Learning Multifunctional Binary Codes for Both Category and Attribute Oriented Retrieval Tasks

Haomiao Liu[1,2], Ruiping Wang[1,2,3], Shiguang Shan[1,2,3], Xilin Chen[1,2,3]
[1]Key Laboratory 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
[3]Cooperative Medianet Innovation Center, China
haomiao.liu@vipl.ict.ac.cn, {wangruiping, sgshan, xlchen}@ict.ac.cn

Abstract

In this paper we propose 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 incapable of 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 networks (CNN) 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.

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 [5]; **II**. retrieving images with specified attributes [29]; and **III**. the combination of the above tasks, *e.g*. looking for clothing of the same style but with a different color. Existing algorithms [5, 29, 1, 39] can be adopted to tackle the above tasks, and have achieved certain degree of successes. However,

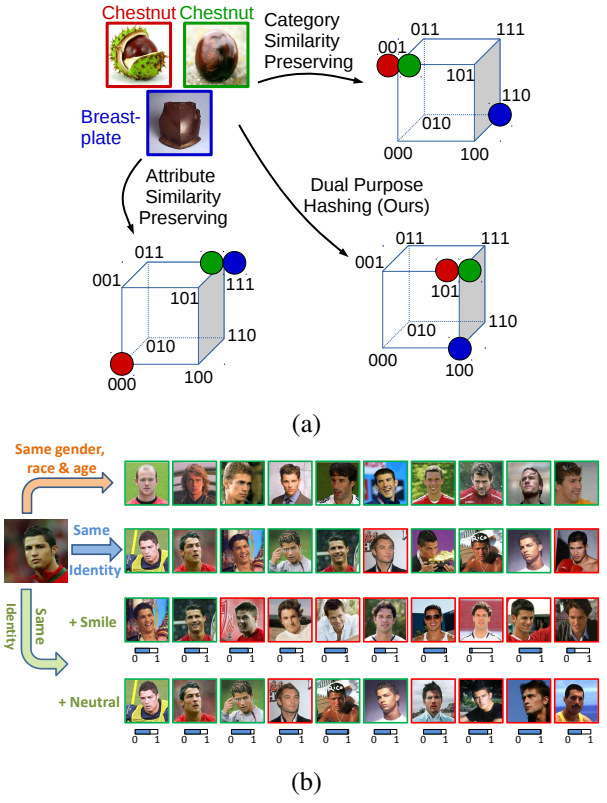


Figure 1. (a) Illustration of the idea of our Dual Purpose Hashing method. (b) A real example showing the three retrieval tasks on a face dataset. The top ranked feedbacks of each task are shown here. In the first two rows, exactly matched images are bounded by green boxes, and red otherwise. In the last two rows, images of the same/different identity are bounded by green/red boxes respectively, and the blue bars indicate the confidence of the corresponding attribute. Best viewed on a computer screen.

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.

A major issue concerning most existing hashing methods is that they are 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. 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 [2, 44], 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. 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 this paper, we aim at learning multifunctional binary codes for multiple image retrieval tasks. Therefore, our work is naturally related to the multi-task learning problem. Specifically, some previous works, *e.g*. [19, 30], have adopted CNN models to simultaneously deal with multiple different tasks, and have achieved some successes. However, our method differs from existing methods in two aspects. First, both [19, 30] learn real-valued features, which does not satisfy the specific requirements of large-scale image retrieval tasks. Second, we elaborately design the loss functions to exploit the huge amount of partially labelled data, which has rarely been considered by previous CNN models.

In large-scale retrieval tasks, hashing [3, 12, 35, 4, 20, 18, 37, 14, 42] is favorable for its low time and space complexity. The pioneering data-independent hashing method Locality Sensitive Hashing (LSH) [3] uses random projections to produce binary bits, and thus LSH usually requires long codes to achieve satisfactory retrieval performance. 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 (semi-)supervised. Unsupervised methods, *e.g*. Spectral Hashing (SH) [36] and Iterative Quantization (ITQ) [4], 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 methods are CCA-ITQ [4], Minimal Loss Hashing (MLH) [20], Semi-Supervised Hashing (SSH) [35], Binary Reconstructive Embedding (BRE) [12], and Supervised Hashing with Kernels (KSH) [18]. 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 [43, 14, 37, 16, 40, 17, 42] significantly improve the state of the arts by jointly learning the image representations and

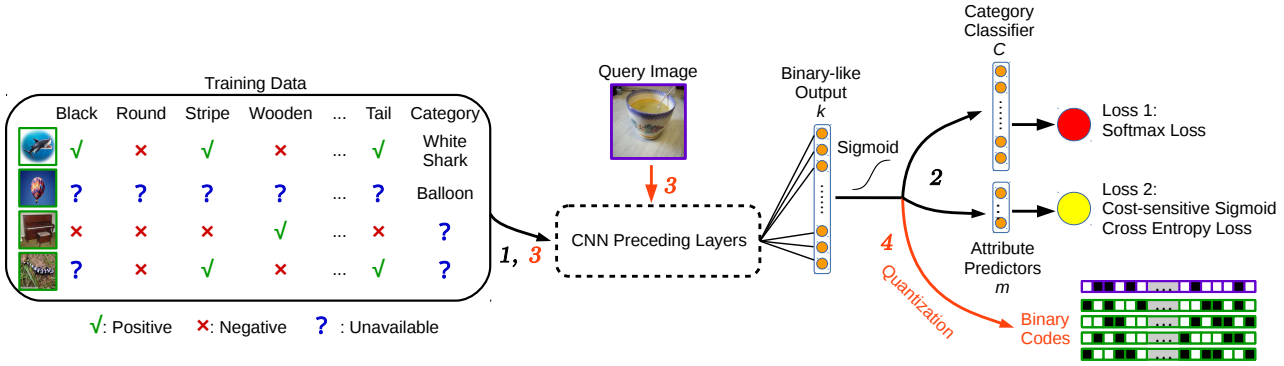


Figure 2. The framework of our DPH method. To simultaneously encode category and visual attributes of images into binary codes, we devise a CNN model that can take partially labelled images as training input (step 1), and train the model on classification and attribute prediction tasks (step 2). The 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).

the hash functions using CNN models.

Other than category-oriented image retrieval, attributes have also been widely adopted in retrieval tasks [25, 10, 26, 13, 32, 33, 22, 39, 29, 7]. Our work is most related to the works that use nameable attributes [21] as queries. [13] 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 [29], fusion strategy [26, 22], relative attributes [25], natural language [7], and other techniques [10, 33] to improve the retrieval performance. In this paper, we adopt the retrieval strategy in [13] for simplicity, while those more complicated ones [29, 26, 22] 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 [23, 15, 8] have made some early attempts to connect attributes with binary codes. [23, 8] discovers 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 [23, 8] unsuitable to be used for attribute-oriented retrieval tasks. [15] improves [23] 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 [15] is 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 [11, 31, 6] 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, $\mathbb{I}\{cond.\}$ 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:

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

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:

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

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 [24] 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** [15] is a subset of the CFW dataset [41] and contains 60,000 images of 500 subjects, among which 20 images of each subject are annotated with 14 attributes. For the images with attribute annotations, 10 images of each subject were used as Test set, and the rest were further divided into two parts (Train-Both and Train-Attribute). The details of partitioning is illustrated in Figure 3(b). Please refer to the original publications [41, 15] for more details about this dataset. On both datasets, the category labels of the Train-Attribute set were made unavailable in the training stage.

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 results of all images. 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: Our datasets are still relatively small in terms of training a deep 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 [9]. 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 [38] (from conv1 to pool5). Since the pre-trained model is not available, we followed the original publication [38] to train the model, except for removing the contrastive loss for simplicity.

For ImageNet-150K, the model was trained for 40 epochs, and 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 newly added layers with a batch size of 200. The momentum and weight decay parameters were set according to the original publications [38, 9]. Besides, on both datasets, we empirically set the weighting parameter $\alpha = 0.1$ in Eqn.(4). All the comparison CNN methods were implemented with Caffe [9] [1].

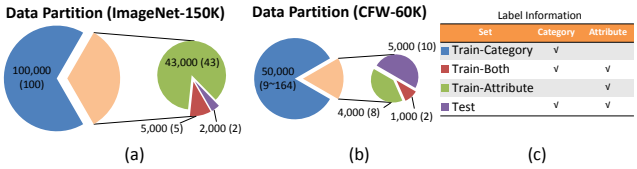


Figure 3. Illustration of data partition in our experiments. (a) ImageNet-150K with 1,000 categories, (b) CFW-60K with 500 categories. The sizes of each set are presented in the figure, and the numbers in the brackets indicate the number of images from each category. (c) The label information of the corresponding sets. Best viewed in color.

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 training sets (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 their Hamming distances to the query image, and the performance is measured by mAP of retrieval, where images from the same category are deemed as relevant. For the attribute part, we report the mean F1-score [34] 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 than accuracy.

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) 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 approached zero while the test loss only decreased slightly. In contrast, when additional training data was introduced, the training loss and test loss of the corresponding tasks were always on the same scale as normally

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

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 all training data can improve the performance on both tasks by a large margin (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 in Section 1, *i.e*. given a query image, retrieving images of the same category. The retrieval is done by ranking the database images according to their Hamming distances to the query image.

Comparative methods: We compare with eight hashing methods: LSH [3], ITQ [4], CCA-ITQ [4], DBC [23], KSH [18], SDH [27], DNNH [14], and DLBHC [16], including representative conventional methods as well as state-of-the-art CNN-based methods.

For fair comparison, the conventional methods were trained using L2-normalized CNN features extracted from the pre-trained models (described in Section 4.1). The comparative methods were implemented using the source codes provided by the authors. As for the CNN-based methods, DLBHC and DNNH exploited the same preceding layers as our DPH method, and were initialized with the identical pre-trained models as ours. Specifically, to make DNNH converge, we randomly sample 10 categories and 20 images per category in each iteration, as in [28], to increase the number of valid triplets in each mini-batch.

All the comparative methods were trained using the combination of "Train-Both" and "Train-Category" sets. 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) of retrieval as metric, where images with the same category label are considered as relevant.

Results: The results are shown in Table 2. We can see that: **First**, when equipped with CNN features, the conventional non-linear method KSH can hardly improve over linear methods. One possible explanation is that the CNN has mapped the images to a feature space where different categories are roughly linearly separable, thus KSH can hardly benefit from the non-linearity of kernel space. **Second**, CNN-based methods significantly improve over conventional 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 methods in learning more suitable representations for the data at hand. **Third**, DNNH performs relatively worse than the other two CNN-based methods [2]. This might be attributed to the batch sampling strategy we used. Thus it seems that the training data should be carefully organized for DNNH when the number of categories is large. **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, *i.e*. category retrieval.

4.4. Evaluation of Attribute Retrieval

Here we test on the second task in Section 1. The attribute prediction scores of DPH can be obtained from the binary codes as described in Section 3.5. In this experiment, given an query image, we randomly select at most three attributes, whose values are specified by the image (thus can be either positive or negative). The system is required to retrieve images that match the selected attributes. 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 the attribute prediction part of retrieval: 1) Similar to [13], we train linear SVMs to predict attributes (we found that the performance of linear and kernel SVMs are almost the same, thus we used linear SVMs for efficiency), using the same CNN features as described in Section 4.3. Then the prediction scores are normalized 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 predict the attributes. For this purpose, we modified our network structure by replacing both the binary-like layer and the classification loss with an attribute prediction loss. We denote this method as **CNN-attribute**. All comparative methods were trained using the combination of "Train-Both" and "Train-Attribute" sets.

[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 [14].

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

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.

	ImageNet-150K					CFW-60K				
	16-bit	32-bit	64-bit	128-bit	256-bit	16-bit	32-bit	64-bit	128-bit	256-bit
LSH [3]	0.032	0.070	0.134	0.215	0.269	0.080	0.110	0.117	0.118	0.118
ITQ [4]	0.102	0.167	0.235	0.284	0.310	0.039	0.058	0.079	0.112	0.135
CCA-ITQ [4]	0.090	0.157	0.223	0.294	0.341	0.048	0.069	0.090	0.113	0.140
DBC [23]	0.207	0.264	0.308	0.344	0.369	0.045	0.060	0.072	0.099	0.129
KSH [18]	0.110	0.181	0.253	0.293	0.320	0.046	0.063	0.086	0.111	0.117
SDH [27]	0.082	0.143	0.222	0.288	0.322	0.026	0.049	0.095	0.140	0.183
DNNH [14]	0.102	0.147	0.213	0.267	0.298	0.035	0.058	0.100	0.148	0.185
DLBHC [16]	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

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.

Evaluation metric: In this task, we report the average retrieval mAP over all valid queries 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, we use the predicted attributes of all images (both query and database) for ranking, while evaluate by the ground-truth annotations. As a result, both wrong predictions of the query image and the database images would hurt the performance.

Results: The results are given in Table 3. On both datasets, the performances of our 256-bit binary codes are comparable to or even better than the baseline methods. However, our method does not need to store the real-valued prediction scores, thus more storage-efficient than SVM-real and CNN-attribute. On the other hand, SVM-binary is as compact as our method, and achieves similar performance with our method on ImageNet-150K, but much worse on CFW-60K. This might be explained by the fact that ImageNet-150K contains more categories and attributes, 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 Table 3(a), we can expect that longer codes of DPH could achieve better performance. A real retrieval result on this task is provided in Figure 4(a). Please refer to the supplementary materials for more examples.

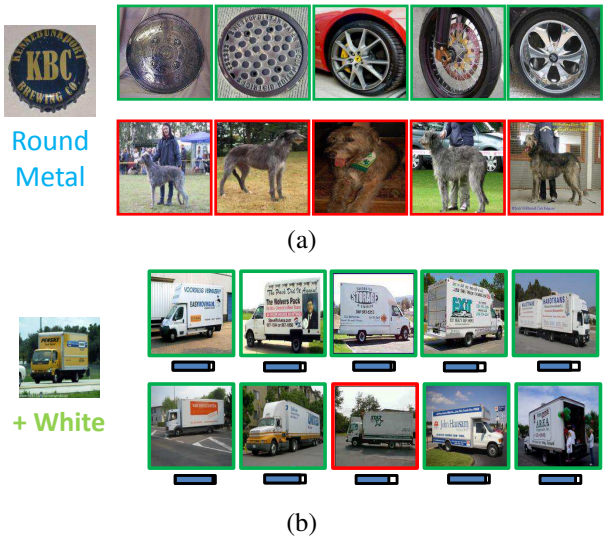


Figure 4. Some real retrieval cases of the two attribute-oriented tasks on ImageNet-150K. Here the "Test" set were used as queries, and the "Train-Both" set and "Train-Attribute" set were used together as database, which is a little different from the evaluation metrics. (a) Results from task II, and the interested attributes are listed below the query image. Top-5 and Bottom-5 feedbacks are shown in the first and second row respectively. (b) Top-10 feedbacks from task III. The notations here are consistent with Figure 1(b). Best viewed in color.

4.5. Evaluation of Combined Retrieval

In this subsection, we evaluate on the third retrieval task in Section 1. Here the system is required to retrieve images belonging to the same category as the query image, while possessing a selected 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 DPH with 256-bit binary codes.

	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

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.

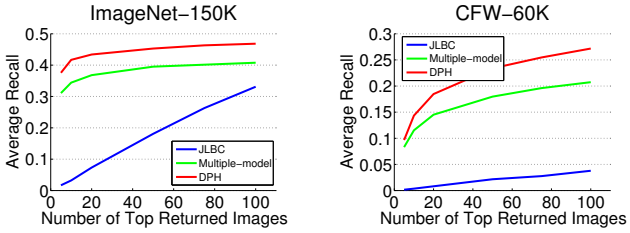
Comparative methods: Since this is a relatively unexplored task, we compare our DPH with two baselines: 1) **JLBC** [15], which is trained on the fully annotated "Train-Both" set with the same CNN features as described above. 2) **Multiple-model**. Here we use CNN-attribute in Section 4.4 for attribute prediction and DLBHC [16] for Hamming distance ranking. The DLBHC model 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 baseline is the same as our DPH method.

Evaluation metric: Only images that match the query image in terms of category and possess the selected attribute are considered as relevant. We use $recall@\{5, 10, 20, 50, 75, 100\}$ to evaluate the comparison 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, which 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 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 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 a real retrieval result 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



(a) (b)

Figure 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.

(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.

Acknowledgements. This work is partially supported by 973 Program under contract No. 2015CB351802, Natural Science Foundation of China under contracts Nos. 61390511, 61379083, and Youth Innovation Promotion Association CAS No. 2015085.

References

[1] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky. Neural codes for image retrieval. In *European Conference on Computer Vision (ECCV), 2014*, pages 584–599. 2014. 1

[2] V. Escorcia, J. C. Niebles, and B. Ghanem. On the relationship between visual attributes and convolutional networks. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 1256–1264, 2015. 2

[3] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Very Large Data Base (VLDB), 1999*, volume 99, pages 518–529, 1999. 2, 6, 7

[4] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011*, pages 817–824, 2011. 2, 6, 7

[5] M. Hadi Kiapour, X. Han, S. Lazebnik, A. C. Berg, and T. L. Berg. Where to buy it: Matching street clothing photos in online shops. In *International Conference on Computer Vision (ICCV), 2015*, pages 3343–3351, 2015. 1

[6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. 3

[7] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2016*, pages 4555–4564, 2016. 3

[8] C. Huang, C. C. Loy, and X. Tang. Unsupervised learning of discriminative attributes and visual representations. In *Computer Vision and Pattern Recognition (CVPR), 2016*, pages 5175–5184, 2016. 3

[9] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *International Conference on Multimedia (MM), 2014*, pages 675–678, 2014. 5

[10] A. Kovashka and K. Grauman. Attribute adaptation for personalized image search. In *International Conference on Computer Vision (ICCV), 2013*, pages 3432–3439, 2013. 3

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS), 2012*, pages 1097–1105, 2012. 3

[12] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Advances in Neural Information Processing Systems (NIPS), 2009*, pages 1042–1050, 2009. 2

[13] N. Kumar, P. Belhumeur, and S. Nayar. Facetracer: A search engine for large collections of images with faces. In *European Conference on Computer Vision (ECCV), 2008*, pages 340–353. 2008. 3, 6

[14] H. Lai, Y. Pan, Y. Liu, and S. Yan. Simultaneous feature learning and hash coding with deep neural networks. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 3270–3278, 2015. 2, 6, 7

[15] Y. Li, R. Wang, H. Liu, H. Jiang, S. Shan, and X. Chen. Two birds, one stone: Jointly learning binary code for large-scale face image retrieval and attributes prediction. In *International Conference on Computer Vision (ICCV), 2015*, pages 3819–3827, 2015. 3, 5, 8

[16] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen. Deep learning of binary hash codes for fast image retrieval. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2015*, pages 27–35, 2015. 2, 6, 7, 8

[17] H. Liu, R. Wang, S. Shan, and X. Chen. Deep supervised hashing for fast image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2016*, pages 2064–2072, 2016. 2

[18] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012*, pages 2074–2081, 2012. 2, 6, 7

[19] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR, 2016*. 2

[20] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *International Conference on Machine Learning (ICML), 2011*, pages 353–360, 2011. 2

[21] D. Parikh and K. Grauman. Interactively building a discriminative vocabulary of nameable attributes. In *Computer Vision and Pattern Recognition (CVPR), 2011*, pages 1681–1688, 2011. 3

[22] M. Rastegari, A. Diba, D. Parikh, and A. Farhadi. Multi-attribute queries: To merge or not to merge? In *Computer Vision and Pattern Recognition (CVPR), 2013*, pages 3310–3317, 2013. 3

[23] M. Rastegari, A. Farhadi, and D. Forsyth. Attribute discovery via predictable discriminative binary codes. In *European Conference on Computer Vision (ECCV), 2012*, pages 876–889, 2012. 3, 6, 7

[24] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, pages 211–252, 2015. 5

[25] A. Sadovnik, A. Gallagher, D. Parikh, and T. Chen. Spoken attributes: Mixing binary and relative attributes to say the right thing. In *International Conference on Computer Vision (ICCV), 2013*, pages 2160–2167, 2013. 3

[26] W. J. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *Computer Vision and Pattern Recognition (CVPR), 2012*, pages 2933–2940, 2012. 3

[27] F. Shen, C. Shen, W. Liu, and H. Tao Shen. Supervised discrete hashing. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 37–45, 2015. 6, 7

[28] L. Shen, Z. Lin, and Q. Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *European Conference on Computer Vision (ECCV), 2016*, pages 467–482, 2016. 6

[29] B. Siddiquie, R. S. Feris, and L. S. Davis. Image ranking and retrieval based on multi-attribute queries. In *Computer Vision and Pattern Recognition (CVPR), 2011*, pages 801–808, 2011. 1, 3

[30] Y. Sun, Y. Chen, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS, 2014*. 2

[31] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 1–9, 2015. 3

[32] R. Tao, A. W. M. Smeulders, and S.-F. Chang. Attributes and categories for generic instance search from one example. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 177–186, 2015. 3

[33] N. Turakhia and D. Parikh. Attribute dominance: What pops out? In *International Conference on Computer Vision (ICCV), 2013*, pages 1225–1232, 2013. 3

[34] https://en.wikipedia.org/wiki/F1_score. 5

[35] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2393–2406, 2012. 2

[36] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS), 2008*, pages 1753–1760, 2008. 2

[37] R. Xia, Y. Pan, H. Lai, C. Liu, and S. Yan. Supervised hashing for image retrieval via image representation learning. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014. 2

[38] D. Yi, Z. Lei, S. Liao, and S. Z. Li. Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*, 2014. 5

[39] F. X. Yu, R. Ji, M.-H. Tsai, G. Ye, and S.-F. Chang. Weak attributes for large-scale image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012*, pages 2949–2956, 2012. 1, 3

[40] R. Zhang, L. Lin, R. Zhang, W. Zuo, and L. Zhang. 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, 2015. 2

[41] X. Zhang, L. Zhang, X.-J. Wang, and H.-Y. Shum. Finding celebrities in billions of web images. *IEEE Transactions on Multimedia*, 14(4):995–1007, 2012. 5

[42] Z. Zhang, Y. Chen, and V. Saligrama. Efficient training of very deep neural networks for supervised hashing. In *Computer Vision and Pattern Recognition (CVPR), 2016*, pages 1487–1495, 2016. 2

[43] F. Zhao, Y. Huang, L. Wang, and T. Tan. Deep semantic ranking based hashing for multi-label image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2015*, pages 1556–1564, 2015. 2

[44] Y. Zhong, J. Sullivan, and H. Li. Face attribute prediction with classification cnn. *arXiv preprint arXiv:1602.01827*, 2016. 2