Dual Purpose Hashing

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

[1]Key Lab of Intelligent Information Processing of Chinese Academy of Sciences(CAS), Institute of Computing Technology, CAS, Beijing, 100190, China
[2]University of Chinese Academy of Sciences, Beijing, 100049, China
haomiao.liu@vipl.ict.ac.cn, {wangruiping, sgshan, xlchen}@ict.ac.cn

Abstract. Recent years have seen more and more demand for a unified framework to address multiple realistic image retrieval tasks concerning both category and attributes. Considering the scale of modern datasets, hashing is favorable for its low complexity. However, most existing hashing methods are designed to preserve one single kind of similarity, thus improper for dealing with the different tasks simultaneously. To overcome this limitation, we propose a new hashing method, named Dual Purpose Hashing (DPH), which jointly preserves the category and attribute similarities by exploiting the Convolutional Neural Network (CNN) models to hierarchically capture the correlations between category and attributes. Since images with both category and attribute labels are scarce, our method is designed to take the abundant partially labelled images on the Internet as training inputs. With such a framework, the binary codes of new-coming images can be readily obtained by quantizing the network outputs of a binary-like layer, and the attributes can be recovered from the codes easily. Experiments on two large-scale datasets show that our dual purpose hash codes can achieve 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: Binary Codes, Hashing, Visual Attribute

1 Introduction

In recent years, more and more images are available on the Internet, posing great challenges to retrieving images relevant to a given query image. At the meantime, the retrieval tasks have also become more diverse. In real-life scenarios, three common retrieval tasks are: **I**. retrieving images from the same category as the query image, e.g. matching street clothing photos in online shops [1]; **II**. retrieving images with specified attributes, e.g. looking for young Asian woman wearing sunglasses [2]; and **III**. the combination of the above tasks, e.g. looking for clothing of the same style but with a different color. Existing algorithms [1,2,3,4] can be adopted to tackle the above tasks, and have achieved certain degree of successes. However, the high complexities of indexing and retrieving with real-valued image representations limit the scalability of such methods. To deal with this problem, hashing is often adopted for its high efficiency in both time and storage.

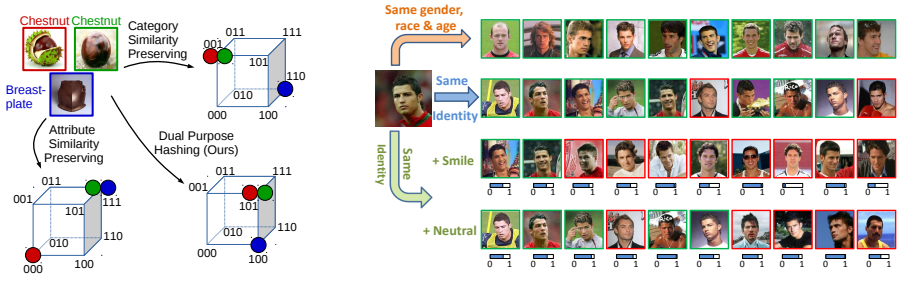


Fig. 1. (a)Illustration of the idea of our Dual Purpose Hashing method. (b) A real example showing the three retrieval tasks on a face dataset, where the query is an image of Cristiano Ronaldo. The top ranked feedbacks of each task are shown here. In the first two rows, images exactly match the query 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 below the images indicate the confidence level of the corresponding attribute. Best viewed on a computer screen.

A major issue concerning most existing hashing methods is that they are usually designed to preserve one single kind of similarity, e.g. semantic similarity defined by categories. Due to the difference between attributes and category, multiple models would be needed to preserve both category and attribute similarities for satisfactory performance. However, such scheme is suboptimal since training multiple models is time-consuming, and the redundancies between the models might harm the storage efficiency. To tackle this issue, we propose a unified framework to jointly preserve both similarities, named Dual Purpose Hashing (DPH), as illustrated in Figure 1(a). In our DPH method, only a single model is learned to produce binary codes that can be used to simultaneously deal with the three tasks above, thus reducing the training time and redundancies in storage. Figure 1(b) shows a real face image retrieval case of our method on a challenging face dataset.

Our basic idea comes from a very natural intuition that category and attributes, as objects' descriptions at different semantic levels, should share some common low-level visual features. This can be partly confirmed from the experimental studies in some recent works [5,6], where it is shown that some nodes in the top layers of CNNs trained for classification tasks are highly correlated with visual attributes. 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 Figure 2. To be specific, our network contains a binary-like layer, which is used to approximate the binary code. By jointly optimizing a classification loss and an attribute prediction loss, our method can encode both similarities into the binary codes. Since most images available on the Internet do not have complete category and attribute labels, our loss function is properly designed to take into account such practical scenarios, namely, even images with only one label can contribute to the model learning.

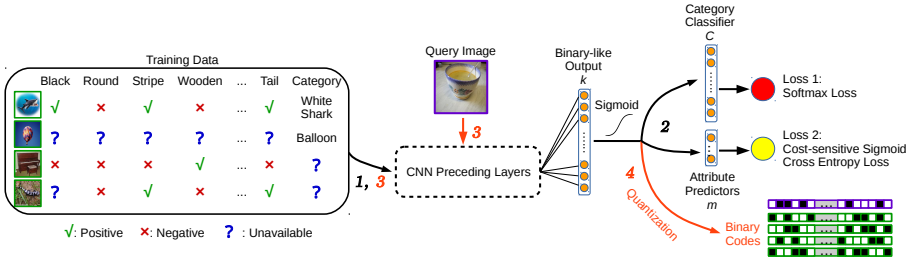


Fig. 2. The framework of our proposed DPH method. To simultaneously encode category and visual attributes of images into binary codes, we devise a CNN model that can take partially labelled images as training input (step 1), and train the model on classification and attribute prediction tasks (step 2).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

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 is learned, images can be indexed by quantizing the outputs of the binary-like layer to compact hash codes. In the first task relevant to category, retrieval can be done similarly to existing hashing methods by utilizing Hamming distance ranking or hash table lookup. For the last two tasks relevant to attributes, real-valued attribute predictions (in general, such real-valued predictions are more powerful than binary-valued alternatives) can be recovered from the binary codes with a simple matrix multiplication operation, which can be efficiently done by only a few summation operations. Compared with directly storing the real-valued attribute predictions, our method only incurs a little increase in computation cost, while dramatically reduces the storage space.

The main contributions of this work are two-fold: First, we present a unified framework to learn hash functions that simultaneously preserve category and attribute similarities for addressing multiple retrieval tasks. Second, 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.

2 Related Works

In large-scale retrieval tasks, hashing [7,8,9,10,11,12,13,14,15] is favorable for its low time and space complexity. The pioneering data-independent hashing method Locality Sensitive Hashing (LSH) [7], uses random projections to produce binary bits. However, LSH usually requires long codes to achieve satisfactory retrieval performance, which demands for large amount of storage space.

To reduce the storage cost, data-dependent hashing methods are proposed to learn more compact binary codes by utilizing a training set. Such methods can be further divided into unsupervised and supervised (semi-supervised). Unsu-

pervised methods only make use of unlabelled training data to learn hash functions. For example, Spectral Hashing (SH) [8] aims at minimizing the weighted Hamming distance of binary codes, where the weights are defined as the similarity metrics of image pairs; Iterative Quantization (ITQ) [11] attempts to orthogonally transform the image descriptor such that the quantization errors are minimized.

Supervised methods are proposed to deal with the more complicated semantic similarities by taking advantage of semantic labels. CCA-ITQ [11] boosts the performance of ITQ by using label information to obtain more discriminative image descriptors; Minimal Loss Hashing (MLH) optimizes an upper bound of a hinge-like loss to learn the hash functions. On the other hand, Semi-Supervised Hashing (SSH) [10] uses unlabelled data to improve the generalization ability of the hash functions. Since the above methods use linear hash functions, they can hardly deal with linearly inseparable data. To overcome this limitation, Binary Reconstructive Embedding (BRE) [9] and Supervised Hashing with Kernels (KSH) [13] are proposed to take advantage of the non-linearity of kernel spaces.

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, thus might lose some task-specific information. To tackle this issue, most recently, several hashing methods [16,15,14,17,18] significantly improve the state-of-the-arts by jointly learning the image representations and the hash functions using CNN models.

Besides category label-oriented image retrieval, attributes have been widely adopted to deal with some other realistic retrieval tasks [19,20,21,22,23,24,25,4,2]. This paper is most related to the works that use nameable attributes [26] as queries. [22] predicts the probability of attributes with SVM classifiers, and uses the product of probabilities to rank the database images. Follow-up works investigate the usage of attribute correlation [2], fusion strategy [21,25], relative attributes [19], and other techniques [20,24] to improve the retrieval performance. In this paper, we adopt the retrieval strategy in [22] for simplicity, while those more complicated ones [2,21,25] are also compatible with our framework. A major issue of these attributes-oriented image retrieval methods is the usage of real-valued features, which limits the scalability and efficiency of such methods.

In light of the successes of hashing methods, recently [27,28] have made some early attempts to connect attributes with binary codes. [27] tries to discover attributes after the hash functions are learned 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 [27] unsuitable to be used for attribute-oriented retrieval tasks. [28] improves [27] by explicitly modeling 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 [28] is obviously inadequate to capture the complex correlation between category and attributes. To address the shortcomings of previous works, we propose to exploit the CNN models to hierarchically extract the correlation between these two semantic descriptions in an end-to-end manner.

3 Approach

Our goal is to learn compact binary codes such that: a) images from the same category are encoded to similar binary codes; b) images with similar attributes should have similar binary codes; c) the learned model should generalize well to new-coming images.

To achieve this goal, we present a hash learning framework as illustrated in Figure 2. The preceding layers of the network consists of several convolution-pooling layers, and optionally followed by several fully connected layers. The structure of these layers is very flexible, thus various successful models [29,30,31] can be adopted in our method. Since directly optimizing binary codes is difficult, the penultimate layer in our network is designed to give binary-like outputs (a fully connected layer with *sigmoid* activation) to approximate the binary codes. During the training stage, the whole network is jointly trained on classification and attribute prediction tasks 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 Section 4.2.

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 $\mathscr{F} : \Omega \rightarrow \{0,1\}^k$. Suppose that the training images are from C known categories, and annotated with a set of m visual attributes. Let $S_{tr} = \{(X_i^{tr}, y_i, \mathbf{a}_i)|i = 1, \cdots, N\}$ denote the training set consisting of N images, where $X_i^{tr} \in \Omega$, $y_i \in \{1, \cdots, C, C+1\}$ is the category label of the i-th image, and $\mathbf{a}_i \in \{0,1,2\}^m$ are the visual attribute labels. More specifically, $y_i = C+1$ means the category label of the i-th image is missing. $\mathbf{a}_{ij} = 1$ and 0 indicates the j-th attribute is present/absent in the i-th image. Besides, we use $\mathbf{a}_{ij} = 2$ to denote that the j-th attribute label of the i-th image is missing. Each training image is required to have at least one available label.

3.2 Category Information Encoding

To preserve category similarity, our basic idea is that if a simple transformation (e.g. softmax classifier) can recover the category label from the binary codes, the category information would have been encoded into the binary codes. Note that 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} as:

$$L_i^{cls} = -\sum_{c=1}^{C} \mathbb{I}\{y_i = c\} log \frac{s_c}{\sum_{l=1}^{C} s_l} \tag{1}$$

where the superscript cls indicates classification, $\mathbb{I}\{\cdot\}$ is 1 when the condition is true and 0 otherwise, s_l denotes the l-th output of the softmax classifier. 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 will not contribute to the classification loss.

3.3 Attributes Encoding

To preserve attribute similarity, the similar idea to Section 3.2 is exploited, 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 m attributes, we define the loss as a logistic regression problem. To handle the missing label case, the standard formulation of logistic regression is modified to suit in our problem. Specifically, the j-th ($j \in \{1, 2, \cdots, m\}$) attribute prediction loss of a single training image X_i^{tr} is defined as a modified cross entropy loss:

$$L_{ij}^{attr} = -\mathbb{I}\{\mathbf{a}_{ij} \neq 2\}[\mathbf{a}_{ij} log(p_{ij}) + (1 - \mathbf{a}_{ij}) log(1 - p_{ij})] \tag{2}$$

where the superscript $attr$ denotes attribute, p_{ij} is the estimated probability that the i-th image possesses the j-th attribute.

Directly optimizing Eqn.(2) might lead to collapsed solution, since the distribution of some attributes are highly imbalanced (i.e. only a tiny portion of images have/do not have these attributes), even predicting all images as negative/positive would result in a relatively low loss. To alleviate the impact of sample imbalance, we propose a cost-sensitive version of Eqn.(2) instead:

$$L_{ij}^{attr}(\mathbf{w}) = -\mathbb{I}\{\mathbf{a}_{ij} \neq 2\}[\frac{w_j}{w_j + 1} \mathbf{a}_{ij} log(p_{ij}) + \frac{1}{w_j + 1}(1 - \mathbf{a}_{ij}) log(1 - p_{ij})] \tag{3}$$

where w_j is a weighting parameter controlling the relative strength of the positive and negative samples. In practice, we set w_j according to the ratio of the negative sample size to the positive sample size on the training set.

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 Eqn.(1) and Eqn.(3) as the overall loss function may be problematic. To be specific, the values of Eqn.(1) and Eqn.(3) 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:

$$L = \frac{1}{\sum_{t=1}^{n} \mathbb{I}\{y_t \leq C\}} \sum_{i=1}^{n} L_i^{cls} + \alpha \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{L_{ij}^{attr}(\mathbf{w})}{\sum_{t=1}^{n} \mathbb{I}\{\mathbf{a}_{tj} \neq 2\}} \tag{4}$$

where α is an extra weighting parameter to control the relative strength of the classification loss and the attribute prediction loss. In case of $\sum_{t=1}^{n} \mathbb{I}\{y_t \leq C\} = 0$ or $\sum_{t=1}^{n} \mathbb{I}\{\mathbf{a}_{tj} \neq 2\} = 0$, the corresponding loss term is set to zero.

The gradients of Eqn.(4) can be easily computed analogically to the standard softmax classifier, except for multiplying the weighting and scaling parameters, thus we do not bother to discuss them in detail. For the training images, their binary codes can be easily obtained by quantizing the corresponding binary-like network outputs.

3.5 Retrieval

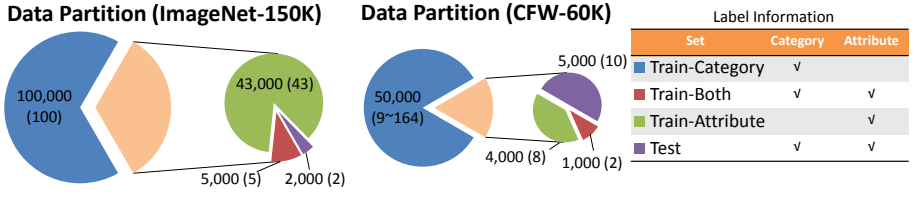
After the model is learned, the binary codes of new-coming images can be similarly obtained as above by propagating through the network and then quantizing the outputs of the binary-like layer. To accomplish the three retrieval tasks, we need to further recover the attribute predictions from the binary codes, which can be done by multiplying the binary codes with the attribute classifier weights. Note that the recovery of attribute prediction scores can be efficiently fulfilled by only a few summation operations, and only one more matrix (holding the attribute classifier weights) of size $k \times m$ (where k is the code length, and m is the number of attributes) needs to be stored compared to other hashing methods. Therefore, our method is efficient in both time and storage.

4 Experiments

In this section, we extensively evaluate our method on two large-scale datasets. First we evaluated the impact of additional partially labelled data on the retrieval and attribute prediction tasks. Then the proposed DPH method was compared with the state-of-the-art retrieval methods on each of the three tasks to validate the advantages of our method.

4.1 Experimental Settings

Datasets: We evaluated our DPH method on two large-scale partially labelled datasets: (1) **ImageNet-150K** is a subset of ILSVRC2012 dataset [32] with 150,000 images. For each of the 1,000 categories, we selected 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 2,000 selected validation images are manually annotated with 25 attributes (including color, texture, shape, material, and structure). We partitioned the dataset into 4 parts (Train-Category, Train-Both, Train-Attribute, and Test) as illustrated in Figure 3(a). Please refer to the supplementary materials for more details about this dataset. (2) **CFW-60K** [28] is a purified subset of the original CFW dataset [33] and contains 60,000 images of 500 subjects, among which 20 images of each subject (10K images in total) are annotated with 14 attributes. For the 10K images with attribute annotations, 5K (10 images for each subject) were used as Test set, and the rest 5K were further divided into two parts (Train-Both and



(a) (b) (c)

Fig. 3. Illustration of data partition in our experiments. (a) ImageNet-150K with 1,000 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 amount of images from each category. (c) The label information of the corresponding sets on the left, where the tick means the corresponding label is available. This figure is best viewed in color.

Train-Attribute). For the remaining 50K images without attribute annotation, they were used as Train-Category set. The details of partitioning is illustrated in Figure 3(b). Please refer to the original publications [33,28] for more details about this dataset. On both datasets, the category labels of the Train-Attribute set were made unavailable to all methods.

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. The results are the mean performance over all images in the Test set. Since the three retrieval tasks are very different from each other, the details of the evaluation metrics for each task will be defined in their corresponding subsections (Section 4.3-4.5) respectively..

Implementation details: Although we have thousands of images as training data, our datasets are still relatively small in terms of training a CNN model from scratch. In consideration of generalization ability, the model parameters were initialized using pre-trained models. For ImageNet-150K, we used the publicly available CaffeNet model provided in the model zoo of Caffe [34]. The model parameters from the conv1 layer to the fc7 layer were used to initialize our models. For CFW-60K, we adopted the CNN structure of [35] as the preceding layers (from conv1 to pool5). Since the pre-trained model is not available, we followed the original publication [35] to train the model with their published dataset, except for removing the contrastive loss for simplicity.

For ImageNet-150K, since the pre-trained model was trained on the same dataset as the target dataset, the model was trained for 40 epochs. In contrast, for CFW-60K, since the pre-trained model was obtained from a different dataset, the model was trained for 100 epochs. We set the learning rate to 10^{-3} for the preceding layers, and 10^{-2} for the other layers with a batch size of 200. The momentum and weight decay parameters were set according to the original publications [35,34]. Besides, on both datasets, we empirically set the weighting parameter $\alpha = 0.1$ in Eqn.(4). All the comparison deep learning methods were implemented with Caffe [34] [1].

[1] The source code of DPH and the ImageNet-150K dataset will be released to the public.

Table 1. Comparison of the 128-bit models trained with different combinations of training data. The retrieval mAP and mean F1-score over all attributes are shown in the last two columns respectively. B: Both, A: Attribute, C: Category.

Model	Dataset	mAP	mean F1-score	Dataset	mAP	mean F1-score
B	ImageNet-150K	0.248	0.753	CFW-60K	0.095	0.817
B + A		0.239	0.856		0.088	0.867
B + C		0.336	0.828		0.233	0.814
B + A + C		0.343	0.879		0.241	0.877

4.2 Evaluation of Partially Labelled Data

We first evaluate the impact of utilizing partially labelled data on both datasets by using 128-bit binary codes as example. For this purpose, 4 models were 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 data (please refer to Section 4.1 and Figure 3 for details) used to train the specific model. In this subsection, the encoding of category and attributes are evaluated separately. For the category part, we rank the database images according to the Hamming distance between their binary codes and the codes of the query image, and the performance is measured by mAP, where images from the same category are deemed as relevant. For the attribute part, for simplicity, we calculate the F1-scores [36] of predicting each attribute, and report the mean F1-score over all attributes. Note that since some attributes are highly unbalanced, e.g. most images do not possess the attribute "orange" in ImageNet-150K, F1-score can more faithfully reflect the real performance.

The comparison results are given in Table 1. We can infer that: First, compared with the "Both" model, exploiting extra training data (B + A and B + C) improves the performance of the corresponding task by a large margin. This observation can be explained by model overfitting, to be specific, in our experiments, when the model was trained only using the "Train-Both" set, the training loss approached zero while the test loss only decreased slightly. In contrast, when additional data was introduced to train the model, the training loss and test loss of the corresponding tasks were always on the same scale as normally expected. This justifies our motivation of using partially labelled data to train the CNN models to alleviate overfitting. Second, compared with training solely on "Train-Both" set, using both kinds of additional data can significantly improve the performance on both tasks (the row "B + A + C" in Table 1), 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, confirming that it is feasible to simultaneously embed category and visual attributes into the binary codes by exploiting partially labelled data. In the following experiments, all our models are trained with the "B + A + C" setting.

4.3 Evaluation of Category Retrieval

In this subsection, we test the effectiveness of our DPH method on the first task specified in Section 1, namely, given a query image, retrieving images of the same

Table 2. Comparison of category retrieval performance (mAP) of our method and other comparative hashing methods on ImageNet-150K and CFW-60K. The best performance of each code length is highlighted in boldface.

	ImageNet-150K					CFW-60K				
	16-bit	32-bit	64-bit	128-bit	256-bit	16-bit	32-bit	64-bit	128-bit	256-bit
ITQ [11]	0.102	0.167	0.235	0.284	0.310	0.039	0.058	0.079	0.112	0.135
CCA-ITQ [11]	0.090	0.157	0.223	0.294	0.341	0.048	0.069	0.090	0.113	0.140
DBC [27]	0.207	0.264	0.308	0.344	0.369	0.045	0.060	0.072	0.099	0.129
KSH [13]	0.110	0.181	0.253	0.293	0.320	0.046	0.063	0.086	0.111	0.117
SDH [37]	0.082	0.143	0.222	0.288	0.322	0.026	0.049	0.095	0.140	0.183
DNNH [15]	0.102	0.147	0.213	0.267	0.298	0.035	0.058	0.100	0.148	0.185
DLBHC [17]	0.197	0.263	0.310	0.339	0.357	0.068	0.109	0.173	0.235	0.279
DPH	0.212	0.274	0.322	0.343	0.353	0.064	0.112	0.186	0.241	0.274

category from the database. The retrieval is done by ranking the binary codes of database images according to the Hamming distances to the query image.

Comparative methods: We compare with seven representative hashing methods: ITQ [11], CCA-ITQ [11], DBC [27], KSH [13], SDH [37], DNNH [15], and DLBHC [17], including representative linear and non-linear conventional hashing methods as well as the state-of-the-art deep CNN-based methods.

For fair comparison, the "shallow" hashing methods were trained using the L2-normalized CNN features extracted from the pre-trained CNN models (described in Section 4.1). The comparative methods were implemented using the source code provided by the original authors except for LSH. Instead, the projection parameters of LSH were randomly drawn from a normal distribution. As for the "deep" methods, DLBHC and DNNH exploited the same preceding layers as our DPH method, and were initialized with the identical pre-trained models as ours. In particular, since the category number is larger than the batch size in our setting, DNNH would fail to converge if the training images are randomly shuffled, mainly because the number of "valid" triplets in each iteration is too small. To make DNNH converge successfully, we hence randomly selected 10 categories and 20 images per category to form each mini-batch.

The comparative methods were trained using the combination of the two sets "Train-Both" and "Train-Category" to preserve the category similarity. Since KSH demands large amount of memory to store the kernel matrix ($O(N^2)$, where N is the number of training images), we used 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 were tuned carefully according to the original publications. The experiments were carried on $\{16, 32, 64, 128, 256\}$-bit binary codes.

Evaluation metric: For evaluation, we use mean Average Precision (mAP), where images with the same category label are considered as relevant.

Results: The comparison results are shown in Table 2. We have the following observations: **First**, when equipped with CNN features, the conventional non-linear hashing method KSH can hardly improve the retrieval performance over linear methods. One possible explanation is that the CNN has mapped the images to a feature space where images from different categories are roughly

linearly separable, thus KSH can hardly benefit from the non-linearity of kernel space. In addition, the smaller training set of KSH is also a possible explanation. **Second**, in terms of retrieval mAP, CNN-based methods significantly improve over conventional hashing methods on CFW-60K, yet have marginal improvement on ImageNet-150K. Note that the pre-trained model on CFW-60K was obtained from a different dataset, while on ImageNet-150K from the same one, validating the advantage of CNN-based hashing methods lies in learning image representations which are more suitable for the data at hand than pre-defined features. **Third**, DNNH performs relatively worse than the other two CNN-based methods [2]. While this might be attributed to the batch generation scheme particularly designed for this method as described above, it seems to imply that the training data should be carefully organized for DNNH to yield favorable performance. **Fourth**, the performance of DPH is among the top of all methods, even though the binary codes were learned for jointly tackling two kinds of different tasks, indicating that our dual purpose hash codes is competent to fulfil the first individual task - category retrieval.

4.4 Evaluation of Attribute Retrieval

In this subsection, we test the effectiveness of our DPH method on the second task described in Section 1. The attribute prediction scores of DPH can be recovered from the binary codes using the method described in Section 3.5. In this experiment, given an image, we randomly select at most three attributes as query, whose values are specified by the image (thus can be either positive or negative). The system is required to retrieve images such that the selected attributes of the top ranked images are the same as the ones of the query image. To be specific, the database images were ranked in descending order by the products of attribute prediction scores.

Comparative methods: We compare with three baseline methods for attribute prediction: 1) Similar to [22], we train linear SVM classifiers to predict attributes (in the experiments, we found that replacing the linear SVMs with kernel SVMs only gives marginal improvement, thus we adopted the linear SVMs for efficiency), using the same CNN features as the "shallow" hashing methods described in Section 4.3. Then the prediction scores are normalized to the range of $(0, 1)$ using *sigmoid* function. We denote this method as **SVM-real**, where "real" indicates that the models were trained on real-valued features. 2) We replace the CNN features in SVM-real with the 256-bit binary codes produced by DLBHC in Section 4.3. This baseline is used to evaluate the necessity of jointly encoding the category and attributes. We denote this method as **SVM-binary**. 3) We finetune the pretrained CNN models to solely predict the attributes. For this purpose, we modified our network structure by removing both the binary-like layer and the classification loss, and concatenating the attribute prediction loss right after the preceding layers. The models were trained using the combination

[2] The source code of DNNH was provided by the original authors, and our re-implementation on NUS-WIDE achieved similar result as reported in [15]

Table 3. Comparison of attribute retrieval performance (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 code as features, thus their performance do not vary with code lengths.

	ImageNet-150K					CFW-60K				
	16-bit	32-bit	64-bit	128-bit	256-bit	16-bit	32-bit	64-bit	128-bit	256-bit
SVM-real	0.903					0.765				
CNN-attribute	0.902					0.771				
SVM-binary	0.805	0.823	0.844	0.861	0.871	0.661	0.680	0.693	0.711	0.729
DPH	0.806	0.828	0.842	0.859	0.868	0.695	0.726	0.758	0.785	0.804

of the two sets "Train-Both" and "Train-Attribute", and the hyper-parameters were set as described in Section 4.1. We denote this method as **CNN-attribute**.

Evaluation metric: In this task, we also use mAP to measure the retrieval performance. Images that match with the query image at all selected attributes are considered as relevant. Note that in this experiment, the predicted attributes of all images (both query and database images) were used for retrieval, while the evaluation is performed on the ground-truth attribute labels. As a result, both wrong predictions of the query image and the database images would hurt the performance. We report the average mAP over all valid attribute queries.

Results: The results are given in Table 3, note that SVM-real and CNN-attribute have very close performances on ImageNet-150K, and their curves overlap. On both datasets, our 256-bit binary codes achieves comparable or even better performance than the baseline methods. Our method does not need to store the real-valued attribute prediction scores, thus compared to SVM-real and CNN-attribute, the storage space required by our method is much smaller. SVM-binary achieves similar performance with our method on ImageNet-150K, but much worse on CFW-60K. This could possibly be explained by the fact that ImageNet-150K contains more categories and attributes than CFW-60K, and the variation is thus more complex. As a result, the 256-bit code might be too short for this task. From the tendency in Figure **??**(a), we can hopefully expect that longer codes of our DPH method could achieve better performance. Some real retrieval results on this task are provided in Figure 4(a). Please refer to the supplementary materials for more examples.

4.5 Evaluation of Combined Retrieval

In this subsection, we evaluate our DPH method on the third retrieval task described in Section 1. In this experiment, the system is required to retrieve images belonging to the same category as the query image, while possessing an attribute that is absent in the query image. To accomplish this task, we use the attribute predictions to filter out the images that do not match in terms of the specified attribute, and then rank the remaining images using the Hamming distances. We compare the results of using 256-bit binary codes in this experiment.

Comparative methods: Since this is a relatively new task, we compare our DPH with two methods: 1) **JLBC** [28] is trained on "Train-Both" set, since it can

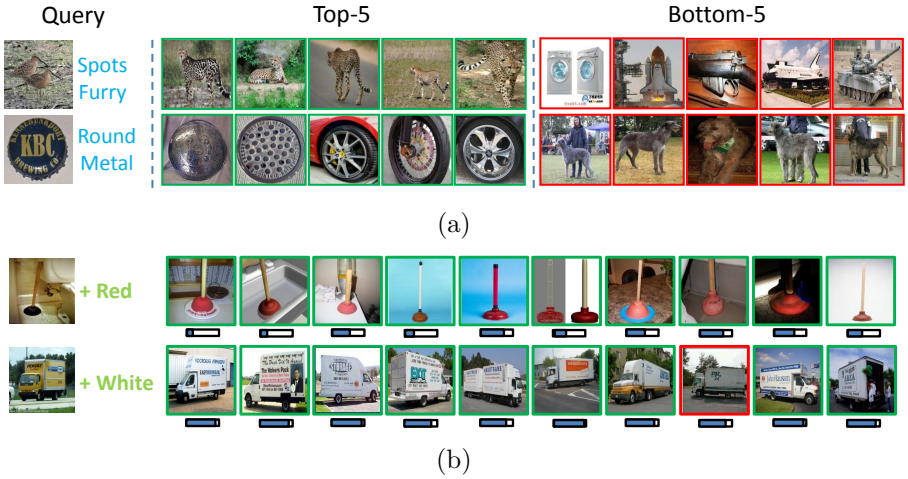
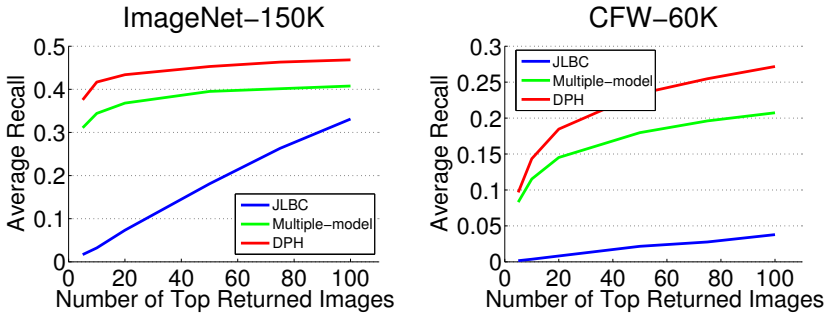


Fig. 4. Some real retrieval cases of the two attribute-oriented tasks on ImageNet-150K. In these tasks, images in the "Test" set were used as queries, and the "Train-Both" set and "Train-Attribute" set were used together as database. (a) The two rows are from task II, and the interested attributes are listed on the right side of the query image. (b) The two rows are from task III, and the top-10 ranked images are displayed. The notations here are consistent with Figure 1(b). This figure is best viewed in color.

only accept fully annotated images as training inputs. We used the same CNN features as described above to train this method. 2) A combination of DLBHC [17] and the CNN-attribute model in Section 4.4 (CNN-attribute for attribute prediction and DLBHC for Hamming distance ranking, which corresponds to training two separate models). The DLBHC model used here was trained to produce $(256 - m)$-bit binary codes, where m is the number of attributes, and the predictions of CNN-attribute were quantized to binary, thus the storage cost of this method is equal to our DPH method. We denote this method as **Multiple-model**.

Evaluation metric: Similar to the previous sections, the query attribute is acquired by attribute predictors, and the performance is evaluated using the ground-truth labels. Only images that match the query image in terms of category and possess the query attribute are considered as relevant. We use $recall@\{5, 10, 20, 50, 75, 100\}$ to evaluate the different methods. In case that the database does not contain any true matches, the recall of such query is simply ignored. We report the average recall over all valid queries.

Results: The results are shown in Figure 5. Our method consistently outperforms the comparative methods. The performance of JLBC on CFW-60K is very unsatisfactory, even though CNN features was used to train this model. This result confirms that our end-to-end framework is necessary for learning dual purpose hash codes. Although each model of the "Multiple-model" method performs quite well on its own task, their combination is clearly outperformed by our DPH method. A possible explanation is that the codes learned by these two models are redundant, while our DPH can suppress the redundancy between category and attributes by exploiting the correlation between them, thus the



(a) (b)

Fig. 5. Comparison of combined retrieval performance (average recall) of our method and other comparative methods on (a) ImageNet-150K and (b) CFW-60K. The results were obtained by 256-bit binary code.

total amount of information they actually carry is less than our dual purpose codes. Moreover, the Multiple-model method needs two networks to produce the binary codes, thus the computation cost is twice as much as our method. We provide some real retrieval results on this task in Figure 4(b). Please refer to the supplementary materials for more results.

4.6 Discussion

To sum up, our DPH method utilized 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 expects that DPH should naturally yield better performances. Indeed, since some attributes often vary significantly even within a single class (e.g. color attributes of towels), the additional attribute information might even make the learning of category more difficult. Even though, the performances of our binary codes on the three retrieval tasks are still satisfactory, while the computation cost of our method is much lower than training multiple models, 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 category 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 can be easily incorporated to further improve the performance of our method.

References

1. Hadi Kiapour, M., Han, X., Lazebnik, S., Berg, A.C., Berg, T.L.: Where to buy it: Matching street clothing photos in online shops. In: Proceedings of the IEEE International Conference on Computer Vision. (2015) 3343–3351
2. Siddiquie, B., Feris, R.S., Davis, L.S.: Image ranking and retrieval based on multi-attribute queries. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE (2011) 801–808
3. Babenko, A., Slesarev, A., Chigorin, A., Lempitsky, V.: Neural codes for image retrieval. In: Computer Vision–ECCV 2014. Springer (2014) 584–599
4. Yu, F.X., Ji, R., Tsai, M.H., Ye, G., Chang, S.F.: Weak attributes for large-scale image retrieval. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2949–2956
5. Escorcia, V., Niebles, J.C., Ghanem, B.: On the relationship between visual attributes and convolutional networks. In: Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on, IEEE (2015) 1256–1264
6. Zhong, Y., Sullivan, J., Li, H.: Face attribute prediction with classification cnn. arXiv preprint arXiv:1602.01827 (2016)
7. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB. Volume 99. (1999) 518–529
8. Weiss, Y., Torralba, A., Fergus, R.: Spectral hashing. In: NIPS. (2008) 1753–1760
9. Kulis, B., Darrell, T.: Learning to hash with binary reconstructive embeddings. In: NIPS. (2009) 1042–1050
10. Wang, J., Kumar, S., Chang, S.F.: Semi-supervised hashing for large-scale search. PAMI **34**(12) (2012) 2393–2406
11. Gong, Y., Lazebnik, S.: Iterative quantization: A procrustean approach to learning binary codes. In: Computer Vision and Pattern Recognition (CVPR), 2011. (2011) 817–824
12. Norouzi, M., Fleet, D.J.: Minimal loss hashing for compact binary codes. In: ICML 2011. (2011) 353–360
13. Liu, W., Wang, J., Ji, R., Jiang, Y.G., Chang, S.F.: Supervised hashing with kernels. In: Computer Vision and Pattern Recognition (CVPR), 2012. (2012) 2074–2081
14. Xia, R., Pan, Y., Lai, H., Liu, C., Yan, S.: Supervised hashing for image retrieval via image representation learning. In: Twenty-Eighth AAAI Conference on Artificial Intelligence. (2014)
15. Lai, H., Pan, Y., Liu, Y., Yan, S.: Simultaneous feature learning and hash coding with deep neural networks. In: Computer Vision and Pattern Recognition (CVPR), 2015. (2015) 3270–3278
16. Zhao, F., Huang, Y., Wang, L., Tan, T.: Deep semantic ranking based hashing for multi-label image retrieval. In: Computer Vision and Pattern Recognition (CVPR), 2015. (2015) 1556–1564
17. Lin, K., Yang, H.F., Hsiao, J.H., Chen, C.S.: Deep learning of binary hash codes for fast image retrieval. In: Computer Vision and Pattern Recognition Workshops (CVPRW), 2015. (2015) 27–35
18. Zhang, R., Lin, L., Zhang, R., Zuo, W., Zhang, L.: Bit-scalable deep hashing with regularized similarity learning for image retrieval and person re-identification. Image Processing **24**(12) (2015) 4766–4779
19. Sadovnik, A., Gallagher, A., Parikh, D., Chen, T.: Spoken attributes: Mixing binary and relative attributes to say the right thing. In: ICCV 2013. (2013) 2160–2167

20. Kovashka, A., Grauman, K.: Attribute adaptation for personalized image search. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 3432–3439
21. Scheirer, W.J., Kumar, N., Belhumeur, P.N., Boult, T.E.: Multi-attribute spaces: Calibration for attribute fusion and similarity search. In: Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, IEEE (2012) 2933–2940
22. Kumar, N., Belhumeur, P., Nayar, S.: Facetracer: A search engine for large collections of images with faces. In: Computer Vision–ECCV 2008. Springer (2008) 340–353
23. Tao, R., Smeulders, A.W.M., Chang, S.F.: Attributes and categories for generic instance search from one example. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 177–186
24. Turakhia, N., Parikh, D.: Attribute dominance: What pops out? In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 1225–1232
25. Rastegari, M., Diba, A., Parikh, D., Farhadi, A.: Multi-attribute queries: To merge or not to merge? In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2013) 3310–3317
26. Parikh, D., Grauman, K.: Interactively building a discriminative vocabulary of nameable attributes. In: Computer Vision and Pattern Recognition (CVPR), 2011. (2011) 1681–1688
27. Rastegari, M., Farhadi, A., Forsyth, D.: Attribute discovery via predictable discriminative binary codes. In: ECCV 2012. Springer (2012) 876–889
28. Li, Y., Wang, R., Liu, H., Jiang, H., Shan, S., Chen, X.: 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. (2015) 3819–3827
29. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS. (2012) 1097–1105
30. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Computer Vision and Pattern Recognition (CVPR), 2015. (2015) 1–9
31. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)
32. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: Imagenet large scale visual recognition challenge. IJCV (2015) 211–252
33. Zhang, X., Zhang, L., Wang, X.J., Shum, H.Y.: Finding celebrities in billions of web images. Multimedia **14**(4) (2012) 995–1007
34. Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., Guadarrama, S., Darrell, T.: Caffe: Convolutional architecture for fast feature embedding. In: Proceedings of the ACM International Conference on Multimedia. (2014) 675–678
35. Yi, D., Lei, Z., Liao, S., Li, S.Z.: Learning face representation from scratch. arXiv preprint arXiv:1411.7923 (2014)
36. `https://en.wikipedia.org/wiki/F1_score`
37. Shen, F., Shen, C., Liu, W., Tao Shen, H.: Supervised discrete hashing. In: Computer Vision and Pattern Recognition (CVPR), 2015. (2015) 37–45

Supplementary Materials: Dual Purpose Hashing

The following sections give details about the attributes defined on ImageNet-150K dataset, and additional real retrieval results. This material is best viewed in color.

1 Example Images of Attributes

In this section, we provide example images of each attribute defined on ImageNet-150K (25 attributes, including color, texture, shape, material, and structure). The attributes were defined and annotated mainly based on the ImageNet-attribute [S1] and Animals with Attribute (AwA) [S2] datasets. Compared to [S1], our dataset covers much more categories (1000 vs 384) and images (50,000 vs 9,600). For each attribute, three positive samples along with three negative samples are shown in Figure 1, 2, and 3 (the leftmost three in each row are positive samples and the rest are negative samples). In our experiments, the attributes are binary, namely, an image either has or does not have the attribute.

2 Real Retrieval Cases

This section gives more real retrieval cases on the attribute-oriented retrieval tasks described in Sections 4.4 and 4.5 of the main paper(the results were obtained with 256-bit binary codes).

2.1 Results on CFW-60K

The results of task II and task III on CFW-60K are shown in Figure 4 and 5 respectively. In task II, the system is required to retrieve images of subjects with the same gender, race, and age group as the subject in the query image. As we can see from the failed cases (Figure 4(b)), for each query image, though the top feedbacks fail to match the exact attributes of the query, all of them have the same gender, race, and age group. By further investigating the failed cases, we found that the main cause is the incorrect attribute predictions of the query image. In task III, either inaccurate attribute prediction or the incapability of binary codes in preserving category similarity would result in failed cases. Here we only show the successful cases to demonstrate the potential of our method in this challenging realistic retrieval scenario.

2.2 Results on ImageNet-150K

The results on ImageNet-150K are shown in Figure 6 and 7. For this dataset, since there are only two images from each category in the "Test" set, to better

evaluate our method for qualitative demonstration, in this supplemental experiment we used the "Test" set as query images, and retrieved images from both "Train-Both" and "Train-Attribute" sets. Some successful retrieval results on task II and task III are provided, suggesting that our method has the potential to be applied in these two realistic yet very challenging object retrieval scenarios.

References

[S1] Russakovsky, O., Fei-Fei, L.: Attribute learning in large-scale datasets. In: ECCV workshop. Springer (2010) 1-14

[S2] Lampert, C.H., Nickisch, H., Harmeling, S.: Learning to detect unseen object classes by between-class attribute transfer. In: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE (2009) 951-958

Fig. 1. Example images of attributes on ImageNet-150K. For each attribute, three positive samples (the leftmost three) and three negative samples (the rightmost three) are shown in this figure.

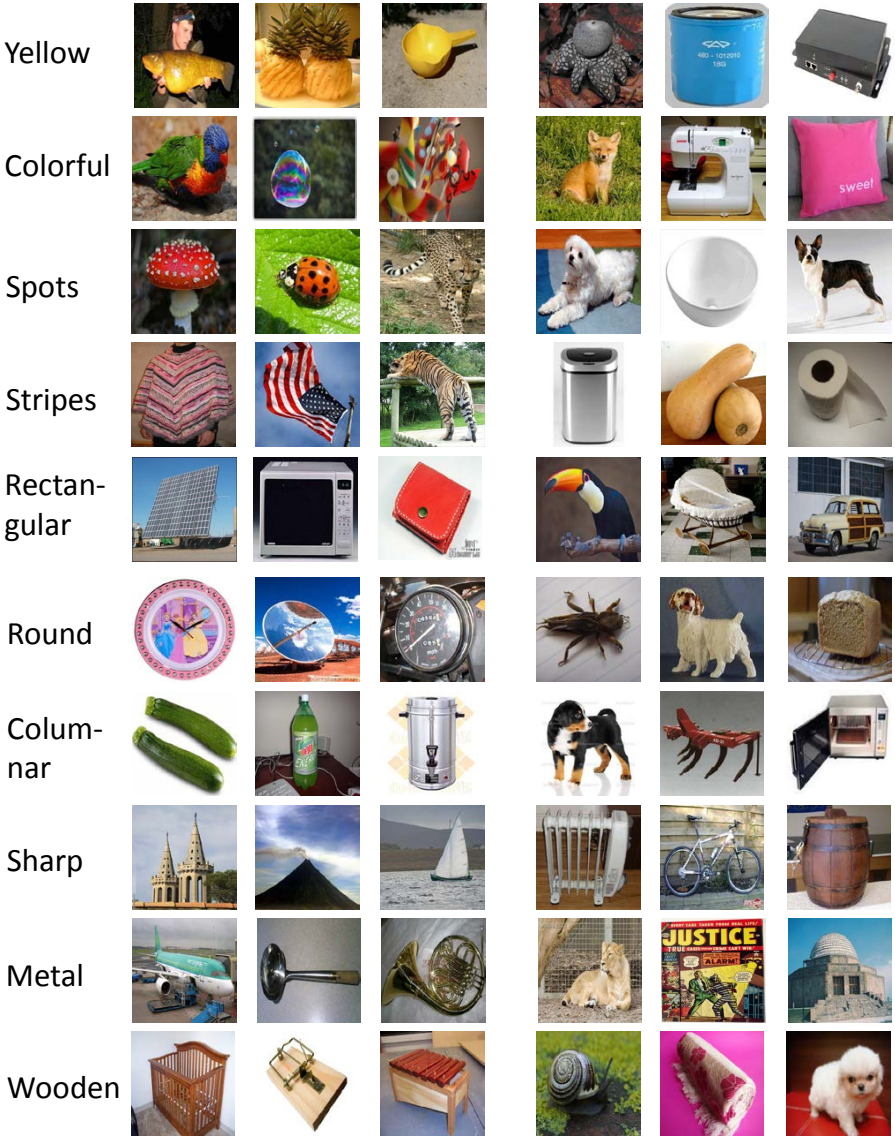


Fig. 2. Example images of attributes on ImageNet-150K. For each attribute, three positive samples (the leftmost three) and three negative samples (the rightmost three) are shown in this figure.

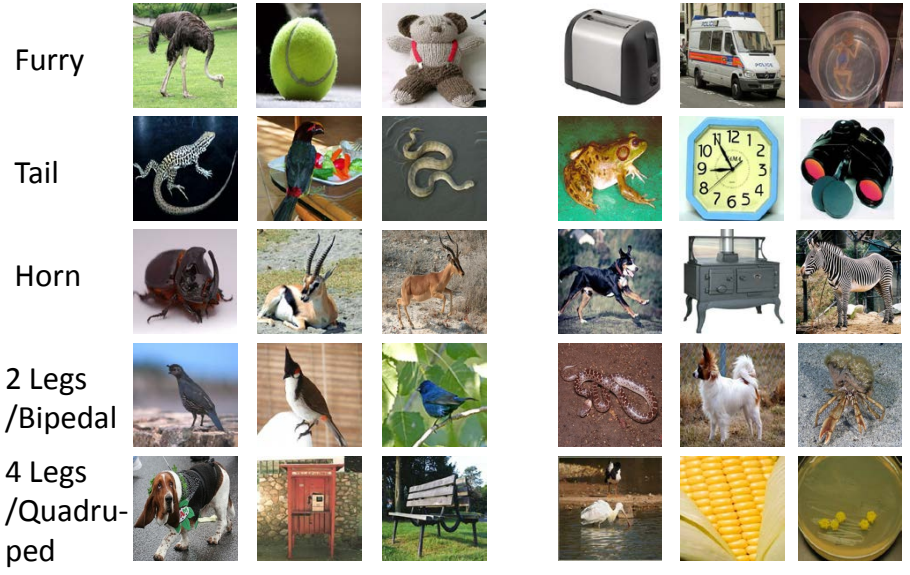


Fig. 3. Example images of attributes on ImageNet-150K. For each attribute, three positive samples (the leftmost three) and three negative samples (the rightmost three) are shown in this figure.

Fig. 4. Some real retrieval results of task II on CFW-60K (retrieving images of subjects with the same gender, race, and age group as the subject in the query image). The results were obtained with 256-bit binary codes. The notations are consistent with the main paper (please refer to Figure 1 in the main paper for details). (a) successful cases, (b) failed cases. In the failed cases, the predicted gender, race, and age group of the 3 query images are: 1) male + white + young (groundtruth: male + white + mid-aged), 2) female + Asian + young (groundtruth: female + white + young), 3) male + Asian + young (groundtruth: male + white + young). Note that as mentioned in our main paper, in this task II the predicted attributes of all images (both query and database images) were used for retrieval, while the evaluation is performed on the ground-truth attribute labels. As a result, both wrong predictions of the query image and the database images would cause a mismatch.

Fig. 5. Some real retrieval results of task III on CFW-60K. The notations are consistent with the main paper (please refer to Figure 1 in the main paper for details).

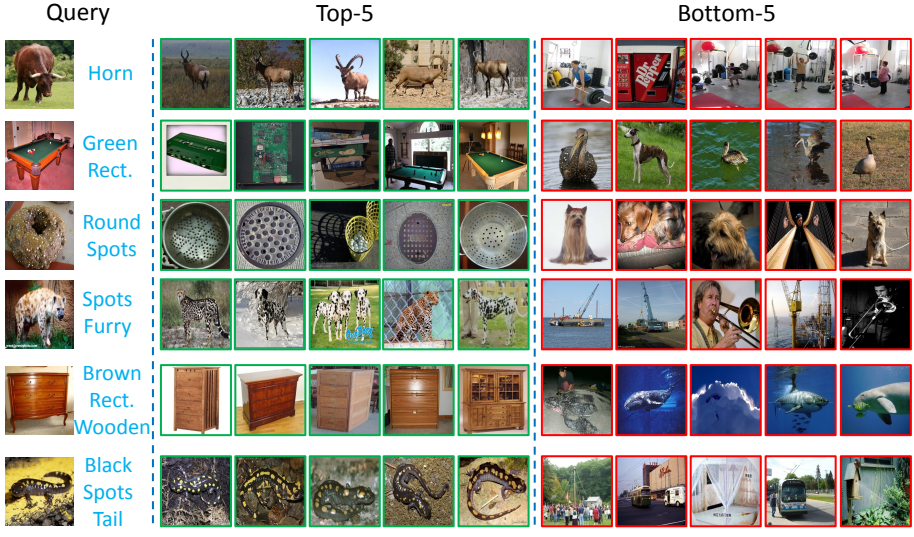


Fig. 6. Some real retrieval results of task II on ImageNet-150K, the interested attributes are listed on the right side of the query images, including color, texture, shape, material, and structure. Images in the "Test" set were used as queries, and the "Train-Both" set and "Train-Attribute" set were used as database. The results were obtained with 256-bit binary codes. The notations are consistent with the main paper (please refer to Figure 1 in the main paper for details).

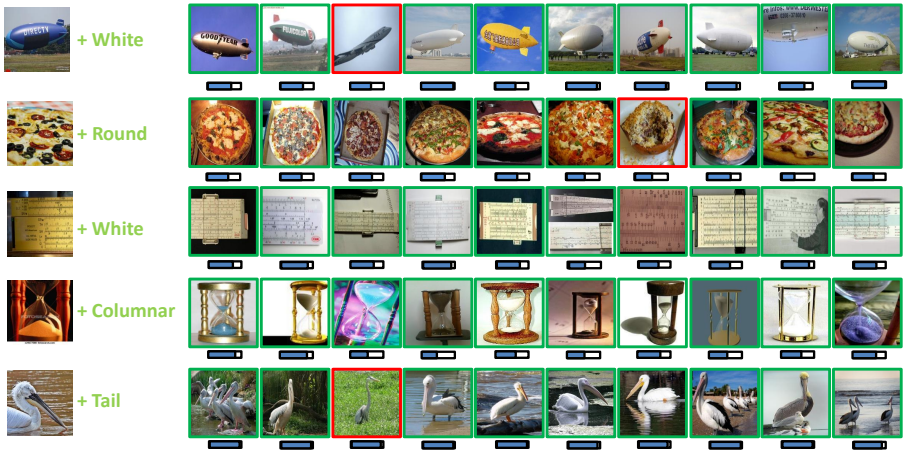


Fig. 7. Some real retrieval results of task III on ImageNet-150K. Images in the "Test" set was used as queries, and the "Train-Both" set and "Train-Attribute" set were used as database. The notations are consistent with the main paper (please refer to Figure 1 in the main paper for details). Note that for each of the second and fifth query, one of the top-10 feedbacks (that is bounded by red box) does not match the query in terms of category (the ground-truth category of the queries are "moving van" and "pelican" respectively, while the ground-truth category of the wrong feedbacks are "trailer truck" and "crane" respectively). We can see that the wrong feedbacks look very similar to the query images, even humans would have some difficulty to perform such a fine-grained categorization task, thus it is reasonable that the retrieval system made such a mistake.