$$

where the superscript $hinge$ stands for hinge loss. Without loss of generality, assume that an attribute has much more positive samples than negative ones. If the model predicts all samples as positive, a certain percentage of positive samples would have $L_{ij}^{hinge} = 0$ and all the negative samples would have large losses, which reduces the number of effective positive samples and thus balances the positive and negative samples.

3.4 Joint Optimization

With the loss functions defined above, the CNN model can be trained with standard back propagation algorithm with mini-batches. However, directly adding up the category loss [Eqs. (2) or (4)] and attribute loss [Eqs. (6) or (7)] as the overall loss function could be problematic. To be specific, the values of the two types of losses might be in different orders of magnitudes. Moreover, due to missing labels, the loss corresponding to different attributes might also be in different orders of magnitudes. As a consequence, some parts of the loss might dominate and thus prevent the others from functioning. To tackle this problem, different parts of the loss function need to be scaled before added up. Suppose that in each iteration, the mini-batch consists of n images, the overall loss function on a mini-batch is defined as follows:

$$L = \frac{\sum_{i=1}^n L_i^{cls}}{\sum_{t=1}^n \mathbb{I}\{y_t \leq C\}} + \alpha \sum_{j=1}^p \sum_{i=1}^n \frac{L_{ij}^{wsce}}{\sum_{t=1}^n \mathbb{I}\{\mathbf{A}_{tj} \neq 2\}}, \quad (8)$$

where α is a weighting parameter to control the relative strength of the two loss terms. To train the model efficiently, each training mini-batch is carefully sampled to make sure that $\sum_{t=1}^n \mathbb{I}\{y_t \leq C\} > 0$ and $\sum_{t=1}^n \mathbb{I}\{\mathbf{A}_{tj} \neq 2\} > 0$, which also helps avoid numerical issues. By substituting L_i^{cls} and L_{ij}^{wsce} with L_i^{mlr} and L_{ij}^{hinge}, we can obtain four combinations of the overall loss function, which will be evaluated in Sect. 4. The gradients of Eq. (8) with respect to $\mathbf{B}_i^r$ and the model parameters ($\mathbf{W}^{cls}$, $\mathbf{b}^{cls}$, $\mathbf{W}^{attr}$, $\mathbf{b}^{attr}$, $\mathbf{W}^{hash}$, $\mathbf{b}^{hash}$, and the parameters in $\phi(\cdot)$) can be easily computed using standard derivation techniques, thus we do not bother to discuss them in detail.

3.5 Retrieval

After the training stage is over, the binary codes of images can be similarly obtained as discussed in Sect. 3.1. For each image in the database, its corresponding binary codes are stored for further usage. In this subsection, we describe the detailed retrieval process of the three retrieval tasks introduced in Sect. 1.

Category retrieval: The goal of category retrieval task is to search for database images with the same category label as the query image. To achieve this goal, we first compute the binary codes of the query image and the Hamming distances between the query codes and the database codes. After that, we rank the Hamming distances in ascending order and feedback the database images in the obtained order as retrieval results.

Attribute retrieval: Attribute retrieval aims at retrieving database images that match with the query image at some user-selected visual attributes. For this task, we need to recover the attribute information of database images from their corresponding binary codes, which can be accomplished by $\mathbf{A} = \sigma(\mathbf{W}^{attr}\mathbf{B} + \mathbf{b}^{attr})$. Note that thanks to the binary nature of $\mathbf{B}$, the matrix multiplications is equivalent to simply selecting and adding up some entries of $\mathbf{W}^{attr}$ according to the indexes of non-zero entries in $\mathbf{B}$, which is very efficient in computation. Similarly, the attribute information of the query image $\mathbf{a}^q$ could also be computed in the same way. To search for the matching images, we rank the database images in descending order of $P(\mathbf{A}_{ij} = \mathbf{a}_j^q, \forall j \in U_s) = \prod_{j \in U_s}(\mathbb{I}\{\mathbf{a}_j^q > 0.5\}A_{ij} + \mathbb{I}\{\mathbf{a}_j^q \leq 0.5\}(1 - A_{ij}))$, where U_s is the set of attributes selected by the user. The above score denotes the joint matching probability of the query image and the i-th database image on the selected subset of attributes.

Combined retrieval: The goal of combined retrieval task is to search for database images with the same category label as the query image, while having some certain attributes specified by the user. To address this challenging task, we first recover the attribute information as described above, and filter out the database images that do not match with the specified attributes (with attribute probability smaller than 0.5). After that, the remaining database images are ranked in ascending order of Hamming distances similarly as in the category retrieval task.

Compared to existing hashing methods that can only deal with the category retrieval task, our method only takes a little bit more memory for storing the attribute prediction parameters $\mathbf{W}^{attr}$ and $\mathbf{b}^{attr}$. Besides, as discussed above, recovering attribute information from binary codes could be very effi-

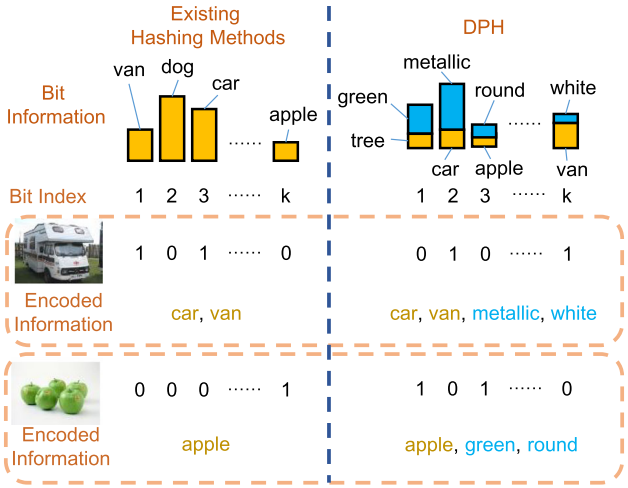


Fig. 6 An illustration of the bit functionality in existing hashing methods and our DPH method. Most existing hashing methods only encode category information (left of the figure), while our DPH method jointly encodes category and visual attribute information (right of the figure). In the top part of this figure, each rectangle represents a bit, where the areas of yellow and blue regions denote the amount of category information and attribute information carried by that bit respectively, and the words linked to the yellow and blue regions serve as an informal illustration of the information encoded by that bit. In the bottom part, two example images are provided to better explain the above idea (Color figure online)

ciently done. Therefore, our method is as efficient as existing hashing methods, while it can perform more retrieval tasks with only one model.

3.6 Analysing Bit Functionality

In most existing hashing methods, only category information is encoded into the binary codes, while in our method, both category and visual attribute information are encoded, as shown in Fig. 6. To give more insights into how the two types of information (i.e. category and attributes) are encoded into the binary codes in our method, we propose a novel method to analyse the contribution of each bit to the two aspects.

For the category information, our basic intuition is that if a certain bit has small within-class variations and large between-class variations, it is highly likely that this bit encodes a large amount of category information. Based on this intuition, for the k-th bit, we compute

$$\mathbf{I}_k^{cat} = -\sum_{y_i=y_j} (\mathbf{B}_{ik} - \mathbf{B}_{jk})^2 + \lambda \sum_{y_i \neq y_j} (\mathbf{B}_{ik} - \mathbf{B}_{jk})^2 \tag{9}$$

as an indicator of the category information amount carried by the corresponding bit, where $\lambda = \frac{\sum_{i,j} \mathbb{I}\{y_i=y_j\}}{\sum_{i,j} \mathbb{I}\{y_i \neq y_j\}}$ is adopted to balance the two terms.

On the other hand, for the attribute information, the absolute value of $\mathbf{I}_k^{attr,j} = \mathbf{W}_{jk}^{attr}$ can indicate the contribution of the k-th bit to the j-th attribute. Therefore, we use

$$\mathbf{I}_k^{attr} = \sum_{j=1}^{p} |\mathbf{W}_{jk}^{attr}| \tag{10}$$

as an indicator of the amount of attribute information carried by the k-th bit, where $|\cdot|$ denotes the absolute value operation.

In practice, the above two metrics might be in different orders of magnitudes, which makes it difficult to directly compare them. To deal with this problem, we use their corresponding maximums and minimums to normalize them into the range of [0, 1].

Moreover, to quantitatively evaluate the alignment between category and attributes, for the j-th attribute, we define

$$\mathbf{v}_j = -\sum_{y_s=y_t} (\mathbf{A}_{sj} - \mathbf{A}_{tj})^2 + \lambda \sum_{y_s \neq y_t} (\mathbf{A}_{sj} - \mathbf{A}_{tj})^2 \tag{11}$$

to denote the alignment between the j-th attribute and the categories, where λ is defined similarly as above. Specifically, the larger this value is, the more aligned the j-th attribute is with the categories. Similarly, $\mathbf{v}$ and $\mathbf{I}^{attr,j}$ are also normalized into the range of [0, 1] to keep in line with the above indicators. On the other hand, the Pearson's correlation coefficient $\mathbf{c}_j^{attr} = corr(\mathbf{I}^{attr,j}, \mathbf{I}^{cat})$ reflects the relationship between the j-th attribute's information and the category information encoded in the binary codes, the larger this value is, the more likely that these two types of information are encoded in the same bits. Naturally, we would expect that attributes with large $\mathbf{v}$ values would also have large $\mathbf{c}^{attr}$ values, which means that the stronger an attribute is aligned with the categories, the more likely that this attribute's information is encoded in the same bits as the category information. Experiments in the following section will show that DPH performs exactly as expected.

4 Experiments

In this section, we extensively evaluate our method on two large-scale datasets. First, we analyse the effect of individual modules of our framework. Then the proposed DPH method is compared with the state-of-the-art retrieval methods on each of the three tasks to validate the advantages of our method. Compared with the preliminary results reported in Liu et al. (2017), the comparative methods are very carefully tuned to produce better results, and all attribute-related experiments are re-evaluated using some more suitable metrics, which will be detailed in the following subsections.

4.1 Experimental Settings

Datasets: We evaluate our DPH method on two large-scale partially labelled datasets: (1) **ImageNet-150K** is a subset

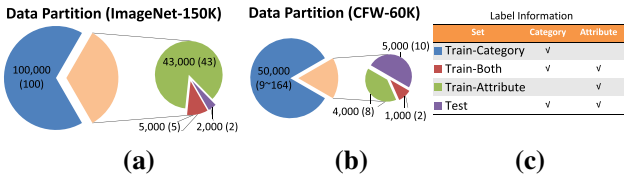


Fig. 7 Illustration of data partition in our experiments. **a** ImageNet-150K with 1000 categories, **b** CFW-60K with 500 categories. The sizes of each set are presented in the figure, and the numbers in the brackets indicate the number of images from each category. **c** The label information of the corresponding sets. Best viewed in color (Color figure online)

of ILSVRC 2012 dataset Russakovsky et al. (2015) with 150,000 images. For each of the 1000 categories, we select 148 images from the training set and 2 images from the validation set. After that, 48 out of the 148 selected training images for each category and all the 2000 selected validation images are manually annotated with 25 attributes (including color, texture, shape, material, and object parts). We partition the dataset into 4 parts (Train-Category, Train-Both, Train-Attribute, and Test) as illustrated in Fig. 7a. (2) **CFW-60K** (Li et al. 2015) is a subset of the CFW dataset (Zhang et al. 2012) and contains 60,000 images of 500 subjects, among which 20 images of each subject are annotated with 14 attributes. For the 10,000 images with attribute annotations, 10 images of each subject are used as Test set, and the rest are further divided into two parts (Train-Both and Train-Attribute) similarly as ImageNet-150K. The details of partitioning is illustrated in Fig. 7b. Please refer to the original publications (Zhang et al. 2012; Li et al. 2015) for more details about this dataset. On both datasets, the category labels of the Train-Attribute set are made unavailable in the training stage to serve as partially labelled data.

Evaluation protocol: All the evaluations are carried out solely on the Test set in a **leave-one-out** manner, namely, each time we select one image from the Test set as query image, and the rest as database. We report the average retrieval performance of all Test set images. Since the three retrieval tasks are very different from each other, we use different evaluation metrics for these tasks.

Category retrieval: we use the standard mean Average Precision (mAP) of retrieval as the metric, where database images with the same category label as the query image are considered as relevant.

Attribute retrieval: we consider the case when the user selects at most three attributes in retrieval. For each possible one-/two-/three-attribute selection, we compute the average retrieval mAP over all valid attribute combinations (e.g. a possible selection of two attributes is "red + blue", and the valid case combinations over these two attributes are "red and blue", "red and not blue", "not red and blue", and "not red and not blue"), and we use the average results on all possible selections to measure the overall retrieval performance. In this experiment, database images that match with the query image at all selected attributes are considered as groundtruth matches. Note that we use the predicted attributes of all images (both query and database) for ranking as described in Sect. 3.5, while evaluate by the groundtruth annotations. As a result, both wrong predictions of the query image and the database images would result in poor performance.

Combined retrieval: considering that the combined retrieval task is much more difficult than the other two tasks, we use the less strict $recall@k$ metric to measure the performance. For each attribute, we compute the mean $recall@k$ metric on all valid query images (images that have at least one match in the database), and use the average on all attributes to measure the retrieval performance of the combined retrieval task.

Implementation details: Although we have exploited partially labelled data in the training stage, our datasets are still relatively small in terms of training a deep CNN model from scratch. In consideration of generalization ability, the model parameters are initialized using pre-trained CNN models. Specifically, for ImageNet-150K, we use the publicly available AlexNet (Krizhevsky et al. 2012) model provided in the model zoo of Caffe (Jia et al. 2014) unless otherwise specified. The pre-trained model parameters from the conv1 layer to the fc7 layer are used to initialize our models. For CFW-60K, we adopt the CNN structure of Yi et al. (2014) (from conv1 to pool5). Since the pre-trained model is not publicly available on CFW-60K, we follow the original publication (Yi et al. 2014) to train the model, except for removing the contrastive loss for simplicity. For fair comparison, all comparative methods use the same pre-trained model as our method.

For both datasets, the model is trained for 150,000 iterations. We set the initial learning rate to 10^{-3} for the pre-trained layers, and 10^{-2} for the newly added layers. The learning rate is multiplied by 0.1 for every 60,000 iterations. To train the model, we use a mini-batch size of 128, where 64 images are sampled from the Train-Attribute subset and the other 64 images are from the rest parts of the training set using the class-aware sampling strategy in Shen et al. (2016). Namely, the 64 images with category labels are uniformly sampled from 16 distinct categories to ensure the balance between different categories as well as enough valid triplets for the metric learning loss. Moreover, the margin parameter m for the metric learning loss is empirically set to 4. The momentum and weight decay parameters are set according to the original publications (Krizhevsky et al. 2012; Yi et al. 2014). Besides, we first set $\alpha = 0$, and compute the category retrieval mAP of this model. After that, the value of α is gradually increased, until the category retrieval mAP significantly drops. Finally, we select the largest α value with acceptable category retrieval mAP in the experiments, i.e. $\alpha = 0.1$. All

Table 1 The retrieval performance of models trained with different combinations of loss functions. Cat., Attr., and Com. represent category, attribute, and combined retrieval tasks respectively

Loss	ImageNet-150K			CFW-60K		
	Cat.	Attr.	Com.	Cat.	Attr.	Com.
cls + wsce	0.353	0.834	0.732	0.439	0.991	0.384
cls + hinge	0.323	0.762	0.491	0.347	0.987	0.296
ml + wsce	0.322	0.816	0.672	0.402	0.989	0.328
ml + hinge	0.326	0.738	0.470	0.405	0.990	0.311

The results are obtained with 256-bit binary codes. For the category and attribute retrieval tasks, the reported results are mAP, and for the combined retrieval task, the performance metric is recall@5

the comparison CNN methods are implemented with Caffe (Jia et al. 2014).[1]

4.2 Module Analysis

In this section, we conduct extensive experiments to validate the effectiveness of the individual modules of our framework, and give insights into the working mechanism of encoding multiple types of information.

4.2.1 Evaluation of Different Combination of Loss Functions

We have described two feasible loss functions for encoding both types of information. In this part, we thoroughly evaluate all four combinations of these loss functions. For this purpose, we train hashing models that produce 256-bit binary codes using different loss combinations, and report the performances of the corresponding models on all three retrieval tasks. Specifically, for the combined retrieval task, we use $recall@5$ as the performance metric in this experiment. The results are shown in Table 1. We can see that the combination of classification loss (for encoding category information) and weighted sigmoid cross entropy loss (for encoding attribute information) gives the best results. Although the two newly-introduced losses did not yield the highest accuracy, their performances are still competitive. Specifically, the performances of these models outperform some of the state-of-the-art methods in the following experiments. Therefore, we can confidently conclude that DPH is compatible with different loss functions, and it is quite feasible to improve the performance of DPH by designing/incorporating more well-suited loss functions and training schemes.

Compared with the classification loss, metric learning based loss functions are difficult to train. Even though we have explicitly designed the mini-batch sampling strategy to incorporate more valid image triplets, a large percentage of possibly beneficial triplets are not sampled, which results in suboptimal performance of the corresponding models on category-related tasks. On the other hand, by taking a closer look at the two attribute-oriented loss functions, we have found that the hinge loss tends to predict most samples as the larger class even though the loss function is designed to ignore some of the well-classified samples, and thus the hinge loss results in inferior performance. Based on the above results, we will use classification loss with weighted sigmoid cross entropy loss in the following experiments.

In our current framework, there are no explicit constraints on the quantization loss of the binary-like codes. In this subsection, we also conduct experiments to show the impact of quantization loss on DPH. For this purpose, two sets of experiments are conducted on ImageNet-150K and CFW-60K with 256-bit binary codes: (1) to show the impact of quantization loss on predicting the attributes, we use the real-valued binary-like codes instead of the binary ones to predict the attributes. The corresponding performances of attribute retrieval and combined retrieval tasks on both datasets are evaluated. (2) To show the impact of quantization loss on the binary codes, we evaluate the performances of binary-like codes on the category retrieval task on ImageNet-150K and CFW-60K datasets with 256-D real-valued codes, and compare the performances with the real binary codes. Specifically, we replace the Hamming distance with Euclidean distance to characterize the similarity between samples.

The results are shown in Table 2. Moreover, we also provide the distribution of the real-valued binary-like codes (all bits) on the Test set of both datasets in Fig. 8. We observe from Table 2 that the quantization loss only has marginal impact on the retrieval performances, validating that DPH can achieve satisfactory performance without explicit quantization loss. In fact, as shown in Fig. 8, the real-valued binary-like codes have small quantization losses even without explicit constraints. A possible explanation is that the sigmoid activation function has large gradients in the linear region and small gradients in the saturation regions. As a result, samples that fall in the linear region are likely to be pushed to the saturation region. On the contrary, once a sample falls in the saturation region, it is difficult to be pushed back to the linear region due to the small gradient values. Therefore, as the training goes on, most samples fall in the saturation region and have small quantization losses.

4.2.2 Evaluation of Partially Labelled Data

In this part, we evaluate the impact of utilizing partially labelled data on both datasets by testing on 256-bit binary codes. For this purpose, 6 models are trained with different training sets: we name these models as Both (**B**), Both + Attribute (**B + A**), Both + Category (**B + C**), and Both + Attribute + Category (**B + A + C**) according to the training

[1] The source code of DPH and the ImageNet-150K dataset are available at "http://vipl.ict.ac.cn/resources/codes".

Table 2 The impact of quantization loss on DPH

Task	ImageNet-150K		CFW-60K	
	Before	After	Before	After
Category Retrieval	0.353	0.355	0.457	0.439
Attribute Retrieval	0.814	0.814	0.991	0.991
Combined Retrieval	0.713	0.713	0.384	0.382

The performances on the three retrieval tasks before and after quantization are given in the table. For the category and attribute retrieval tasks, the reported results are mAP, and for the combined retrieval task, the performance metric is recall@5

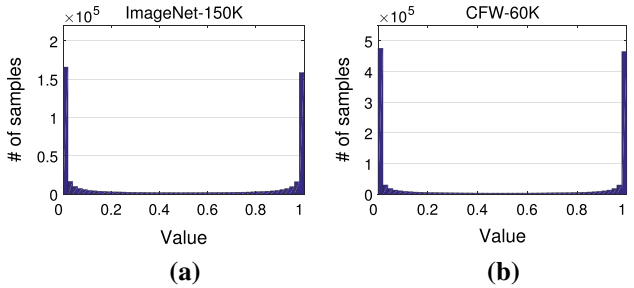


Fig. 8 Distribution of the 256-bit real-valued binary-like codes (all bits) on the test set of **a** ImageNet-150K and **b** CFW-60K datasets

sets (please refer to Sect. 4.1 and Fig. 7 for details) used to train the specific model. Moreover, we also conduct experiments of training the DPH model with varying ratios of partially labelled data to show the impact of training sample number. Specifically, we randomly select 50% images from the Train-Category subset to form an additional training subset, which is denoted as 0.5C. Similarly, a 0.5A subset is randomly selected from the Train-Attribute subset. Based on the additional training subsets, two DPH models are trained with **B + 0.5C** and **B + 0.5A** respectively. In this subsection, the encoding of category and attributes are evaluated separately. For the category part, we use the retrieval mAP as a measurement. For the attribute part, we adopt a more straightforward metric to denote the attribute encoding performance, i.e. the average harmonic mean of TPR and TNR over all attributes, where TPR and TNR denote true positive rate and true negative rate respectively. Note that since some attributes are highly unbalanced, e.g. more than 98% images do not possess the attribute "orange" in ImageNet-150K, this metric can more faithfully reflect the real attribute prediction performance than the flat sample-wise accuracy.

The comparison results are given in Table 3. We can infer that: **First**, compared with the "Both" model, exploiting extra training data (B + A and B + C) significantly improves the performance of the corresponding task. This observation can be explained by model overfitting, to be specific, in our experiments, in the training stage of the "Both" model, the training loss approaches zero while the test loss only decreases slightly. In contrast, when additional training data

Table 3 Comparison of the 256-bit models trained with different combinations of training data

	ImageNet-150K		CFW-60K	
	mAP	average HM	mAP	average HM
B	0.268	0.825	0.180	0.858
B + 0.5A	0.266	0.881	0.177	0.889
B + A	0.264	0.878	0.179	0.899
B + 0.5C	0.322	0.850	0.369	0.856
B + C	0.363	0.848	0.445	0.870
B + A + C	0.355	0.886	0.439	0.911

The mAP of the category retrieval task and the average harmonic mean of TPR and TNR over all attributes (average HM) are shown in the table. B: Both, A: Attribute, C: Category, where 0.5A and 0.5C are 50% samples randomly selected from A and C respectively

is introduced, the training loss and test loss of the corresponding tasks are always on the same scale as normally expected. This justifies our motivation of using partially labelled data in training the CNN models to alleviate overfitting. **Second**, models trained with fewer partially labelled data (B + 0.5A and B + 0.5C) achieve better performance than the "B" models on the corresponding tasks, yet they generally underperform the "B + C" models and have similar performances with the "B + A" models. One possible explanation is that as a mid-level image representation, the attribute information is less abstract and is easier to learn than the category information, and thus the model could learn to well encode the attribute information from a smaller amount of data. The above results also suggest that DPH can efficiently make use of the partially labelled data in the training stage to encode the two kinds of information. **Third**, compared with training solely on "Train-Both" set, using all training data can improve the performance on both tasks by a large margin (the row "B + A + C" in Table 3), and the performance of this dual-purpose model is comparable with or even better than the performances of the "B + A" and "B + C" models on the corresponding tasks, confirming that it is feasible to simultaneously embed category and visual attributes into the binary codes by exploiting partially labelled data. **Fourth**, using more images with attribute annotations in the training stage slightly degrades the category retrieval performances ("B + A + C" vs. "B + C"). A possible explanation is that encoding the category information requires suppressing the within-class variation (including the attribute variation), and vice versa. As a result, the two types of information are actually competing for the limited information capacity of binary codes. However, the additional attribute data and task endow the DPH method with the ability of attribute-oriented retrieval, which is not possessed by existing hashing methods, at the cost of only a marginal drop in category retrieval

task. In the following experiments, all our models are trained with the "B + A + C" setting.

4.2.3 Evaluation of More Advanced Backbone Architecture

Many recent works have shown that different layers in CNN models extract semantic information at different abstraction levels (Escorcia et al. 2015; Zhong et al. 2016; Bau et al. 2017). For example, the lower layers usually capture textures or object parts, while the higher layers characterise task-specific features. Considering that our framework aims at encoding both high- and low-level semantics (category and attributes), it would be beneficial to make use of image features from multiple different CNN layers. To this end, inspired by the skip-layer connections proposed by He et al. (2016), which naturally enables information flow between different CNN layers, we replace the backbone network with the 18-layer ResNet model and evaluate the retrieval performance accordingly. Specifically, the last fully connected layer of the pre-trained ResNet-18 model (pre-trained on the ILSVRC 2012 (Russakovsky et al. 2015) object recognition task) is replaced with the dual purpose hashing module as described in Sect. 3. The model is trained for 50,000 iterations with the same initial learning rate as described above, and the learning rate is multiplied by 0.1 for every 20,000 iterations. Moreover, the weighting parameter α is set identically to the above AlexNet model ($\alpha = 0.1$) for fair comparison. Since the ResNet-18 model is trained for general object classification instead of face recognition task, we only compare the results on ImageNet-150K and use 256-bit binary codes for evaluation.

Figure 9 shows the comparison results of the two backbone networks on the three retrieval tasks. We can see that the ResNet-18 counterpart achieves much higher retrieval performances on the category retrieval and combined retrieval tasks, while the AlexNet-based model performs better on the attribute retrieval task. A possible explanation is that the ResNet-18 model achieves better object classification accuracy than AlexNet, which suggests that ResNet-18 can better suppress the within-class variations (including attributes) than AlexNet. As a result, the feature representations extracted by ResNet contains less attribute information, and thus performs inferior to AlexNet on the attribute retrieval task while outperforms the AlexNet-based model on the other two category-related tasks. Considering that the number of images with attribute labels is very small compared to the ILSVRC 2012 training set, it is difficult to change the knowledge that has already been learned in the pre-training stage with such a small number of attribute labels. However, if we have more images with attribute labels, it is likely that we can obtain a ResNet-18 model that outperforms the AlexNet-based one on all three tasks, yet it is beyond the scope of this paper.

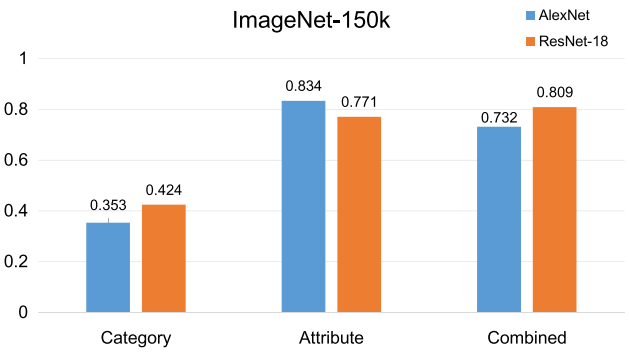


Fig. 9 Comparison of our proposed DPH method with AlexNet and ResNet-18 as the backbone network respectively. The retrieval mAP of category retrieval and attribute retrieval tasks as well as the $recall@5$ metric of the combined retrieval tasks are reported

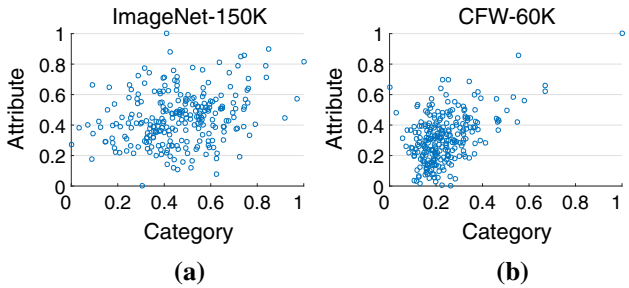


Fig. 10 Analysis of the functionality of each individual bit, where the x axis and y axis correspond to the normalized category and attribute information metrics respectively. Each bit is represented by a circle in the graph. **a** Results on ImageNet-150K. **b** Results on CFW-60K

4.2.4 Analysis of Bit Functionality

In this part, we analyse the functionality of each individual bit to give more insights into the working mechanism of our method. For this purpose, we use the method described in Sect. 3.6 with 256-bit binary codes to quantitatively analyse the amount of task-specific information carried by each bit on both datasets.

The results on two datasets are shown in Fig. 10, and the correlation coefficients between the two types of information are 0.2661 and 0.4756 on the two datasets respectively. Such results suggest that the two types of information are somehow correlated. Namely, the amount of attribute information carried by a certain bit is more or less dependent on the amount of category information carried by that bit. Note that on the CFW-60K dataset, there are more attributes that are consistent within each class (e.g. gender and race), and thus the two types of information have stronger correlation on that dataset.

Figure 11 shows the alignment between attributes and category $\mathbf{v}$, against the correlation coefficients between attribute information and category information $\mathbf{c}^{attr}$ as described in Sect. 3.6. We observe from the figure that these two indicators

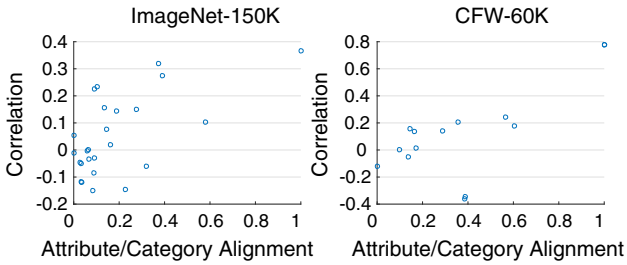


Fig. 11 The alignment between attributes and category, against the correlation coefficients between attribute information and category information. Each circle corresponds to an attribute

are strongly correlated with Pearson's coefficients 0.6079 and 0.7412 on the two datasets respectively. Such result suggests that the stronger an attribute is aligned with the categories, the more likely that this attribute's information is encoded in the same bits as the category information.

4.3 Comparison with State of the Arts

In this part, we compare our proposed method with state-of-the-art methods on the three retrieval tasks.

4.3.1 Evaluation of Category Retrieval

First, we test the effectiveness of our DPH method on the category retrieval task. Apart from the object dataset and the human face dataset (ImageNet-150K and CFW-60K), in this experiment, we also evaluate all comparative hashing methods on one more dataset to better validate the effectiveness of the proposed DPH method. Specifically, we select a different scenario (i.e. scenes) and combine the SUN397 dataset (Xiao et al. 2010) with the SUN Attribute dataset (Patterson et al. 2014) for experiments, which is denoted as SUN-120K. The SUN397 dataset has 108,754 images of 397 distinct scene categories, and there are 14,140 images of 717 scene categories in the SUN Attribute dataset (20 images per category, where each image is labelled with 102 visual attributes). After removing the duplicate images, there are about 120K images left in the combined dataset. Among the two datasets, there are 395 overlapped scene categories. We randomly select 10 images from each of the 395 categories to test the model performances (3,950 images in total).

We compare with nine hashing methods: LSH (Gionis et al. 1999), ITQ (Gong and Lazebnik 2011), CCA-ITQ (Gong and Lazebnik 2011), DBC (Rastegari et al. 2012), KSH (Liu et al. 2012), SDH (Shen et al. 2015), DNNH (Lai et al. 2015), HashNet (Cao et al. 2017), and SSDH (Yang et al. 2015), including representative conventional methods as well as state-of-the-art CNN-based methods.

For fair comparison, the conventional methods are trained using L2-normalized CNN features extracted from the pre-trained models (described in Sect. 4.1). Specifically, on SUN-120K dataset, the 4,096-D features extracted from the penultimate fully-connected layer of the CNN model in Wang et al. (2015) are used as feature inputs, and the above CNN model is also used to initialize the model parameters for deep hashing methods. The comparative methods are implemented using the source codes provided by the original authors. Specifically, for the CNN-based methods, i.e. DNNH, HashNet, and SSDH, we adopt the same backbone network as our DPH method, and initialize the model parameters with the identical pre-trained models as ours.

All the comparative methods are trained using the combination of "Train-Both" and "Train-Category" sets (on SUN-120K dataset, all but the test images are used to train the models). Moreover, since KSH demands a large amount of memory to store the kernel matrix ($O(N^2)$, where N is the number of training images), we use 20,000 images randomly selected from the training set for this method, which has already consumed more than 16GB of memory in the training stage. All the hyper-parameters of the comparative methods are tuned carefully according to the original publications. The results on {32, 64, 128, 256}-bit binary codes are given.

The results are shown in Table 4. We can see that: **First**, when equipped with CNN features, the linear and non-linear conventional methods (e.g. CCA-ITQ vs. KSH) have similar performances. One possible explanation is that the CNN has mapped the images to a feature space where different categories are already linearly separable, thus non-linear methods such as KSH can hardly benefit from the non-linearity of kernel space. **Second**, CNN-based methods significantly improve over conventional methods on CFW-60K and SUN-120K, yet have marginal improvement on ImageNet-150K. Note that the pre-trained models on CFW-60K and SUN-120K are obtained from a different dataset (Webface (Yi et al. 2014) and MIT Places (Zhou et al. 2014) respectively), while on ImageNet-150K from the same one, validating the advantage of CNN-based methods in learning more suitable representations for the data at hand. **Third**, metric learning based deep hashing methods (DNNH and HashNet[2]) perform relatively worse than classification-based methods (SSDH and DPH). A possible explanation is that the metric learning based methods need abundant valid training image pairs/triplets for effective training, while the commonly adopted mini-batch sampling strategies (e.g. random sampling or the class-aware sampling in our method) cannot perfectly satisfy such requirements when the number of categories is large. Such sampling strategies could be a potential research direction for these methods, yet they are beyond

[2] With the source codes released by the original authors, dozens of HashNet models under different hyperparameter settings are trained and the best results among these models are reported.

Table 4 Comparison of category retrieval performance (mAP) of our method and other comparative hashing methods on ImageNet-150K, CFW-60K, and SUN-120K

	ImageNet-150K				CFW-60K				SUN-120K			
	32-bit	64-bit	128-bit	256-bit	32-bit	64-bit	128-bit	256-bit	32-bit	64-bit	128-bit	256-bit
LSH Gionis et al. (1999)	0.070	0.134	0.215	0.269	0.110	0.117	0.118	0.118	0.094	0.153	0.196	0.225
ITQ Gong and Lazebnik (2011)	0.167	0.235	0.284	0.310	0.058	0.079	0.112	0.135	0.195	0.225	0.245	0.256
CCA-ITQ Gong and Lazebnik (2011)	0.157	0.223	0.294	0.341	0.069	0.090	0.113	0.140	0.162	0.229	0.285	0.318
DBC Rastegari et al. (2012)	0.264	0.308	0.344	0.369	0.060	0.072	0.099	0.129	0.263	0.298	0.321	0.337
KSH Liu et al. (2012)	0.181	0.253	0.293	0.320	0.063	0.086	0.111	0.117	0.182	0.230	0.262	0.298
SDH Shen et al. (2015)	0.143	0.222	0.288	0.322	0.049	0.095	0.140	0.183	0.216	0.259	0.287	0.302
DNNH Lai et al. (2015)	0.147	0.213	0.267	0.298	0.163	0.259	0.348	0.402	0.145	0.192	0.227	0.250
HashNet Cao et al. (2017)	0.194	0.226	0.257	0.275	0.121	0.202	0.274	0.322	0.220	0.245	0.262	0.269
SSDH Yang et al. (2015)	0.263	0.310	0.339	0.357	0.262	0.343	0.409	0.449	0.381	0.394	0.394	0.372
DPH	0.274	0.322	0.343	0.355	0.247	0.329	0.400	0.439	0.360	0.383	0.378	0.377

the scope of this paper. Moreover, HashNet is too strict with dissimilar image pairs (i.e. the binary codes of dissimilar images should be as different as possible). Considering that there are many visually similar categories in our datasets, such strict constraints significantly increase the difficulty of model training and thus possibly cause the unsatisfactory performances of HashNet on our datasets. In practice, the classification-based methods require groundtruth category labels, while the metric learning based methods only take advantage of similar/dissimilar information, which might be easier to obtain. Even though, by comparing Tables 1 and 4, DPH can also use such labels, and the corresponding performances are comparable with or even better than DNNH and HashNet, suggesting that the proposed framework is a general one. **Fourth**, although the binary codes in our method are learned for jointly encoding two kinds of information, the performance of DPH is still among the top of all methods, indicating that our dual purpose hash codes are competent to fulfil the first individual task, i.e. category retrieval. Note that the SSDH method is trained with the classification loss similarly as DPH, and an additional quantization loss is also adopted in SSDH to further improve the retrieval performance of binary codes. Therefore, the SSDH method could be seen as an approximation of the classification-loss-only version of DPH. As shown in Table 4, SSDH and DPH have similar performances on the category retrieval task, which further validates that the DPH model is competitive in the category retrieval task. More importantly, the SSDH model

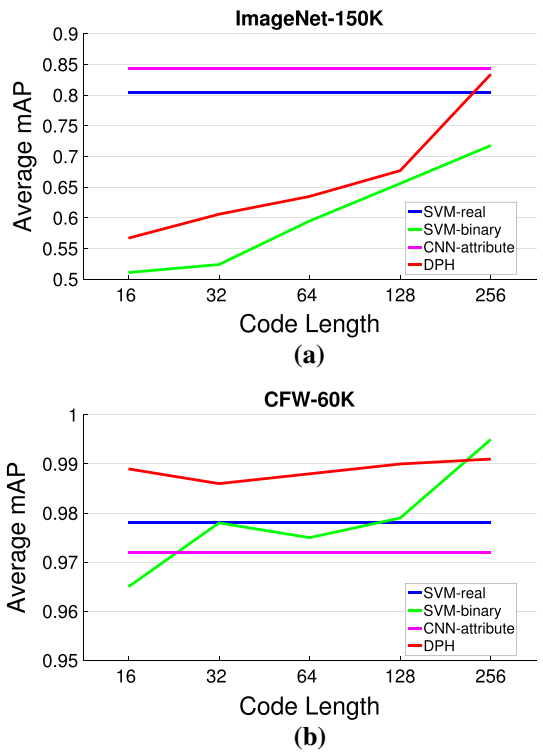
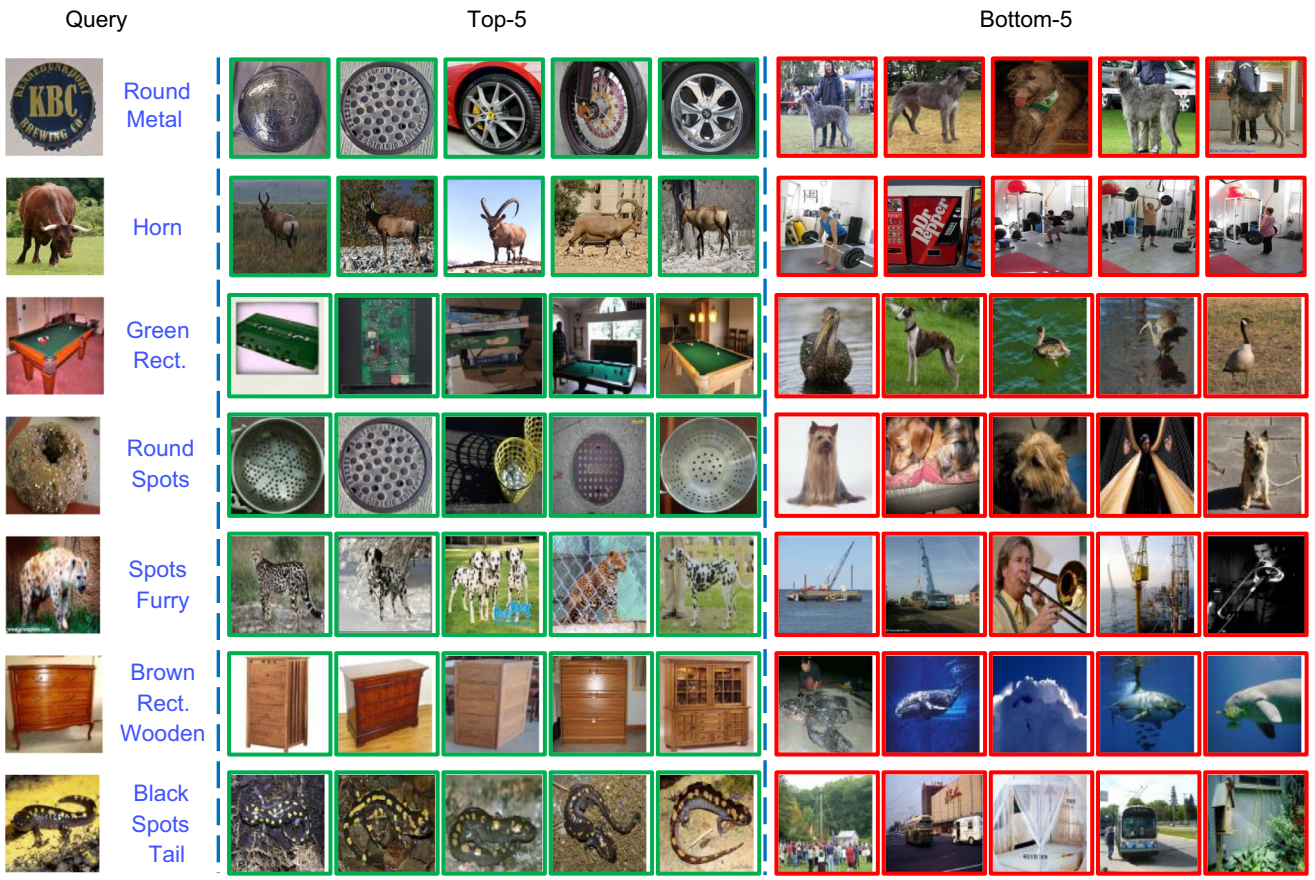


Fig. 12 Comparison of attribute retrieval performances (average mAP) of our method and other comparative methods on **a** ImageNet-150K and **b** CFW-60K. Note that SVM-real and CNN-attribute do not use binary codes as features, and thus their performances do not vary with code lengths



(a)

(b)

Fig. 13 Some real retrieval cases of the attribute retrieval task on both datasets. Here the "Test" set are used as queries, and the "Train-Both" set and "Train-Attribute" set are used together as database, which is a little different from the evaluation metrics. The selected attributes on ImageNet-150K are listed next to the query image, while the selected attributes on CFW-60K are gender, age, and race. Top-5 and Bottom-5 feedbacks are shown in the right part of the figure respectively. The notations here are consistent with Fig. 1. Best viewed in color (Color figure online)

could only perform the category retrieval task, verifying the advantage of DPH in simultaneously dealing with more diverse retrieval tasks.

4.3.2 Evaluation of Attribute Retrieval

Here we test on the second task in Sect. 1, i.e. attribute retrieval. We compare with three state-of-the-art methods for the attribute prediction part of retrieval: (1) Similar to Kumar et al. (2008), we train linear SVMs to predict attributes (we have found that the performance of linear and kernel SVMs are almost the same, thus we use linear SVMs for efficiency), using the same CNN features as described in Sect. 4.3.1. Then the prediction scores are normalized to [0, 1] using $sigmoid$ function. We denote this method as **SVM-real**, where "real" indicates that the models are trained on real-valued features. Although (Kumar et al. 2008) is proposed more than ten years ago, it is still a competitive non-deep attribute prediction method. More importantly, Kumar et al. (2008) separately predicts the binary attributes similarly to our method. Therefore, we choose this baseline method instead of the more advanced ones (e.g. Siddiquie et al. 2011; Sadovnik et al. 2013) for fair comparison. (2) We replace the CNN features in SVM-real with the 256-bit binary codes produced by SSDH in Sect. 4.3.1, which has the best performance among the comparative methods on the category retrieval task. This baseline is used to evaluate the necessity of jointly learning to encode the category and attribute information. We denote this method as **SVM-binary**. (3) We finetune the pre-trained CNN models to predict the attributes. For this purpose, we modify our network structure by directly adding the weighted sigmoid cross entropy loss (Sect. 3.3) after the feature extraction backbone network, which could be seen as an approximation of the attribute-loss-only version of DPH. We denote this method as **CNN-attribute**. All comparative methods are trained using the combination of "Train-Both" and "Train-Attribute" sets. The results with {16, 32, 64, 128, 256}-bit binary codes are reported.

The results are given in Fig. 12. We can observe that: **First**, the performances of our 256-bit binary codes are comparable to or even better than the baseline methods on both datasets. Note that our method even surpasses SVM-real and CNN-attribute, which are specially trained for attribute prediction task, suggesting that the additional category information might be helpful for the attribute learning task. Compared with the baseline methods, our method does not need to store the real-valued prediction scores, thus more storage-efficient than SVM-real and CNN-attribute. **Second**, SVM-binary is as compact as our method, yet achieves much inferior performance than our method. This might be explained by the fact that the binary codes learned by SSDH aims at encoding as much category information as possible, and thus suppresses the within-class attribute variations. As a result, the attribute information is lost and cannot be recovered from the 256-bit binary codes learned by SSDH.

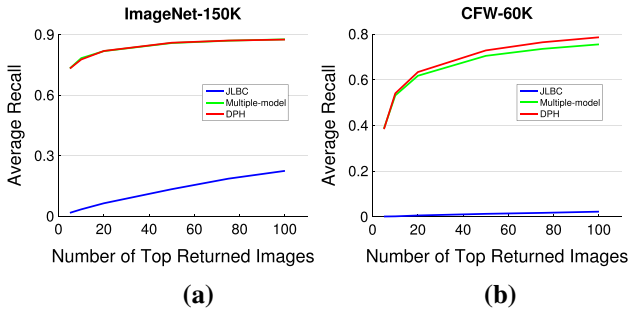


Fig. 14 Comparison of combined retrieval performance (average recall) of our method and other comparative methods on **a** ImageNet-150K and **b** CFW-60K. The results are obtained with 256-bit binary code. In this experiment, the "Multiple-model" baseline and our DPH method achieve very similar results on ImageNet-150K, and thus their corresponding curves (green and red curves respectively) highly overlap (Color figure online)

Some real retrieval examples on the attribute retrieval task are provided in Fig. 13, where the selected attributes on ImageNet-150K are listed next to the query image, and the selected attributes on CFW-60K are gender, age, and race. We can see that the top-ranked feedbacks (middle column) well match the query image, while the least matched images (rightmost column) have very different attributes compared with the query image. Such results further validate the effectiveness of our method in the attribute retrieval task.

4.3.3 Evaluation of Combined Retrieval

In this subsection, we evaluate on the third retrieval task in Sect. 1, i.e. the combined retrieval task. Since this is a relatively unexplored task, there are only a few methods that can address this problem. We compare our DPH with two baseline methods: (1) **JLBC** (Li et al. 2015), which can only use the fully annotated "Train-Both" set to train the model. We use the same CNN features as described above for this method. (2) **Multiple-model**. This baseline consists of two models, one for encoding each type of information, which corresponds to enforcing some bits to totally characterize category information and the others to characterize attributes. Here we use CNN-attribute in Sect. 4.3.2 for attribute prediction and SSDH (Yang et al. 2015) for Hamming distance ranking. The SSDH model is trained to produce $(k - p)$-bit binary codes, where k and p are the code length and number of attributes respectively, and the predictions of CNN-attribute are quantized to binary so that the storage cost of this baseline is the same as our DPH method. The experiments are conducted on 256-bit binary codes.

The results are shown in Fig. 14. We can see that: **First**, although CNN features are used to train the JLBC model

Fig. 15 Some real retrieval cases of the combined retrieval task on both datasets. Here the "Test" set are used as queries, and the "Train-Both" set and "Train-Attribute" set are used together as database, which is a little different from the evaluation metrics. The query image and the selected attribute are shown in the leftmost part of each row, and the corresponding top-10 feedbacks are given in the right part of each row. The notations here are consistent with Fig. 1. Best viewed in color (Color figure online)

on CFW-60K, due to the discrepancy between the Webface and CFW-60K datasets, the model performance is very unsatisfactory on this dataset, which confirms that our end-to-end framework is necessary for learning dual purpose hash codes that match with the target database. **Second**, our proposed DPH method performs on par with or better than the "Multiple-model" method on both datasets. Note that our method only needs one model to encode the two types of information, while the "Multiple-model" baseline needs two models. As a result, the "Multiple-model" method costs twice as much time as our method when generating binary codes and attribute predictions, validating the high efficiency of our proposed method in real applications. More importantly, if one choose to vary the number of bits for encoding the two kinds of information, it is difficult to decide the necessary number of bits for encoding each kind of information before training, which inevitably further increases the difficulty in training a model that separately encodes category

and attribute information in different bits. Such results further validate the advantage of letting the model automatically decide the functionality of each individual bit over manually deciding it.

We also provide some real retrieval results of the combined retrieval task on both datasets in Fig. 15. The top-10 feedbacks as well as their corresponding confidence levels on the selected attributes are shown in the right part of the figure. We can see that our model can successfully find the desired images, and even the wrong feedbacks (e.g. the image with a red bounding box in the first row) are very similar to the desired ones, which further validates the effectiveness of the proposed method in the combined retrieval task.

In the current framework, we use the predicted attribute values to filter out some images and then sort the remaining images by Hamming distance ranking. However, due to possible mistakes in attribute prediction, some of the relevant images might be filtered out in the first stage, and it would be impossible to successfully retrieve these images with the current pipeline. To deal with this problem, in the future, we would like to consider a soft filtering scheme, e.g. by weighting the Hamming distance between samples via the attribute prediction values.

4.4 Discussion

To sum up, our DPH method utilizes more supervised information than those state-of-the-art methods specifically designed for each individual task (i.e. category retrieval and attribute retrieval), one thus naturally expects that DPH should yield better performances on all tasks. However, as we can see from the above experiments, some attributes often vary significantly even within a single class (e.g. color attributes of trucks), the additional attribute information actually makes the learning of category more difficult. Specifically, the methods designed for category retrieval task mainly aim at suppressing such within-class attribute variations, while our dual purpose hash learning method aims at simultaneously addressing two somehow controversial goals, i.e. preserving the attribute variations and minimizing the overall within-class variations. Even though, the performances of our binary codes on the three retrieval tasks are still very competitive, while the computation cost of our method is much lower than training multiple models for these tasks, indicating that jointly preserving both category and attribute similarities for the three tasks is advantageous.

5 Conclusions

In this paper, we propose a method to learn hash functions that simultaneously preserve category and attribute similarities for multiple retrieval tasks. Our DPH method has achieved very competitive retrieval performances against state-of-the-art methods specifically designed for each individual task. The promising performance of our method can be attributed to: a) The utilization of CNN models for hierarchically capturing correlation between categories and attributes in an end-to-end manner. b) The loss functions specifically designed for the partially labelled training data, which can significantly improve the generalization ability of the models. Note that our framework is quite general, thus more powerful network structures and loss functions other than the ones discussed in this paper can be easily incorporated to further improve the performance.

Acknowledgements This work was done at the Institute of Computing Technology, Chinese Academy of Sciences, where Haomiao Liu pursued the PhD degree. This work is partially supported by 973 Program under contract No. 2015CB351802, Natural Science Foundation of China under contracts Nos. 61390511, 61772500, CAS Frontier Science Key Research Project No. QYZDJ-SSWJSC009, and Youth Innovation Promotion Association No. 2015085.

References

Al-Halah, Z., Lehrmann, A. M., & Sigal, L. (2018). *Towards traversing the continuous spectrum of image retrieval*. arXiv preprint arXiv:1812.00202.

Bau, D., Zhou, B., Khosla, A., Oliva, A., & Torralba, A. (2017). Network dissection: Quantifying interpretability of deep visual representations. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp 3319–3327).

Cakir, F., He, K., & Sclaroff, S. (2018). Hashing with binary matrix pursuit. In: *Proceedings of the European conference on computer vision (ECCV)* (pp. 332–348).

Cao, J., Li, Y., & Zhang, Z. (2018). Partially shared multi-task convolutional neural network with local constraint for face attribute learning. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 4290–4299).

Cao, Y., Liu, B., Long, M., & Wang, J. (2018). Hashgan: Deep learning to hash with pair conditional wasserstein gan. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1287–1296).

Cao, Y., Long, M., Liu, B., & Wang, J. (2018). Deep cauchy hashing for hamming space retrieval. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1229–1237).

Cao, Z., Long, M., Wang, J., & Yu, P. S. (2017). Hashnet: Deep learning to hash by continuation. In: *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 5609–5618).

Deng, C., Chen, Z., Liu, X., Gao, X., & Tao, D. (2018). Triplet-based deep hashing network for cross-modal retrieval. *IEEE Transactions on Image Processing (TIP)*, *27*(8), 3893–3903.

Escorcia, V., Niebles, J. C., & Ghanem, B. (2015). On the relationship between visual attributes and convolutional networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1256–1264).

Gionis, A., Indyk, P., & Motwani, R. (1999). Similarity search in high dimensions via hashing. *Very Large Data Base (VLDB)*, *99*, 518–529.

Gong, Y., & Lazebnik, S. (2011). Iterative quantization: A procrustean approach to learning binary codes. In: *Proceedings of the IEEE*

conference on computer vision and pattern recognition (CVPR) (pp. 817–824).

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In: *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 2961–2969).

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 770–778).

Hu, R., Xu, H., Rohrbach, M., Feng, J., Saenko, K., & Darrell, T. (2016). Natural language object retrieval. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 4555–4564).

Huang, C., Loy, C. C., & Tang, X. (2016). Unsupervised learning of discriminative attributes and visual representations. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 5175–5184).

Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., & Darrell, T. (2014). Caffe: Convolutional architecture for fast feature embedding. In: *International conference on multimedia (MM)* (pp. 675–678).

Jiang, Q. Y., Cui, X., & Li, W. J. (2018). Deep discrete supervised hashing. *IEEE Transactions on Image Processing (TIP)*, *27*(12), 5996–6009.

Jiang, Q. Y., & Li, W. J. (2017). Deep cross-modal hashing. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3232–3240).

Kokkinos, I. (2017). Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 5454–5463).

Kovashka, A., & Grauman, K. (2013). Attribute adaptation for personalized image search. In: *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 3432–3439).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In: *Advances in neural information processing systems (NIPS)* (pp. 1097–1105).

Kulis, B., & Darrell, T. (2009). Learning to hash with binary reconstructive embeddings. In: *Advances in neural information processing systems (NIPS)* (pp. 1042–1050).

Kumar, N., Belhumeur, P., & Nayar, S. (2008). Facetracer: A search engine for large collections of images with faces. In: *European conference on computer vision (ECCV)* (pp. 340–353).

Lai, H., Pan, Y., Liu, Y., & Yan, S. (2015). Simultaneous feature learning and hash coding with deep neural networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3270–3278).

Li, Y., Wang, R., Liu, H., Jiang, H., Shan, S., & Chen, X. (2015). Two birds, one stone: Jointly learning binary code for large-scale face image retrieval and attributes prediction. In: *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 3819–3827).

Liu, H., Wang, R., Shan, S., & Chen, X. (2016). Deep supervised hashing for fast image retrieval. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2064–2072).

Liu, H., Wang, R., Shan, S., & Chen, X. (2017). Learning multifunctional binary codes for both category and attribute oriented retrieval tasks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 6259–6268).

Liu, L., Chen, J., Fieguth, P., Zhao, G., Chellappa, R., & Pietikäinen, M. (2019). From bow to cnn: Two decades of texture representation for texture classification. *International Journal of Computer Vision*, *127*(1), 74–109.

Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., et al. (2020). Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, *128*(2), 261–318.

Liu, W., Wang, J., Ji, R., Jiang, Y. G., & Chang, S. F. (2012). Supervised hashing with kernels. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2074–2081).

Liu, X., He, J., Deng, C., & Lang, B. (2014). Collaborative hashing. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2139–2146).

Liu, X., He, J., Lang, B., & Chang, S. F. (2013). Hash bit selection: a unified solution for selection problems in hashing. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1570–1577).

Liu, X., Huang, L., Deng, C., Lu, J., & Lang, B. (2015). Multi-view complementary hash tables for nearest neighbor search. In: *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 1107–1115).

Liu, Z., Luo, P., Qiu, S., Wang, X., & Tang, X. (2016). Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1096–1104).

Long, Y., Liu, L., Shen, Y., & Shao, L. (2018). Towards affordable semantic searching: Zero-shot retrieval via dominant attributes. In: *Thirty-Second AAAI conference on artificial intelligence*.

Norouzi, M., & Fleet, D. J. (2011). Minimal loss hashing for compact binary codes. In: *International conference on machine learning (ICML)* (pp. 353–360).

Parikh, D., & Grauman, K. (2011). Interactively building a discriminative vocabulary of nameable attributes. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1681–1688).

Patterson, G., Xu, C., Su, H., & Hays, J. (2014). The sun attribute database: Beyond categories for deeper scene understanding. *International Journal of Computer Vision (IJCV)*, *108*(1–2), 59–81.

Rastegari, M., Diba, A., Parikh, D., & Farhadi, A. (2013). Multi-attribute queries: To merge or not to merge? In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3310–3317).

Rastegari, M., Farhadi, A., & Forsyth, D. (2012). Attribute discovery via predictable discriminative binary codes. In: *European conference on computer vision (ECCV)* (pp. 876–889).

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., et al. (2015). Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, *115*(3), 211–252.

Sadovnik, A., Gallagher, A., Parikh, D., & Chen, T. (2013). Spoken attributes: Mixing binary and relative attributes to say the right thing. In: *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 2160–2167).

Scheirer, W. J., Kumar, N., Belhumeur, P. N., & Boult, T. E. (2012). Multi-attribute spaces: Calibration for attribute fusion and similarity search. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2933–2940).

Shen, F., Shen, C., Liu, W., & Tao Shen, H. (2015). Supervised discrete hashing. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 37–45).

Shen, L., Lin, Z., & Huang, Q. (2016). Relay backpropagation for effective learning of deep convolutional neural networks. In: *European conference on computer vision (ECCV)* (pp. 467–482).

Siddiquie, B., Feris, R. S., & Davis, L. S. (2011). Image ranking and retrieval based on multi-attribute queries. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 801–808).

Sun, Y., Chen, Y., Wang, X., & Tang, X. (2014). Deep learning face representation by joint identification-verification. In: *Advances in neural information processing systems (NIPS)* (pp. 1988–1996).

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper

with convolutions. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1–9).

Tao, R., Smeulders, A. W. M., & Chang, S. F. (2015). Attributes and categories for generic instance search from one example. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 177–186).

Turakhia, N., & Parikh, D. (2013). Attribute dominance: What pops out? In: *Proceedings of the IEEE international conference on computer vision (ICCV)* (pp. 1225–1232).

Veit, A., Belongie, S., & Karaletsos, T. (2017). Conditional similarity networks. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 830–838).

Wang, J., Kumar, S., & Chang, S. F. (2012). Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *34*(12), 2393–2406.

Wang, L., Lee, C. Y., Tu, Z., & Lazebnik, S. (2015). *Training deeper convolutional networks with deep supervision*. arXiv preprint arXiv:1505.02496.

Weiss, Y., Torralba, A., & Fergus, R. (2008). Spectral hashing. In: *Advances in neural information processing systems (NIPS)* (pp. 1753–1760).

Xia, R., Pan, Y., Lai, H., Liu, C., & Yan, S. (2014). Supervised hashing for image retrieval via image representation learning. In: *Twenty-eighth AAAI conference on artificial intelligence*.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., & Torralba, A. (2010). Sun database: Large-scale scene recognition from abbey to zoo. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3485–3492).

Yang, E., Deng, C., Liu, W., Liu, X., Tao, D., & Gao, X. (2017). Pairwise relationship guided deep hashing for cross-modal retrieval. In: *Thirty-first AAAI conference on artificial intelligence*.

Yang, H. F., Lin, K., & Chen, C. S. (2015). Supervised learning of semantics-preserving hash via deep convolutional neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, *40*(2), 27–35.

Yi, D., Lei, Z., Liao, S., & Li, S. Z. (2014). *Learning face representation from scratch*. arXiv preprint arXiv:1411.7923.

Yu, F. X., Ji, R., Tsai, M. H., Ye, G., & Chang, S. F. (2012). Weak attributes for large-scale image retrieval. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 2949–2956).

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., & Savarese, S. (2018). Taskonomy: Disentangling task transfer learning. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 3712–3722).

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

Zhang, X., Zhang, L., Wang, X. J., & Shum, H. Y. (2012). Finding celebrities in billions of web images. *IEEE Transactions on Multimedia (TMM)*, *14*(4), 995–1007.

Zhang, Z., Chen, Y., & Saligrama, V. (2016). Efficient training of very deep neural networks for supervised hashing. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1487–1495).

Zhao, F., Huang, Y., Wang, L., & Tan, T. (2015). Deep semantic ranking based hashing for multi-label image retrieval. In: *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 1556–1564).

Zhong, Y., Sullivan, J., & Li, H. (2016). *Face attribute prediction with classification cnn*. arXiv preprint arXiv:1602.01827.

Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., & Oliva, A. (2014). Learning deep features for scene recognition using places database. In: *Advances in neural information processing systems (NIPS)* (pp. 487–495).