

Binary Constrained Deep Hashing Network for Image Retrieval without Human Intervention

Thanh-Toan Do, Dang-Khoa Le Tan, Trung Pham, Tuan Hoang, Ngai-Man Cheung

Abstract—Learning compact binary codes for image retrieval problem using deep neural networks has attracted increasing attention recently. However, training deep hashing networks is challenging due to the binary constraints on the hash codes, the similarity preserving properties, and the requirement for a vast amount of labelled images. To the best of our knowledge, none of the existing methods has tackled all of these challenges completely in a unified framework. In this work, we propose a novel end-to-end deep hashing approach, which is trained to produce binary codes directly from image pixels without human intervention. In particular, our main contribution is to propose a novel pairwise loss function, which simultaneously encodes the distances between pairs of binary codes, and the binary quantization error. We propose an efficient parameter learning algorithm for this loss function. In addition, to provide similar/dissimilar images for our pairwise loss function, we exploit 3D models reconstructed from unlabeled images for automatic generation of enormous similar/dissimilar pairs. Extensive experiments on three image retrieval benchmark datasets demonstrate the superior performance of the proposed method.

I. INTRODUCTION

Content-based image retrieval (CBIR) is an important problem in computer vision with vary applications as well as interdisciplinary connections with other subfields. Technically, given a input image treated as a query, the CBIR searches for visually similar images in a database. In the state-of-the-art image retrieval systems, images are represented as high-dimensional feature vectors which later can be searched via classical distance such as the Euclidean or Cosin distance [1], [2], [3], [4], [5], [6], [7].

The deep learning has given great attention to the computer vision community due to its superiority in many vision tasks such as classification, detection, segmentation [8], [9], [10], [11], [12], [13], and image retrieval [14], [15], [16] as well. Using the real-valued features from off-the-shelf networks to represent images for image retrieval task has achieved impressive retrieval results [16], [15]. Few recent researches show that, fine-tuning deep networks for image retrieval task further boosts retrieval performance [14]. However, to fine-tune a deep network, it requires an enormous amount of labelled images which is sometimes difficult to achieve. It is because annotating images with labels or tags requires skilled manpower, and the label of an image is not always well defined.

Despite achieving high retrieval accuracy, directly using of real-valued high dimensional features from deep networks is not applicable for large scale retrieval problem, due to the expensive storage and time-consuming searching. In this paper, we aim to go beyond real-valued representations of images in a deep learning-based image retrieval system. Specifically, we are interested in learning very compact image representations, i.e., mapping images to low-dimensional binary codes.

Due to its fast computation and efficient storage, using binary hash codes to represent images is an attractive approach in large scale vision problems [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28]. However, learning binary codes in deep networks is challenging. This is because one has to deal with the binary constraint on the hash codes, i.e., the final network outputs must be binary. A naive solution is to adopt the *sign* activation layer to produce binary codes. However, due to the non-smoothness of the *sign* function, it causes the *vanishing gradient* problem when training the network with the standard back propagation [29]. Another important requirement of hashing is the similarity preservation, i.e., similar/dissimilar images should have similar/dissimilar binary codes. To achieve this requirement under deep models, previous deep hashing methods require vast amounts of manually well-defined labelled datasets to supervise the training or fine-tuning, i.e., images with the same semantic (e.g., category / class) label are encouraged to have similar hash codes. However, such large labelled datasets are not always available, especially in some problems which are not directly based on classification such as image retrieval — which is the considered problem in this work.

In this paper, we aim to address the above challenges in learning deep neural networks to produce binary hash codes directly from images without human annotation. In particular, we propose to use a pairwise loss function with binary constraints to model the relative similarities between pairs of hash codes, and explicitly force the network output to be binary values. To tackle this binary-constrained optimization problem, our work takes a principal approach that uses penalty method and alternating optimization. Our approach is fundamentally different from existing works which relax the binary constraints with the range constraints in a rather heuristic manner [20], [21], [28], [30], [26].

Since our loss function is pairwise, it only requires relative relationship for pairs of images, i.e., matching and non-matching images. Clearly such relationship can be obtained without resorting semantic labels. Inspired by the recent works that mine match and non-match images without human intervention [14], [31], our work exploits 3D models built

Thanh-Toan Do and Trung-Pham are with the University of Adelaide, Australia. Email: {thanh-toan.do, trung.pham}@adelaide.edu.au

Dang-Khoa Le Tan, Tuan Hoang, Ngai-Man Cheung are with Singapore University of Technology and Design. Email: {letandang_khoa, tuan_hoang, ngaiman_cheung}@sutd.edu.sg

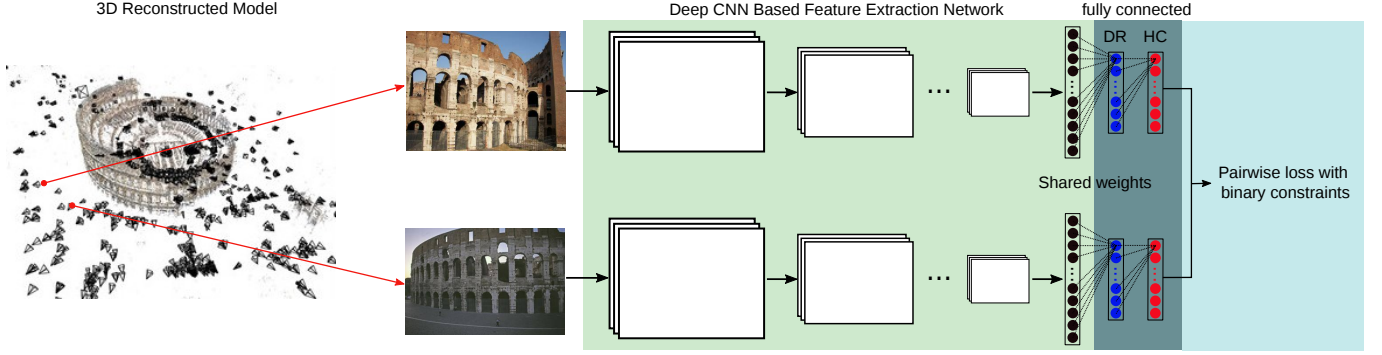


Fig. 1. The overview of the proposed deep hashing. Training data is generated automatically by exploiting 3D reconstructed models. The network architecture for learning binary codes comprises of four components: (i) convolutional layers which is followed by a MAC layer (black layer) for extracting image representations; (ii) a fully connected layer for Dimensional Reduction (blue layer); (iii) a Hash Code mapping layer (red layer); (iv) a novel pairwise loss function with binary constraints.

from unlabelled images using SfM. For example, images capturing the same scene are considered matched, otherwise non-matched. As a result, the training of our hashing network can be done completely and automatically by using unlabelled image as input. Our deep hashing architecture is illustrated in Figure 1.

In summary, we make the following contributions. 1) We propose a novel end-to-end deep learning framework for learning compact binary codes in which the input for training the framework is only unlabelled images. We leverage the 3D reconstructed models to automatically create the training data. 2) We propose a novel pairwise loss function for training deep hashing networks. We reformulate the problem to simultaneously encode distances between hash codes, and binary quantization error. We also develop an alternating optimisation strategy to optimize these variables along with the network parameters. 3) We perform extensive experiments on image retrieval task to demonstrate the superior improvement of our hash codes over the binary codes obtained from the state-of-the-art methods.

II. RELATED WORK

In this section, we review previous works related to the context of our work. Those include (i) end-to-end deep hashing and (ii) end-to-end image retrieval with unsupervised fine-tuning.

End-to-end deep hashing: Joint learning image representations and binary hash codes in an end-to-end deep learning-based supervised hashing framework has shown considerable improvements in various computer vision tasks [20], [21], [28], [26], [22], [32]. By joint optimization, the produced hash codes are more sufficient to preserve the semantic similarity between images. In those works, the network architectures often consist of a feature extraction sub-network and a subsequent hashing layer to produce hash codes. Ideally, the hashing layer should adopt a *sign* activation function to output exactly binary codes. However, due to the vanishing gradient difficulty of the *sign* function, an approximation procedure must be employed. For example, *sign* can be approximated by a tanh-like function $y = \tanh(\beta x)$, where β is a free parameter controlling the trade off between the smoothness and the binary quantization loss [28]. However, it is non-trivial to determine an optimal

β . A small β causes large binary quantization loss while a big β makes the output of the function close to the binary values, but the gradient of the function almost vanishes, making back-propagation infeasible. The problem still remains when the logistic-like functions [20], [21], [26], [32] are used. In [22], the absolute function and l_1 regularization are used for dealing with the binary constraint on the codes. However, both absolute function and l_1 regularization are non-differentiable. The authors worked around this difficulty by assuming that both of them are differentiable everywhere, but there may be some performance degradation. Another drawback of above mentioned supervised deep hashing networks is the requirement for a large amount of semantic annotated data which will be used for encoding the semantic similarities in the loss function. However, such large annotated data is usually unavailable in large scale vision problems such as visual search.

Recently, in [27], the authors proposed DeepBit which is an unsupervised end-to-end hashing, i.e., no semantic (class / category) label information is required. Starting with a pre-trained network (i.e., VGG [9]), the softmax layer of VGG is replaced by a hash layer. Their loss function enforces several criteria on the codes produced by the hash layer, i.e., the output codes should: minimize the quantization loss; be distributed evenly; be invariant to rotation. The authors assumed that the fully connected features produced by the pre-trained network are already discriminative enough for image retrieval task. Hence, no similarity preserving criterion is considered on the hash codes. However, this assumption may not hold. As showed in [15], [31], [14], because the pre-trained models, e.g. VGG [9], AlexNet [8], are specialized for image classification, for new tasks such as image retrieval, fine-tuning for those pre-trained networks is necessary to enhance the discrimination of deep features and so improve the performance.

Unsupervised fine-tuning: In the last few years, image retrieval has witnessed an increasing of performance due to better image representations, i.e., features obtained from pre-trained CNN models, which are trained on image classification task, are recently adopted. For example, [33], [16] used convolutional activations, [34] used fully connected activations. Fine-tuning the networks has also shown further improvements [15]. However, fine-tuning the networks requires

the availability of annotated data. This causes difficulty for unsupervised fine-tuning. Recently, in [14], in order to prepare the data to fine-tune the network for the place recognition task, the authors used a weakly supervised approach, in which they used Google Street View Time Machine for getting GPS-tagged panoramic images taken at nearby spatial locations on the map. Two images taken far from each other are considered as non-matching, while the matching images are selected by the most similar nearby images.

In [31], the authors made further improvements over [14] showing how matching and non-matching pairs can be discovered in a totally unsupervised way. Using a large amount of images downloaded from Flickr with keywords of popular landmarks and cities, they applied Structure from Motion [35] for building multiple 3D models. Images belonging to the same 3D model and sharing enough 3D points are considered as matching, while images belonging to different 3D models are considered as non-matching.

III. BINARY CONSTRAINED DEEP NETWORK WITHOUT HUMAN INTERVENTION

Figure 1 presents the overall proposed pipeline which trains a deep network for learning binary codes without human intervention. In the following, we describe the proposed framework including: the network architecture, the automation of training data generation, the pairwise loss function, and finally the optimization process for learning the network parameters.

A. Network architecture

The proposed network (Figure 1) comprises of four components: (i) a feature extraction component for extracting image representations; (ii) a fully connected layer for reducing dimension of the image features (Dimensionality Reduction – DR layer). The number of units of this layer equals to the code length required to represent each input image; (iii) a fully connected layer mapping the reduced real-valued features to binary values (Hash Code – HC layer); (iv) a pairwise loss function which acts on the output of HC layer.

Note that the choice of the feature extraction component is flexible in our framework. It can be the standard convolutional neural network architectures such as AlexNet [8] or VGG [9], in which outputs of their last fully connected layer are used as inputs of DR layer. Alternatively, the recent architecture which replaces the fully layers of VGG or AlexNet by a Maximum Activations of Convolutions (MAC) layer [33] can also be used. In this case, the MAC feature will be used as the input for DR layer. For the image retrieval task focused in this paper, we adopt the VGG network with a MAC layer as we empirically find that using MAC layer gives better performance than fully connected layer. The MAC layer can be profiled as follows: Given an input image, the output of convolutional layers is a 3D tensor $W \times H \times K$, where K is the number of output feature maps which equals to 512, and $W \times H$ is spatial size of the last convolutional layer. Let \mathcal{X}_k be k^{th} feature map. The MAC image representation is constructed by

$$\mathbf{u} = [u_1, \dots, u_k, \dots, u_K]^T, \text{ where } u_k = \max_{x \in \mathcal{X}_k} x \quad (1)$$

It is worth noting that although the unsupervised fine-tuning and pairwise loss for learning deep network have been used in [31], our work differs from [31] at following important aspects: Firstly, while their target is to learn mid-dimensional feature vectors (i.e., 512 dimensions), our work aims to learn compact binary codes. To this end, we have two additional layers, dimensionality reduction (DR) and hash code (HC), which play the role for producing compact binary codes. Secondly, our pairwise loss (detailed in Section III-C) has binary constraints, unlike constraint-free loss as [31]. The new layers and binary constrained loss function are crucial for hashing, i.e., it ensures the model to produce compact binary codes. Thirdly, we propose a parameter learning scheme to cope with a loss function with binary constraints (detailed in Section III-D). This contrasts to [31] which simply uses standard back-propagation to train the network.

B. Training data

The training input for our network is pairs of matching / non-matching images. One way to achieve training pairs is to access to semantic (class) labelled images so that we could train a hash function which returns similar hash codes for images with the same label, or vice versa. Unfortunately, such labelled dataset is not always available, especially for some problems which are not directly based on classification such as visual search.

Inspired by [31], we generate a training set of matching and non-matching image pairs automatically by exploiting 3D reconstructed models. In particular, we make use of the 3D models given by [31], in which there are 713 3D models reconstructed from images downloaded from Flickr. Most of reconstructed models are popular landmarks and cities. The authors released 3D models and 30K images which are subset of images they used to build models. 5,974 and 1,691 images are selected as training queries and validation queries, respectively.

We sample matching images as follows. Given a query image and its 3D model membership, we extract images that belong to the same 3D model and co-observe enough 3D points (i.e., $\frac{\mathcal{P}(i) \cap \mathcal{P}(q)}{\mathcal{P}(q)} \geq \tau$, where $\mathcal{P}(i)$ is the set of 3D points observed by image i ; τ is a threshold). Among these, one image is randomly sampled and used as the matching image for the query. The value of τ is set to 0.2 in our experiments. The set of matching pairs is kept during the training. This matching pair generation is similar to [31].

For non-matching training pairs, we perform two stages of generation. The offline stage generates pairs for training the network at first iterations. After a certain iterations, we use the current network to perform the online non-matching pair generation (i.e., regenerating non-matching pairs) and using new pairs to continue the training. In particular, the offline generation is performed as follows: Given a query image, we select top k “nearest” images from 3D models different than the model containing the query. The distance between images is computed by using feature vectors extracted from the pre-trained network [31]. Among these k non-matching images, we randomly sample m images (with at most one image per

model) as the non-matching ones for the query. Note that our non-matching mining strategy is different from [31], in which the authors selected m top nearest ones (i.e., hardest negative). We empirically find that randomly selecting hard negative images gives better results than selecting the hardest ones. This is consistent with the observation in [10], [36], i.e., selecting the hardest negative can in practice lead to bad local minima in training. After a certain of iterations, we perform the online non-matching pair generation. The online generation is similar to the offline ones, except that the distance between images is computed by using the binary codes generated by the hash code layer of the current network. The values of k and m are empirically set to 70 and 6 in our experiments.

C. Pairwise loss

Given the training image pairs, we aim to learn the network which not only produces binary codes but also ensures the discrimination of the codes, i.e., matching images should likely have similar binary codes, or vice versa. As the Hamming distance between two vectors of binary codes is one-to-one corresponding to Euclidean distance, we propose to minimize the following binary constrained loss function which acts on the pairs

$$\begin{aligned} \min_{\mathbf{W}} \mathcal{L}(i, j) &= y_{ij} \|\mathbf{f}_i - \mathbf{f}_j\|_2 + (1 - y_{ij}) \max(0, c - \|\mathbf{f}_i - \mathbf{f}_j\|_2) \\ \text{s.t. } \mathbf{f}_i, \mathbf{f}_j &\in \{-1, 1\}^L \end{aligned} \quad (2)$$

where \mathbf{W} is the network parameters; \mathbf{f}_i and \mathbf{f}_j are output hash codes for input images i and j ; label $y_{ij} \in \{0, 1\}$ indicates that the image pair i, j is matching ($y_{ij} = 1$) or non-matching ($y_{ij} = 0$); L is the code length; c is a constant.

It is worth noting that without the binary constraint, the loss function becomes a hinge embedding loss [37]. Technically, the loss function will encourage matching pairs to have close hash codes, and non-matching pairs to have distant hash codes. When a non-matching pair has a large enough distance, i.e. $\geq c$, it is not be taken into account in the loss. The constraint (3) is to ensure the network output is binary.

Optimizing the loss function (2) with the binary constraint (3) using stochastic gradient method is difficult since the constraints are not differentiable. In order to overcome this challenge, we utilize the idea of the penalty method [38]. As will be discussed, this leads to a formulation which avoids solving the exact binary constraint but instead minimizes the binary quantization loss. This makes sense because when the binary quantization loss approaches zero, the binary constraints are automatically (approximately) satisfied. Specifically, we introduce a new auxiliary binary variables $\mathbf{B} = \{\mathbf{b}_i\}_{i=1}^N \in \{-1, 1\}^{L \times N}$ where N is number of current training images, and minimize the following loss function

$$\begin{aligned} \min_{\mathbf{W}, \mathbf{B}} \mathcal{L}(i, j) &= y_{ij} \|\mathbf{f}_i - \mathbf{f}_j\|_2 + (1 - y_{ij}) \max(0, c - \|\mathbf{f}_i - \mathbf{f}_j\|_2) \\ &\quad + \alpha (\|\mathbf{f}_i - \mathbf{b}_i\|_2 + \|\mathbf{f}_j - \mathbf{b}_j\|_2) \\ \text{s.t. } \mathbf{b}_i, \mathbf{b}_j &\in \{-1, 1\}^L \end{aligned} \quad (4)$$

where α is a weighting parameter. The third term of (4) forces the output of Hash Code layer (i.e., $\mathbf{f}_i, \mathbf{f}_j$) as close to binary

values as possible, i.e., it minimizes the binary quantization loss. Although the new loss function still contains constraints, variables \mathbf{B} and \mathbf{W} are decoupled. This allows us to apply alternative optimization over these variables. We will show shortly that \mathbf{W} now can be optimised using the stochastic gradient decent method and \mathbf{B} has a closed-form solution.

D. Parameter learning

In order to minimize the loss function (4) under the constraint (5), we propose an alternating optimization approach, i.e., we learn each variable (network parameter \mathbf{W} or \mathbf{B}) at a time while holding the other fixed.

Fix \mathbf{W} , solve \mathbf{B} : Let $\mathbf{F} = \{\mathbf{f}_i\}_{i=1}^N$, if \mathbf{W} is fixed, the optimal solution for \mathbf{B} is $\text{sgn}(\mathbf{F})$.

Fix \mathbf{B} , solve \mathbf{W} : When \mathbf{B} is fixed, the binary constraint (5) can be ignored, thus, the network parameters \mathbf{W} can be optimized by minimizing the loss (4) using the standard back-propagation. A number of epochs is run until \mathbf{W} converges to local minima before switching to \mathbf{B} .

The whole learning process is summarized in the Algorithm 1. In that Algorithm, $\mathbf{W}_{(t)}$, $\mathbf{F}_{(t)}$, $\mathbf{B}_{(t)}$ are values of $\mathbf{W}, \mathbf{F}, \mathbf{B}$ at t^{th} iteration. We implement the proposed approach using the MatConvNet toolbox [39]. At begining (line 2 in Algorithm 1), we initialize our feature extraction component with the pre-trained MAC network [31] in which its loss layer is removed. The DR layer is initialized by using PCA weights on the training dataset (with pre-trained MAC features). The HC layer is initialized by a random orthogonal matrix.

During an iteration k , we fix the training pairs and alternative solving \mathbf{B} and \mathbf{W} . When fixing \mathbf{B} and learning \mathbf{W} (line 7 in Algorithm 1), we train the network with a fix number of epochs np . The values of K , T and np are set to 3, 5 and 10, respectively. The values of c and α in (4) are set to $\frac{L}{2}$ and 1, respectively. In an iteration k , for each query, we generate $m = 6$ non-matching pairs (line 9 in Algorithm 1). The mini-batch size when learning \mathbf{W} is 28 pairs (i.e. 4 query images, each provides 1 matching pair and 6 non-matching pairs). The best network is selected based on mean Average Precision (mAP) on the validation set.

IV. EXPERIMENTS

In order to evaluate the proposed method, named P2B (Pixels to Binary codes), we conduct extensive image retrieval experiments on standard image retrieval benchmarks.

A. Dataset and evaluation protocol

We conduct experiments on Holidays [3], Oxford5k [40] and Oxford105k [40] datasets which are widely used in evaluating image retrieval systems [6], [1], [5].

Holidays The Holidays dataset consists of 1,491 images of different locations and objects, 500 of them being used as queries. Although the dataset includes a variety of scene types, most of images in dataset are natural scenes.

Oxford5k The Oxford5k dataset consists of 5,063 images of buildings and 55 query images corresponding to 11 distinct buildings in Oxford.

Algorithm 1 P2B Deep Hashing Learning

Input:

reconstructed 3D models and their images; L : code length; K : number of online non-matching pairs generation; T : number of iterations for training network, given a fixed set of matching / non-matching pairs.

Output:

Set of network parameters \mathbf{W} .

```

1: Offline generate matching and non-matching pairs using pre-
   trained MAC network
2: Initialize the network  $\mathbf{W}_{(0)}$ 
3: for  $k = 1 \rightarrow K$  do
4:   for  $t = 1 \rightarrow T$  do
5:     Compute  $\mathbf{F}_{(t-1)}$  using  $\mathbf{W}_{(t-1)}$ 
6:     Compute  $\mathbf{B}_{(t)} = \text{sgn}(\mathbf{F}_{(t-1)})$ 
7:     Fix  $\mathbf{B}_{(t)}$ , optimize  $\mathbf{W}_{(t)}$  (using  $\mathbf{W}_{(t-1)}$  as initialization)
       using back-propagation. Save  $\mathbf{W}_{(t)}$ .
8:   end for
9:   Regenerate non-matching pairs using  $\mathbf{W}_{(T)}$ 
10:  Reinitialize  $\mathbf{W}_{(0)} = \mathbf{W}_{(T)}$ 
11: end for
  
```

Oxford105k In order to evaluate the proposed method at larger scale, we merge Oxford5k dataset with 100k distracted images downloaded from Flickr [40], forming the Oxford105k dataset.

The ground truth of queries have been provided with the datasets. Following [17], [27], we evaluate the performance of methods at compact code lengths, i.e. 8, 16, and 32 bits, with mAP.

B. Compared methods

We compare the proposed approach P2B against several state-of-the-art unsupervised hashing methods, i.e. Iterative Quantization (ITQ) [19], Binary Autoencoder (BA) [17], Spherical Hashing (SPH) [41], K-means Hashing (KMH) [24], DeepBit [27].

As we focus on image retrieval, our primary comparisons are unsupervised hashing methods. Nevertheless, for the sake of completeness and demonstration of the performance, we also conduct comparisons with supervised hashing methods. However, it is important to note that supervised hashing methods are not suitable for the image retrieval task. The reason is that image retrieval task is different from the classification, i.e., the label information in the CBIR system is not well-defined in general. In fact, as pointed out by recent work [42], if the label is well-defined for a specific retrieval, a simple classifier can achieve better performance than state-of-the-art supervised hashing methods, even with shorter code lengths. To the best of our knowledge, none of supervised hashing methods has been thoroughly evaluated on standard image retrieval datasets (i.e. Holidays, Oxford5K, and Oxford105K).

Different from supervised hashing, unsupervised hashing is much more appropriate for image retrieval problem because they do not require any manually labelled images for training. However, in this section, we also give comparisons with supervised hashing methods (Seciton IV-D) for reference purposes. We compare P2B with Kernel-based Supervised Hashing (KSH) [43], Binary Reconstructive Embedding (BRE) [44],

	Holidays		Oxford5k	
L	16	32	16	32
DeepBit (pretrained)	12.0	21.4	10.4	13.7
DeepBit (fine-tuned)	14.1	23.2	13.2	17.4
P2B	18.7	31.0	25.5	41.2

TABLE I
MAP RESULTS OF P2B AND DEEPBIT [27] ON HOLIDAYS AND OXFORD5K DATASETS.

ITQ-CCA [19], Supervised Semantics-preserving Deep Hashing (SSDH) [26].

C. Comparison with unsupervised hashing methods

All compared unsupervised methods (except DeepBit) require image features as input, instead of raw images. To make a fair comparison, we use the pre-trained network [31] to extract the MAC features of 30K images used to create 3D models and use them as input for non deep-based compared methods.

Figure 2 shows the comparative retrieval results in term of mAP. On the Oxford5k and Oxford105k datasets, the results clearly show that the proposed P2B significantly outperforms other methods, i.e., P2B outperforms the most competitive ITQ [19] $\geq 4.5\%$ mAP at all code lengths. The improvements are clearer at the lower code lengths, i.e., at $L = 8$, P2B outperforms ITQ 5.6% and 6.7% mAP on Oxford5k and Oxford105k, respectively. The superior results of P2B on Oxford5k and Oxford105k may be because the Oxford5k building dataset may share similar visual characteristic with the training dataset which is mostly created from 3D models of landmarks.

On the Holidays dataset which mostly contains natural images, the proposed P2B also outperforms ITQ [19]. Note that the proposed P2B and ITQ have same complexity when computing binary codes. Given the image representation, ITQ first applies PCA for dimensionality reduction and then a rotation to rotate real-valued codes to binary codes. These two operations correspond to our *DR* and *HC* layers.

Comparison with DeepBit [27]:

Here we compare P2B with DeepBit [27], which is a deep learning-based unsupervised hashing method. We report the results of DeepBit with two settings. In the first setting, we use their released pre-trained models. In the second setting, we fine-tune these released models using 30K images used to construct 3D models. The comparative results between two methods are presented in Table I. The results show that although the fine-tuning helps DeepBit to boost its performance, the proposed P2B significantly outperforms fine-tuned DeepBit on both datasets.

Figure 3 shows top 10 retrieved images of P2B and DeepBit in response to example queries from Oxford5k and Holidays datasets. As can be seen, P2B returns much more relevant results than DeepBit. Interestingly, the misretrieved images returned by P2B are still visually similar to the queries although they are from different scenes. For example, in the Oxford5k case, both the query and the misretrieved image have repetitive small windows. In the Holidays case, the misretrieved images and the query capture a small lake surrounded by mountains.

Visualization of hash codes of P2B and DeepBit:

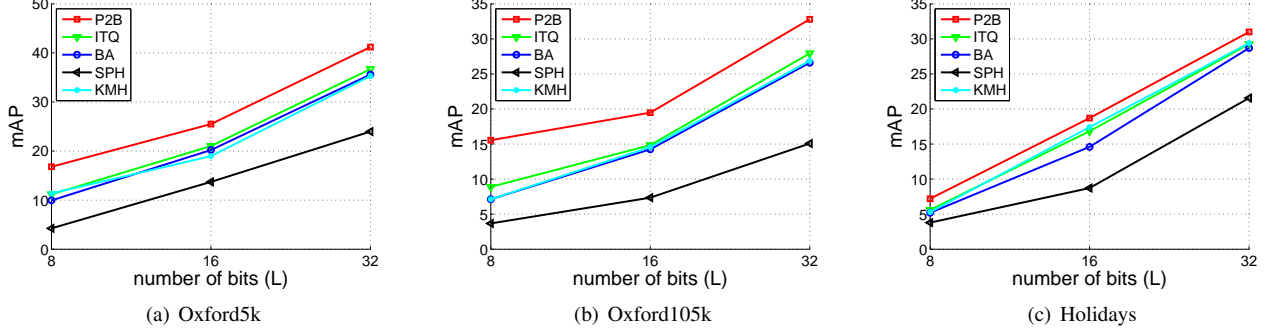


Fig. 2. mAP comparison between the proposed P2B and the state-of-the-art unsupervised hashing methods on Oxford5k, Oxford105k, and Holidays datasets.











































Query	Top 10 Retrieved Images										
<div>Oxford5k</div> 											P2B
	✓	✓	✓	✓	✓	✓	✓	✓	✓	✗	
											DeepBit
	✗	✗	✓	✓	✗	✓	✓	✓	✓	✗	
<div>Holidays</div> 											P2B
	✓	✓	✓	✓	✗	✓	✓	✓	✗	✗	
											DeepBit
	✓	✓	✓	✗	✗	✗	✗	✓	✗	✗	

Fig. 3. Top 10 retrieved images of the proposed P2B and DeepBit[27] in response to example queries on Oxford5k and Holidays datasets.

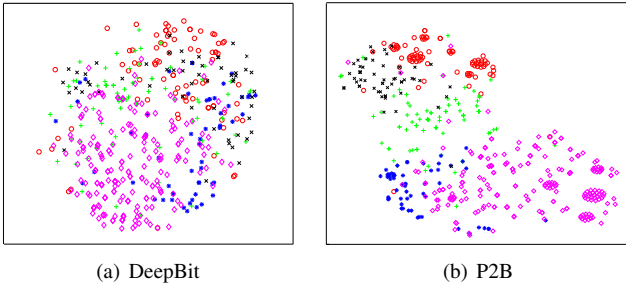


Fig. 4. t-SNE visualization of hash codes generated by the proposed P2B and DeepBit [27] on 5 classes of Oxford5k dataset. The points with same color are same class. The code length is $L = 16$.

We utilize t-SNE [45] to visualize the hash codes generated by P2B and DeepBit on the Oxford5k dataset (only 5 classes with largest numbers of samples are picked). Figure 4 clearly shows that the hash codes generated by P2B are more separated than the ones generated by DeepBit. This suggests that the proposed method can learn more discriminative hash codes than DeepBit.

D. Comparison with supervised hashing methods

To create the training data for supervised hashing methods from the automatically-created 3D models, we follow the following steps. As the number of images for each 3D model varies, i.e., the largest model contains 80 images, while the smallest model contains only 23 images, we select top 100 biggest 3D models and randomly select 60 images per model for training. Note that as KSH and BRE use the full similarity matrix (let be S) when training, it is difficult for these methods to handle larger training data. We try two approaches to define the similarity for each image pair in the similarity matrix S . In the first approach, we check every pair in the matrix S and use the same matching pair generation strategy in Section III-B for determining matching pair. A pair (i, j) is matched, i.e. $S(i, j) = 1$, if both images belong to the same 3D model and co-observe enough 3D points, otherwise $S(i, j) = 0$. By using this first approach, we find that on the average, each image has only ~ 1 matching image. In the other words, the matrix S is very sparse. In the second approach, images belonged to same model are considered as matching,

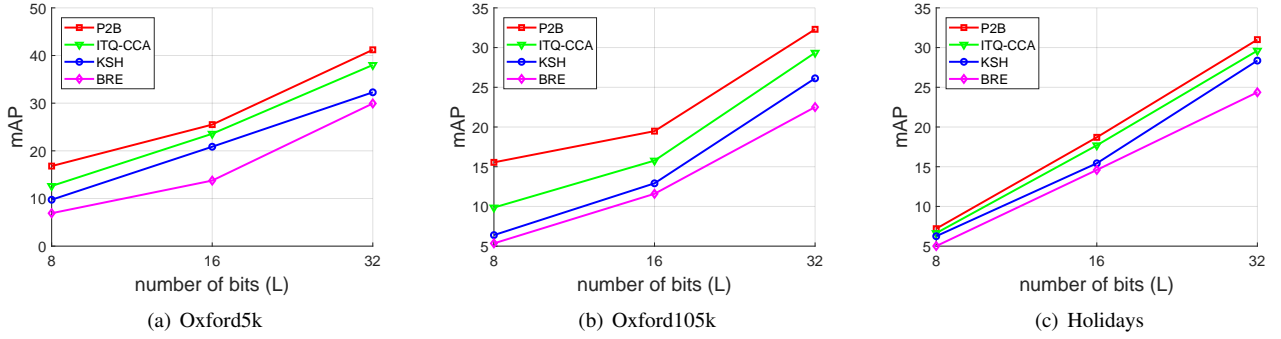


Fig. 5. mAP comparison between the proposed P2B and the state-of-the-art supervised hashing methods on Oxford5k, Oxford105k, and Holidays datasets.

L	128	256	512
SSDH	46.23	50.47	59.17
P2B	60.53	69.20	74.84

TABLE II
MAP RESULTS OF SSDH [26] AND P2B ON OXFORD5K WHEN BOTH ARE FINE-TUNED ON OUR DATASET (SECTION III-B).

otherwise, non matching. In the other words, each 3D model is considered as a class. Our empirical results finds that for KSH and BRE, the similarity matrix constructed by the second approach gives better results than the first approach, e.g., for KSH on Oxford5k dataset, the mAP at $L = 32$ is 19.11% and 32.28% for the first and the second approach, respectively. This is understandable. Although the first approach uses better strategy for creating matching pairs, the matrix S is very sparse. Hence the non-matching pairs strongly dominate the matching pairs during training, leading to poor results. In the following, we consistently use the second approach (i.e., considering each 3D model as a class) for building the training data for compared methods.

Figure 5 shows the comparative results between methods. It clearly shows that P2B outperforms compared methods, especially on Oxford5k and Oxford105k datasets. The poor accuracy of compared methods may be because the “labels” (i.e., the 3D model indexes) which are used to supervise their training are much weaker than true semantic label used in their original works.

Compare with SSDH [26]:

Here we compare our P2B to the state-of-the art deep supervised hashing method SSDH [26]. It is worth noting that in [26], the authors mainly evaluate their work on manually-labeled datasets, e.g., CIFAR10, MNIST, SUN397, etc. They do not present a fully evaluation on image retrieval benchmarks such as Holidays, Oxford5k, Oxford105k.

To make a fair comparison with SSDH, we fine-tune the SSDH model on our training dataset, in which each 3D model is considered as a class. Follow SSDH setting, we present comparative results at high code lengths, i.e., 128, 256, and 512. As shown in Table II, P2B improves over SSDH with large margins, i.e., our method gets 14.3%, 18.73%, and 15.67% better than SSDH at code length 128, 256 and 512, respectively. It is worth mentioning that in SSDH [26], the authors also reported a mAP of 63.80% (at $L = 512$ bits) on the Oxford5k dataset with the model fine-tuned on

a semi-manually labelled landmark dataset [15]. This mAP is higher than the mAP 59.17% when fine-tuned SSDH on our training dataset (Table II). This is understandable because the dataset [15] used in SSDH paper is semi-manually labeled. Hence, it is likely less noisy than our automatically-created dataset. In spite of that, at the same code length 512, our P2B achieves 74.84% mAP which is significantly higher than 63.80% of SSDH. The superior performance of P2B over SSDH confirms the effectiveness of the proposed framework for learning discriminative compact binary codes.

V. CONCLUSION

In this paper, we proposed an end-to-end framework for learning compact binary codes for image retrieval without human intervention. Specifically, we proposed a novel pairwise loss function and an alternating optimization to attack the associated binary constrained problem. We also proposed to automatically mine matching/non-matching pairs for training the network by exploiting the 3D reconstruction information. The experimental results on three image retrieval benchmarks show that the proposed method outperforms both state-of-the-art unsupervised and supervised hashing methods.

REFERENCES

- [1] R. Arandjelovic and A. Zisserman, “All about VLAD,” in *CVPR*, 2013.
- [2] T.-T. Do, Q. Tran, and N.-M. Cheung, “FAemb: a function approximation-based embedding method for image retrieval,” in *CVPR*, 2015.
- [3] H. Jégou, M. Douze, and C. Schmid, “Improving bag-of-features for large scale image search,” *IJCV*, pp. 316–336, 2010.
- [4] T. Hoang, T.-T. Do, D.-K. L. Tan, and N.-M. Cheung, “Selective deep convolutional features for image retrieval,” in *ACM-MM*, 2017.
- [5] H. Jégou, M. Douze, C. Schmid, and P. Pérez, “Aggregating local descriptors into a compact image representation,” in *CVPR*, 2010.
- [6] H. Jégou and A. Zisserman, “Triangulation embedding and democratic aggregation for image search,” in *CVPR*, 2014.
- [7] T.-T. Do and N.-M. Cheung, “Embedding based on function approximation for large scale image search,” *TPAMI*, 2017.
- [8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *NIPS*, 2012.
- [9] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *CoRR*, 2014.
- [10] F. Schroff, D. Kalenichenko, and J. Philbin, “Facenet: A unified embedding for face recognition and clustering,” in *CVPR*, 2015.
- [11] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *CVPR*, 2015.
- [12] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick, “Mask R-CNN,” in *ICCV*, 2017.

- [13] T.-T. Do, A. Nguyen, I. D. Reid, D. G. Caldwell, and N. G. Tsagarakis, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *ICRA*, 2018.
- [14] R. Arandjelovic, P. Gronát, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: CNN architecture for weakly supervised place recognition," in *CVPR*, 2016.
- [15] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky, "Neural codes for image retrieval," in *ECCV*, 2014.
- [16] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki, and S. Carlsson, "From generic to specific deep representations for visual recognition," in *CVPRW*, 2015.
- [17] M. A. Carreira-Perpinan and R. Raziherchikolaei, "Hashing with binary autoencoders," in *CVPR*, 2015.
- [18] T.-T. Do, D.-K. Le Tan, T. T. Pham, and N.-M. Cheung, "Simultaneous feature aggregating and hashing for large-scale image search," in *CVPR*, 2017.
- [19] Y. Gong and S. Lazebnik, "Iterative quantization: A procrustean approach to learning binary codes," in *CVPR*, 2011.
- [20] F. Zhao, Y. Huang, L. Wang, and T. Tan, "Deep semantic ranking based hashing for multi-label image retrieval," in *CVPR*, 2015.
- [21] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *CVPR*, 2015.
- [22] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *CVPR*, 2016.
- [23] T.-T. Do, A.-D. Doan, and N.-M. Cheung, "Learning to hash with binary deep neural network," in *ECCV*, 2016.
- [24] K. He, F. Wen, and J. Sun, "K-means hashing: An affinity-preserving quantization method for learning binary compact codes," in *CVPR*, 2013.
- [25] T.-T. Do, A.-D. Doan, D. T. Nguyen, and N.-M. Cheung, "Binary hashing with semidefinite relaxation and augmented lagrangian," in *ECCV*, 2016.
- [26] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *TPAMI*, 2017.
- [27] K. Lin, J. Lu, C.-S. Chen, and J. Zhou, "Learning compact binary descriptors with unsupervised deep neural networks," in *CVPR*, 2016.
- [28] 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*, pp. 4766–4779, 2015.
- [29] G. E. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, pp. 1527–1554, 2006.
- [30] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *CVPRW*, 2015.
- [31] F. Radenovic, G. Tolias, and O. Chum, "CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples," in *ECCV*, 2016.
- [32] B. Zhuang, G. Lin, C. Shen, and I. D. Reid, "Fast training of triplet-based deep binary embedding networks," in *CVPR*, 2016.
- [33] G. Tolias, R. Sircé, and H. Jégou, "Particular object retrieval with integral max-pooling of CNN activations," in *ICLR*, 2016.
- [34] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *CVPRW*, 2014.
- [35] F. Radenovic, J. L. Schönberger, D. Ji, J. Frahm, O. Chum, and J. Matas, "From dusk till dawn: Modeling in the dark," in *CVPR*, 2016.
- [36] B. G. V. Kumar, B. Harwood, G. Carneiro, I. D. Reid, and T. Drummond, "Smart mining for deep metric learning," in *ICCV*, 2017.
- [37] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, "Discriminative learning of deep convolutional feature point descriptors," in *ICCV*, 2015.
- [38] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. World Scientific, 2006.
- [39] A. Vedaldi and K. Lenc, "Matconvnet - convolutional neural networks for MATLAB," *CoRR*, 2014.
- [40] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *CVPR*, 2007.
- [41] J.-P. Heo, Y. Lee, J. He, S.-F. Chang, and S.-e. Yoon, "Spherical hashing," in *CVPR*, 2012.
- [42] A. Sablayrolles, M. Douze, N. Usunier, and H. Jégou, "How should we evaluate supervised hashing?" in *ICASSP*, 2017.
- [43] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang, "Supervised hashing with kernels," in *CVPR*, 2012.
- [44] B. Kulis and T. Darrell, "Learning to hash with binary reconstructive embeddings," in *NIPS*, 2009.
- [45] L. van der Maaten and G. Hinton, "Visualizing high-dimensional data using t-sne," *Journal of Machine Learning Research*, 2008.