The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Deep Region Hashing for Generic Instance Search from Images

Jingkuan Song, Tao He, Lianli Gao, Xing Xu, Heng Tao Shen[*]
Center for Future Media and School of Computer Science and Engineering,
University of Electronic Science and Technology of China.
{jingkuan.song, hetaoconquer}@gmail.com, {lianli.gao,xing.xu}@uestc.edu.cn, shenhengtao@hotmail.com

Abstract

Instance Search (INS) is a fundamental problem for many applications, while it is more challenging comparing to traditional image search since the relevancy is defined at the instance level. Existing works have demonstrated the success of many complex ensemble systems that are typically conducted by firstly generating object proposals, and then extracting handcrafted and/or CNN features of each proposal for matching. However, object bounding box proposals and feature extraction are often conducted in two separated steps, thus the effectiveness of these methods collapses. Also, due to the large amount of generated proposals, matching speed becomes the bottleneck that limits its application to large-scale datasets. To tackle these issues, in this paper we propose an effective and efficient Deep Region Hashing (DRH) approach for large-scale INS using an image patch as the query. Specifically, DRH is an end-to-end deep neural network which consists of object proposal, feature extraction, and hash code generation. DRH shares full-image convolutional feature map with the region proposal network, thus enabling nearly cost-free region proposals. Also, each high-dimensional, real-valued region features are mapped onto a low-dimensional, compact binary codes for the efficient object region level matching on large-scale dataset. Experimental results on four datasets show that our DRH can achieve even better performance than the state-of-the-arts in terms of mAP, while the efficiency is improved by nearly 100 times.

Introduction

Large-scale instance search (INS) is to efficiently retrieve the images containing a specific instance in a large scale image dataset, giving a query image of that instance. It has long been a hot research topic due to the many applications such as image classification and detection (Vedaldi and Zisserman 2012; Wang et al. 2016).

Early research (Philbin et al. 2007; Qin, Wengert, and Gool 2013) usually extracts image-level handcrafted features such as color, shape and texture to search a user query. Recently, due to the development of deep neural networks, using the outputs of last fully-connected network layers as global image descriptors to support image classification and retrieval have demonstrated their advantage over prior state-of-the-art (Ng, Yang, and Davis 2015; Razavian et al. 2014; Gao et al. 2017). More recently, we have witnessed the research attention shifted from features extracted from the fully-connected layers to deep convolutional layers (Babenko and Lempitsky 2015), and it turns out that convolutional layer activations have superior performance in image retrieval. To improve the search efficiency, lots of hashing methods were proposed, from traditional handcrafted based hashing methods (Liu et al. 2013; Song et al. 2017; Zhang et al. 2016; Liu et al. 2014a; Zhu, Zhang, and Huang 2014; Song et al. 2013; Yang et al. 2016; Zhu et al. 2013; 2017) to deep learning based hashing methods (Zhao et al. 2015; Lin et al. 2015; Li, Wang, and Kang 2016; Yao et al. 2016; Wang et al. 2016; Song et al. 2018). However, using image-level features may fail to search instances in images. As illustrated in Fig. 1, sometimes a target region is only a small portion of a database image. Directly comparing the query image and a whole image in the database may result an inaccurate matching. Therefore, it is necessary to consider the local region information.

Recently, many successful INS approaches (Tao et al. 2014; Mohedano et al. 2016) were proposed by combining fast filtering with more computationally expensive spatial verification and query expansion. More specifically, all images in a database are firstly ranked according to their distances to the query, and then a spatial verification is applied to re-rank the search results, and finally a query expansion is followed to improve the search performance. An intuitive way for spatial verification is the sliding window strategy. It generates sliding windows at different scales and aspect ratios over an image. Each window is then compared to the query instance in order to find the optimal location that contains the query. A more advanced strategy is to use object proposal algorithms (Uijlings et al. 2013; Girshick et al. 2014; Pont-Tuset et al. 2016; Chavali et al. 2016). It evaluates many image locations and determines whether they contain the object or not. Therefore, the query instance only need to compare with the object proposal instead of all the image locations. Existing works (Tao et al. 2014; Tao, Smeulders, and Chang 2015) have demonstrated the success of many complex ensemble systems that are conducted by firstly generating object bounding box propos-

[*]Heng Tao Shen and Lianli Gao are the corresponding authors.

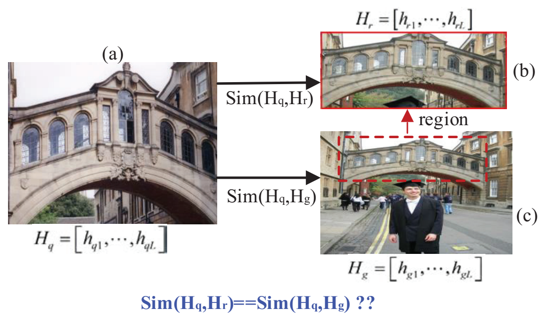
$H_r = [h_{r1}, \cdots, h_{rL}]$
(a)
(b)
Sim(Hq,Hr)
region
Sim(Hq,Hg)
(c)
$H_q = [h_{q1}, \cdots, h_{qL}]$
$H_g = [h_{g1}, \cdots, h_{gL}]$
Sim(Hq,Hr)==Sim(Hq,Hg) ??

Figure 1: Purely using global hash codes to calculate the similarity between query instance and database images is not accurate enough. (a) is a query instance with a deep hash code representation H_q; (c) is a database image with a deep global region hash code H_g by taking the whole image as a region; (b) is a region pooled from (c) and with a deep region hash code H_l. However, it is clear that conducting INS using global region (c) is not as accurate as using local region (b).

als and then extracting handcrafted and/or CNN features for each bonding box. Because the region detection and feature extraction are conducted in two separate steps, the detected regions may not be the optimal one for feature extraction, leading to suboptimal results.

Faster R-CNN (Ren et al. 2015) introduces a region proposal network which integrates feature extraction and region proposal in a single network. The detection network shares the full-image convolutional features, thus enabling nearly cost-free region proposals. Inspired by this, (Salvador et al. 2016) utilize Faster R-CNN to propose some object candidates and perform search on the extracted convolutional features. However, efficiency is a big issue for these region proposal based methods. The successfully selective search and RPN are still generating hundreds to thousands of candidate regions. If a dataset has N images and each image contains M regions, each region is represented by a d-dimensional feature vector, then an exhaustive search for each query requires $N \times M \times d$ operation to measure the distances to the candidate regions. Therefore, the expensive computational cost for nearest neighbors search in high-dimensional data limits their application to large scale datasets.

In this paper we propose a fast and effective Deep Region Hashing (DRH) approach for visual instance search in large scale datasets with an image patch as the query. It is worthwhile to highlight the following aspects of DRH: (1) We propose an effective and efficient end-to-end Deep Region Hashing (DRH) approach, which consists of object proposal, feature extraction, and hash code generation, for large-scale INS with an image patch as the query. Given an image, DRH can automatically generate the hash codes for the whole image and the object candidate regions. (2) We design different strategies for object proposals based on the convolutional feature map. The region proposal is nearly cost-free, and we also integrate hashing strategy into our approach to enable efficient INS on large-scale datasets. (3) Extensive experimental results on four datasets show that our generated hash codes can achieve even better performance than the state-of-the-arts using real-valued features in terms of mAP, while the efficiency is improved by nearly 100 times.

Methodology

This paper explores instance search from images using hash codes of image regions detected by an object proposal CNN or sliding window. In our setup, query instances are defined by a bounding box over the query images. The framework of our DRH-based instance search is shown in Fig.2, and it has two phases, i.e., offline DRH training phase and online instance search phase. In this section, we describe these two phases in details.

Deep Region Hashing

As is shown in Fig. 2, the DRH training phase consists of three components, i.e., CNN-based feature map generation, region proposals and region hash code generation. Next, we illustrate each of them.

CNN-based Representations As mentioned above, we focus on leveraging convolutional networks for hash learning. We adopt the well known architecture in (Conneau et al. 2016) as our basic framework. As shown in Fig. 2, our network has 5 groups of convolution layers, 4 max convolution-pooling layers, 1 Region of Interest Pooling layer (RoI) and 1 hashing layer. Following (Conneau et al. 2016), we use 64, 128, 256, 512, 512 filters in the 5 groups of convolutional layers, respectively. More specifically, our network firstly processes an arbitrary image with several convolutional (conv) and max pooling layers to produce a conv feature map. Next, for each region proposal, a region of interest (RoI) pooling layer extracts a fixed-length (i.e., 512) feature from the feature map. Each feature map is then fed into the region hash layer that finally outputs hash codes representing the corresponding RoI.

Region Pooling Layer There are two ways to generate region proposals: 1) sliding window (the number of region proposal is determined by a sliding window overlaping parameter λ) and 2) the Region Proposal Network (RPN) proposed by Fater RCNN (Ren et al. 2015). It provides a set of object proposals, and we use them as our region proposals by assuming that the query image always contains some objects.

1) **Sliding window.** Given a raw image with the size of $W \times H$, our network firstly generates a conv feature map $W_c \times H_c$, where W, W_c and H, H_c are the width and height of the original image and conv map. Next, we choose windows of any possible combinations of width $\left\{W_c, \frac{W_c}{2}, \frac{W_c}{3}\right\}$ and height $\left\{H_c, \frac{H_c}{2}, \frac{H_c}{3}\right\}$. We use a sliding window strategy directly on the conv map with overlap in both directions. The percentage of overlap is measured by a overlapping parameter λ. Next, we utilize a simple filtering strategy to discard those windows. If $\frac{W}{H} \le th$, we discard the scale $\frac{W_c}{3}$. When $\frac{H}{W} \le th$, we discard $\frac{H_c}{3}$, otherwise we keep the original settings. When the width and height are set to W_c and H_c, we

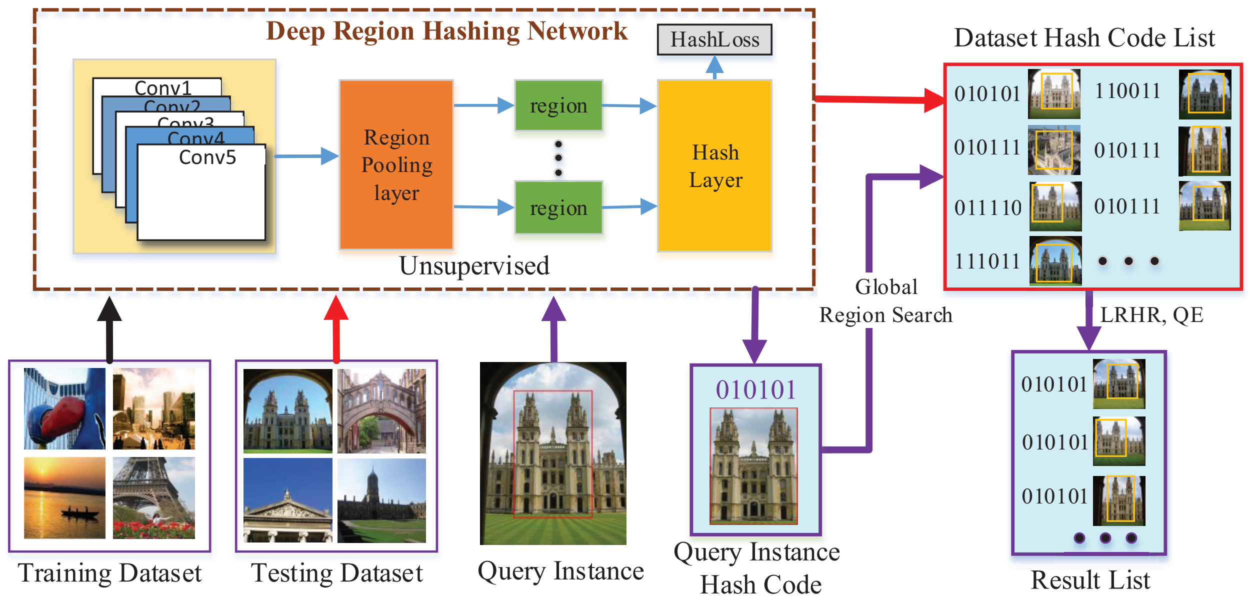

Figure 2: The overview of DRH. It contains two parts, i.e., deep region hashing network part and instance search part. The first part trains a network for feature extraction, object proposal and hash code generation, while the second part utilizes DRH to get the query results.

can generate a global region image descriptor, which will be used as the input for global region hash code generation.

2) **RPN network.** Faster R-CNN (Ren et al. 2015) proposed a Region Proposal Network (RPN), which takes conv feature map as input and outputs a set of rectangular object proposals, each with an abjectness scores. Given a conv feature map $W_c \times H_c$, it generates $W_c \times H_c \times 9$ (typically $216,00$ regions). Specifically, the RPN proposed by Faster R-CNN is able to propose object regions and extract the convolutional activations for each of them. Therefore, for each region proposal, it can compute a descriptor by aggregating the activations of that window in the RoI pooling layer, giving raise to the region-wise descriptors. In order to conduct image-level operation, we obtain a global region by setting the whole conv map as a region. To conduct instance-level search, we obtain local regions which have smaller size than the global region.

After region proposal generation either adopting sliding window based approach or RPN approach, for each image we obtain a set of regions which include a global region and several local regions. Next, we apply max pooling to the corresponding region window on the conv map to generate ROI feature for each region.

Region Hashing Layer For instance search, directly using RoI features to compare the query feature with all the ROI features is effective and it should work well in small dataset. However, in terms of large-scale dataset, this approach is computationally expensive and requires large memory since it involves huge vector operations. In this paper, we aim to improve its accuracy and efficiency. Inspired by the success of hashing in terms of search accuracy and search time cost, we propose a region hashing layer to generate hash codes for the region proposals.

Here, we convert each RoI feature to a set of binary hash code. Given a RoI feature $x_r \in \mathbb{R}^{1\times512}$, the latent region hash layer output is $h_r = \sigma(W_r x_r + b_r)$, where W_r is the hash function and b_r is the bias. Next, we perform the binary hash for the output $h_r \in \mathbb{R}^{1\times L_r}$ to obtain a binary code $y_r = sgn(\sigma(W_r x_r + b_r)) \in \{0,1\}^{L_r}$. To obtain a qualified binary hash code, we propose the following optimization problem:

$$\min_{W_r,b_r} \ell = \frac{1}{2}g(y_r,h_r) - \frac{\alpha}{2}t(h_r) + \frac{\beta}{2}r(W_r) + \frac{\eta}{2}o(b_r) \quad (1)$$

where $g(y_r,h_r)$ is the penalty function to minimize the quantization loss between the RoI feature h_r and y_r. $t(h_r)$, $r(W_r)$ and $o(b_r)$ are the regularization term for h_r, W_r and b_r. α, β and η are parameters. We define $g(y_r,h_r)$ and $t(h_r)$ as bellow:

$$g(y_r,h_r) = \|y_r - h_r\|_F^2, \quad t(h_r) = tr\left(h_r h_r^T\right) \quad (2)$$

where $-t(h_r)$ aims to maximize the variance of learned binary vectors to ensure balanced bits and $tr(.)$ operation is to calculate the trace of a matrix . The last two terms $r(W_r)$ and $o(b_r)$ are regularizers to control the scales of the parameters, and they are defined as:

$$r(W_r) = \|W_r\|_F^2, \quad o(b_r) = \|b_r\|_F^2 \quad (3)$$

To solve this optimization problem, we employ the stochastic gradient descent method to learn parameters

W_r, b_r. The gradient of the objective function in (2) with respect to different parameters are computed as following:

$$\begin{aligned}\frac{\partial \ell}{\partial W_{\mathrm{r}}} &= \left((h_r - \mathrm{y}_r - \alpha h_r) \odot \sigma'(W_r h_r + b_r) \right) x_r{}^T + \beta W_r \\ \frac{\partial \ell}{\partial b_r} &= (h_r - \mathrm{y}_r - \alpha h_r) \odot \sigma'(W_r h_r + b_r) + \eta b_r\end{aligned} \tag{4}$$

where $\odot$ is element-wise multiplication. The parameters W_{r} and b_r are updated with gradient descent algorithm until convergence.

Network Training We train our models using stochastic gradient descent with momentum, with back-propagation used to compute the gradient of the objective function $\nabla Loss\,(W_r, b_r)$ with respect to all parameters W_r and b_r. More specifically, we randomly initialize the RoI pooling layer and region hashing layer by drawing weights from a zero-mean Gaussian distribution with standard deviation 0.01. The shared convolutional layers are initialized by pre-training a model (Radenovic, Tolias, and Chum 2016). Next, we tune all the layers of the RoI layer and the hashing layer to conserve memory. In particular, for RPN training we fix the convolutional layers and only update the RPN layers parameters. The pascal_voc 2007 dataset is used to train the RPN and for this study we choose the Top $\tilde{3}00$ regions as input to the region hash layer. For sliding window based region proposal, we use different scales to directly extract regions and then input them into the region hash layer. At this stage, the parameters for conv layers and region pooling layers are obtained. Next, we fix the parameters of conv layers and the region pooling layer to train the region hash layer. In addition, learning rate is set to 0.01 in the experiments. The parameters α, β, η were empirically set as 100, 0.001 and 0.001 respectively.

Instance Search using DRH

This section describes the INS pipeline by utilizing DRH approach. This pipeline consists of a Global Region Hashing (gDRH) search, Local Region Hashing (lDRH) re-ranking and two Region Hashing Query Expansions (QE).

Global Region Hashing Search (gDRH). It is conducted by computing the similarity between the hash code of a query and the global region hash codes of dataset images. For each item in the dataset, we use the hash codes generated by the whole image region. Next, the image query list is sorted based on the hamming distance of dataset items to the query. After gDRH, we choose top M images to form the 1^{st} ranking list $X^{rank} = \left\{x_1^{rank}, \cdots, x_M^{rank}\right\}$.

Local Region Hashing Re-search (lDRH). After we obtained the X^{rank}, we perform the local region hashing search by computing the hamming distance between query instance hash code and hash codes of each local region in the top M images. To clarify, the hashing query expansions introduced below is optional. For each image, the distance score is calculated by using the minimal hamming distances between the hash code of query instance and all the local region hash codes within that image. Finally, the images are re-ordered based on the distances and we get the final query list $X_1^{rank'} = \left\{x_1^{rank'}, \cdots, x_M^{rank'}\right\}$.

Region Hashing Query Expansions (QE). In this study, we further investigate two query expansion strategies based on the global and local hash codes:

1) *Global Region Hashing Query expansion (gQE).* After the Global Region Search, we conduct the global hashing query expansion, which is applied after the gDRH and before the lDRH. Specifically, we firstly choose the global hash code of the top q images within the X^{rank} to calculate similarities between each of them with each global hash code of the X^{rank} images. For x_i^{rank}, the similarity score is computed by using the max similarity between x_i^{rank} and the top q images' global region hash codes. Next, the X^{rank} is re-ordered to $X^{rankq} = \left\{x_1^{rankq}, \cdots, x_M^{rankq}\right\}$ and finally we conduct lDRH on the X^{rankq} list.

2) *Local Region Hashing Query Expansion (lQE).* Here, to conduct local region hashing query expansion, initially we need to conduct gDRH and lDRH to get the $X^{rank'}$. Here gQE is optional. Next, we get the top q best matched region hash code from images within the $X^{rank'}$ list and then compare the q best region hash codes with all the regions of the $X^{rank'}$. Thus, we choose the max value as the similarity score for each image and finally the $X^{rank'}$ is re-ranked based on those similarity scores.

Experiment

We evaluate our algorithm on the task of instance search. Firstly, we study the influence of different components of our framework. Then, we compare DRH with state-of-the-art algorithms in terms of efficiency and effectiveness on four standard datasets.

Datasets and Evaluation Metric

We consider two publicly available datasets. 1) **Oxford Buildings** (Philbin et al. 2007) contains two sub-datasets: *Oxford 5k*, which contains 5,062 high revolutionary (1024×768) images, including 11 different landmarks (i.e., a particular part of a building), and each represented by several possible queries; and *Oxford 105K*, which combines Oxford 5K with $100,000$ distractors to allow for evaluation of scalability. 2) **Paris Buildings** (Philbin et al. 2008) contains two sub-datasets as well: *Paris 6k* including 6,300 high revolutionary (1024×768) images of Paris landmarks; and *Paris 106 K*, which is generated by adding $100,000$ Flickr distractor images to *Paris 6k* dataset. Compared with Oxford Buildings, it has images representing buildings with some similarities in architectural styles. The hashing layer is trained on a dataset which is composed of *Oxford 5K* training examples and *Paris 6k* training examples.

For each dataset, we follow the standard procedures to evaluate the retrieval performance. In this paper, we use mean Average Precision (mAP) as the evaluation metric.

Performance using Different Settings

There are several different settings affecting the performance of our algorithm. To comprehensively study the performance of DRH with different settings, we further study different components of DRH including: 1) Hash code

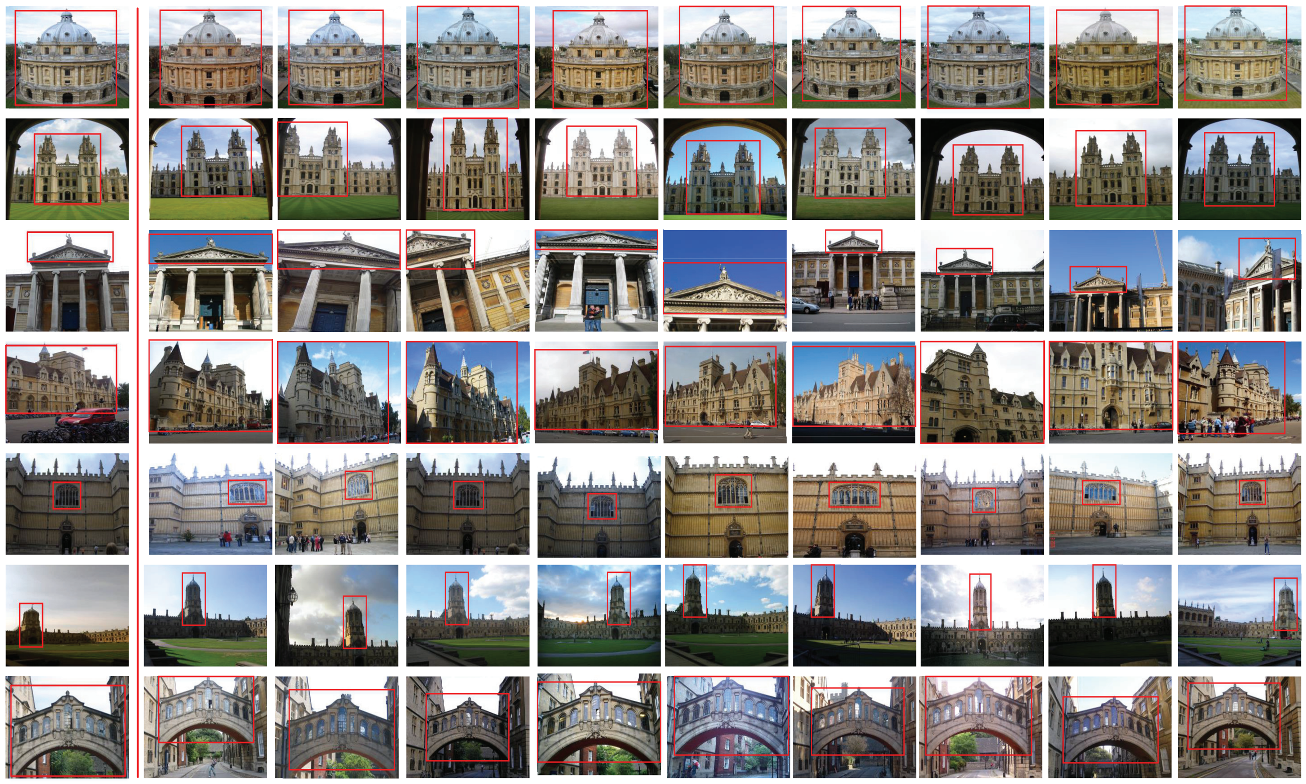

Figure 3: The qualitative results of DRH searching on 5K dataset. 7 queries (i.e., *redcliffe_camera, all_souls, ashmolean, balliol, bodleian, christ_church and hertford*) and the corresponding search results are shown in each row. Specifically, the query is shown in the left, with selected top 9 ranked retrieved images shown in the right.

length L; 2) region proposal component and its parameters; and 3) query expansion and its settings.

The Effect of Hash Code Length L Firstly, we investigate the influence of hash code length by using gDRH to conduct INS. We report the mAP of gDRH with different code length L, and also the mAP of INS using the max pooling on conv5 feature. The results are shown in Tab. 1. These results show that with the increase of hash code length L from 128 to 4096, the performance of gDRH increases from 42.5% to 78.3% on the *oxford 5k* dataset, while the mAP increases from 53.1% to 81.5% on the *Paris 6k* dataset. Comparing $L = 1024$ to $L = 4096$, the code length increased to 4 times, but the performance is only improved by 3.5% and 4.2%. Therefore, to balance the search efficiency and effectiveness, we use $L = 1024$ as the default setting. On the other hand, the best performance of gDRH (i.e., $L = 4096$) is slightly lower than that of using max pooling on conv5 features, with 1.4% and 0.9% decreases, respectively. This is due to the information loss of hash codes.

The Effect of Region Proposals In this sub-experiment, we explore firstly the two ways of region proposal: sliding window based and RPN based approaches. Both of them can generate different number of object proposals. For RPN, we choose 300 regions to generate region hash code for lDRH. For sliding window, we tune $\lambda = [0.4, 0.5, 0.6, 0.7]$ to get different number of object proposals. We further study the mAP of different top M results. gDRH can get some initial retrieval results, and in this subsection, we refine the results by using lDRH, i.e., gDRH+lDRH. The reported results (Tab. 2) are on the Oxford $5k$ dataset. From these results, we can make several observations.

Table 1: The performance (mAP) variance with different code lengths L using gDRH.

Method	oxford 5K	Paris 6K
gDRH (128-bits)	0.425	0.531
gDRH (256-bits)	0.583	0.629
gDRH (512-bits)	0.668	0.724
gDRH (1024-bits)	0.748	0.773
gDRH (4096-bits)	0.783	0.815
Max Conv Feature 512	0.797	0.824

1) Compared with RPN based DRH, the sliding window based DRH performs slightly better. It also requires less number of region boxes and does not need training a RPN network, thus we adopt sliding window based DRH as the

Table 2: The influence of Top M, λ and the way of region generation. This experiment is conducted on the oxford $5k$ dataset with $L = 1024$.

	Sliding Window based DRH				RPN-DRH
gDRH+lDRH	$\lambda = 0.4$ ∼21boxes	$\lambda = 0.5$ ∼36 boxes	$\lambda = 0.6$ ∼40boxes	$\lambda = 0.7$ ∼60boxes	∼300 boxes
M=100	0.773	0.774	0.776	0.772	0.772
M=200	0.779	0.780	0.781	0.778	0.778
M=400	0.779	0.781	**0.783**	0.780	0.780
M=800	0.782	0.784	**0.786**	0.783	**0.783**

Table 3: The effect of query expansions (gQE and lQE).

Settings	gDRH	gQE	lDRH	lQE	oxford 5k	Paris 6k
No QE	yes	no	no	no	0.745	0.773
	yes	no	yes	no	**0.783**	**0.801**
gQE (different q)	yes	yes,q=4	no	no	0.813	0.823
	yes	yes,q=5	no	no	0.809	0.828
	yes	yes,q=6	no	no	**0.815**	0.835
	yes	yes,q=7	no	no	0.789	**0.842**
gQE+lQE (q=6)	yes	yes	no	no	0.815	0.835
	yes	yes	yes	no	0.804	0.831
	yes	no	yes	yes	0.833	0.819
	yes	yes	yes	yes	**0.851**	0.849
gQE+lQE (q=7)	yes	yes	no	no	0.789	0.842
	yes	yes	yes	no	0.804	0.834
	yes	no	yes	yes	0.826	0.821
	yes	yes	yes	yes	0.838	**0.854**

default approach to conduct the following experiments. RPN is supposed to generate better region proposals than sliding window, but a worse performance is achieved in the experiment. One potential reason is that RPN is trained on pascal_voc dataset, which is robust to propose the object of 'buildings'. On the other hand, the objects in 'Oxford buildings' are usually very large, thus they can be readily captured by sliding windows.
2) For sliding window based DRH, λ affects the performance. In general, the performance is not very sensitive to λ. The best performance is achieved when $\lambda = 0.6$, and it only generates about 40 regions for each image. In the following experiments, the default setting for λ is 0.6.
3) The top M results have no significant impact on mAP either for DRH. In general, with the increase of M, the performance increases as well. When $M = 800$, the algorithm achieves the best results. When $M = 400$, the performance is 0.3% lower than $M = 800$. Considering the memory cost, we choose $M = 400$ for the following experiments.

The Effect of Query Expansions In this subsection, we study the effect of query expansions. This study aims to explore: the effect of lDRH, the effect of q for gQE, the effect of gQE and lQE for DRH, as well as several combinations. The experiment results are shown in Tab. 3. From Tab. 3, we have some observations below:
1) The lDRH improves the INS. In terms of mAP, for oxford $5k$ and paris $6k$ datasets, it increases by 3.8% and 2.8% , respectively.
2) The gQE improves the performance of DRH. When $q = 6$, it performs the best (i.e., 85.1%) on the Oxford $5k$ dataset, and when $q = 7$ it gains the best performance on the Paris $6k$ dataset. In the following experiments, q is set as 6.
3) The experimental results show that by combining gQE and lQE, DRH performs the best (i.e., 85.1% for oxford $5k$ and 85.4% for Paris $6k$ datasets). Therefore, query expansion strategy can improve INS.

Comparing with State-of-the-art Methods

To evaluate the performance of DRH, we compare our methods with the state-of-the-art deep features based methods. These methods can be divided into four categories: 1) Feature Encoding approaches (Iscen et al. 2014; Iscen, Rabbat, and Furon 2016; Ng, Yang, and Davis 2015). 2) Aggregating Deep features (Razavian et al. 2014; Babenko and Lempitsky 2015; Tolias, Sicre, and Jégou 2015; Kalantidis, Mellina, and Osindero 2015; Mohedano et al. 2016; Salvador et al. 2016; Tao et al. 2014; Tao, Smeulders, and Chang 2015). 3) Integrating with Locality and QE. 4) Hashing methods (Jegou, Douze, and Schmid 2008; Liu et al. 2014b). Lots of hashing methods are proposed for image retrieval .

The comparison results are shown in Tab. 4 and we also show some qualitative results in Fig. 3. From these results, we have the following observations. (1) First, our approach performs the best on both two large-scale datasets oxford $105k$ and Paris $106k$ with the mAP of 0.825 and 0.802, respectively. In addition, the performance on oxford $105k$ is higher than Tolias *et al.* +AML+QE (Tolias, Sicre, and Jégou 2015) with a 9.3% increase. (2) On Paris $6k$ dataset, our performance is slightly lower than that of Tolias *et al.* +AML+QE (Tolias, Sicre, and Jégou 2015) by 1.1%, but on the Oxford $5k$ dataset, it outperforms Mohedano *et al.*+Rerank+LQE (Mohedano et al. 2016) by 6.3%. (3) The performance of CNN features aggregation is better than CNN features in general, and it can be further improved by integrating re-ranking and query expansion strategies. On the other hand, hashing methods perform the worst in terms of mAP. This is probably due to the information loss of hash codes. (4) The qualitative results show that DRH can retrieve instances precisely, even for the cases when the query image is different from the whole target image, but similar to a small region of it.

Efficiency Study

In this subsection, we compare the efficiency of our DRH to the state-of-the-art algorithms. The time cost for instance search usually consists of two parts: filtering and re-ranking. For our DRH, the re-ranking is conducted using M=400 candidates, and each of which has around 40 local regions. Therefore, the time cost for re-ranking can be ignored compared with the filtering step. For the comparing algorithms, they have different re-ranking strategies, and some of them do not have this step. Therefore, we only report the time for filtering step of these comparing algorithms.

We choose (Tolias, Sicre, and Jégou 2015) as the representative algorithm for non-hashing algorithms. To make fair comparison, we implement the linear scan of the conv5 feature (Tolias, Sicre, and Jégou 2015) and hash codes using CPP. All the experiments are done on a server with In-

Table 4: Comparison to state-of-the-art methods using Deep representations.

	Methods	Oxford 5K	Oxford 105K	Paris 6K	Paris 106K
CNN Feature Encoding	Iscen et al.-Keans (Iscen et al. 2014)	0.656	0.612	0.797	0.757
	Iscen et al.-Rand (Iscen et al. 2014)	0.251	0.437	0.212	0.444
	Ng et al. (Ng, Yang, and Davis 2015)	0.649	-	0.694	-
	Iscen et al. (Iscen, Rabbat, and Furon 2016)	0.737	0.655	0.853	0.789
CNN Featues Aggregation	Razavian et al. (Razavian et al. 2014)	0.556	-	0.697	-
	Bebenko et al. (Babenko and Lempitsky 2015)	0.657	0.642	-	-
	Tolias et al. (Tolias, Sicre, and Jégou 2015)	0.669	0.616	0.830	0.757
	Kalantidis et al. (Kalantidis, Mellina, and Osindero 2015)	0.684	0.637	0.765	0.691
	Mohedano et al. (Mohedano et al. 2016)	0.739	0.593	0.820	0.648
	Salvador et al. (Salvador et al. 2016)	0.710	-	0.798	-
	Tao et al. (Tao et al. 2014)	0.765	-	-	-
	Tao et al. (Tao, Smeulders, and Chang 2015)	0.722	-	-	-
Integration with Locality and QE	Kalantidis et al. +GQE (Kalantidis, Mellina, and Osindero 2015)	0.749	0.706	0.848	0.794
	Tolias et al. +AML+QE (Tolias, Sicre, and Jégou 2015)	0.773	0.732	**0.865**	0.798
	Mohedano et al. (Mohedano et al. 2016) +Rerank+LQE	0.788	0.651	0.848	0.641
	Salvador et al. (Salvador et al. 2016) +CS-SR+QE	0.786	-	0.842	-
Hashing	Jegou et al. (Jegou, Douze, and Schmid 2008)	0.503	-	0.491	-
	Liu et al. (Liu et al. 2014b)	0.518	-	0.511	-
ours	DRH(gDRH+lDRH)	0.783	0.754	0.801	0.733
	DRH All	**0.851**	**0.825**	0.849	**0.802**

Table 5: The time cost (ms) for different algorithms

Datasets	CNN feature (Tolias 2015)) 512-D conv5	DRH 512-D hash	DRH 1024-D hash
Oxford 105K	1078	3	12
Paris 106K	1137	3	12

tel@Core i7-6700K. The graphics is GeGorce GTX TITAN X/PCle/SSE2. The search time is average number of milliseconds to return top $M = 400$ results to a query, and the time costs are reported in Tab. 5.

There results show that directly using max pooling features extracted from conv5 to conduct INS search is much slower than using our hash codes. It takes 1078 ms on Oxford $105k$ dataset and 1137 ms on Paris $106k$ dataset for (Tolias, Sicre, and Jégou 2015). By contrast, when the code length is 512, our DRH is more than 300 times faster than (Tolias, Sicre, and Jégou 2015), and it takes 3 ms on both Oxford $105k$ and Paris $106k$ datasets. Even when the code length increases to 1024, DRH can achieve nearly 100 times faster than (Tolias, Sicre, and Jégou 2015). Therefore, our method has the ability to work on large scale datasets.

Conclusion

In this work, we propose an unsupervised deep region hashing (DRH) method, a fast and efficient approach for large-scale INS with an image patch as the query input. We firstly utilize deep conv feature map to extract nearly 40 regions to generate hash codes to represent each image. Next, the gDRH search, gQE, lDRH and lQE are applied to further improve the INS search performance. Experiment on four datasets demonstrate the superiority of our DRH compared to others in terms of both efficiency and effectiveness.

Acknowledgments

This work is supported by the Fundamental Research Funds for the Central Universities (Grant No. ZYGX2016J085), the National Natural Science Foundation of China (Grant No. 61772116, No. 61502080, No. 61632007) and the 111 Project (Grant No. B17008).

References

Babenko, A., and Lempitsky, V. S. 2015. Aggregating deep convolutional features for image retrieval. *CoRR* abs/1510.07493.

Chavali, N.; Agrawal, H.; Mahendru, A.; and Batra, D. 2016. Object-proposal evaluation protocol is 'gameable'. In *CVPR*.

Conneau, A.; Schwenk, H.; Barrault, L.; and LeCun, Y. 2016. Very deep convolutional networks for natural language processing. *CoRR* abs/1606.01781.

Gao, L.; Guo, Z.; Zhang, H.; Xu, X.; and Shen, H. T. 2017. Video captioning with attention-based LSTM and semantic consistency. *IEEE Trans. Multimedia* 19(9):2045–2055.

Girshick, R.; Donahue, J.; Darrell, T.; and Malik, J. 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*.

Iscen, A.; Furon, T.; Gripon, V.; Rabbat, M. G.; and Jégou, H. 2014. Memory vectors for similarity search in high-dimensional spaces. *CoRR* abs/1412.3328.

Iscen, A.; Rabbat, M.; and Furon, T. 2016. Efficient large-scale similarity search using matrix factorization. In *CVPR*.

Jegou, H.; Douze, M.; and Schmid, C. 2008. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 304–317.

Kalantidis, Y.; Mellina, C.; and Osindero, S. 2015. Cross-dimensional weighting for aggregated deep convolutional features. *CoRR* abs/1512.04065.

Li, W.; Wang, S.; and Kang, W. 2016. Feature learning based deep supervised hashing with pairwise labels. In *IJCAI*, 1711–1717.

Lin, Z.; Ding, G.; Hu, M.; and Wang, J. 2015. Semantics-preserving hashing for cross-view retrieval. In *CVPR*.

Liu, X.; He, J.; Lang, B.; and Chang, S. 2013. Hash bit selection: A unified solution for selection problems in hashing. In *CVPR*, 1570–1577.

Liu, X.; He, J.; Deng, C.; and Lang, B. 2014a. Collaborative hashing. In *CVPR*.

Liu, Z.; Li, H.; Zhou, W.; Zhao, R.; and Tian, Q. 2014b. Contextual hashing for large-scale image search. *IEEE Trans. Image Processing* 23(4):1606–1614.

Mohedano, E.; Salvador, A.; McGuinness, K.; Marqués, F.; O'Connor, N. E.; and Giró i Nieto, X. 2016. Bags of local convolutional features for scalable instance search. *CoRR* abs/1604.04653.

Ng, J. Y.; Yang, F.; and Davis, L. S. 2015. Exploiting local features from deep networks for image retrieval. *CoRR* abs/1504.05133.

Philbin, J.; Chum, O.; Isard, M.; Sivic, J.; and Zisserman, A. 2007. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*.

Philbin, J.; Chum, O.; Isard, M.; Sivic, J.; and Zisserman, A. 2008. Lost in quantization: Improving particular object retrieval in large scale image databases. In *CVPR*.

Pont-Tuset, J.; Arbeláez, P.; Barron, J.; Marques, F.; and Malik, J. 2016. Multiscale combinatorial grouping for image segmentation and object proposal generation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Qin, D.; Wengert, C.; and Gool, L. V. 2013. Query adaptive similarity for large scale object retrieval. In *CVPR*, 1610–1617.

Radenovic, F.; Tolias, G.; and Chum, O. 2016. CNN image retrieval learns from bow: Unsupervised fine-tuning with hard examples. *CoRR* abs/1604.02426.

Razavian, A. S.; Sullivan, J.; Maki, A.; and Carlsson, S. 2014. Visual instance retrieval with deep convolutional networks. *CoRR* abs/1412.6574.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NIPS*.

Salvador, A.; Giro-i Nieto, X.; Marques, F.; and Satoh, S. 2016. Faster r-cnn features for instance search. In *CVPR Workshops*.

Song, J.; Yang, Y.; Yang, Y.; Huang, Z.; and Shen, H. T. 2013. Inter-media hashing for large-scale retrieval from heterogeneous data sources. In *SIGMOD*, 785–796.

Song, J.; Gao, L.; Liu, L.; Zhu, X.; and Sebe, N. 2017. Quantization-based hashing: a general framework for scalable image and video retrieval. *Pattern Recognition*.

Song, J.; He, T.; Gao, L.; Xu, X.; Alan, H.; and Shen, H. T. 2018. Binary generative adversarial networks for image retrieval. In *AAAI*.

Tao, R.; Gavves, E.; Snoek, C. G. M.; and Smeulders, A. W. M. 2014. Locality in generic instance search from one example. In *CVPR*.

Tao, R.; Smeulders, A. W. M.; and Chang, S. F. 2015. Attributes and categories for generic instance search from one example. In *CVPR*.

Tolias, G.; Sicre, R.; and Jégou, H. 2015. Particular object retrieval with integral max-pooling of CNN activations. *CoRR* abs/1511.05879.

Uijlings, J. R. R.; van de Sande, K. E. A.; Gevers, T.; and Smeulders, A. W. M. 2013. Selective search for object recognition. *IJCV* 104(2):154–171.

Vedaldi, A., and Zisserman, A. 2012. Sparse kernel approximations for efficient classification and detection. In *CVPR*, 2320–2327.

Wang, J.; Zhang, T.; Song, J.; Sebe, N.; and Shen, H. T. 2016. A survey on learning to hash. *CoRR* abs/1606.00185.

Yang, Y.; Luo, Y.; Chen, W.; Shen, F.; Shao, J.; and Shen, H. T. 2016. Zero-shot hashing via transferring supervised knowledge. In *ACM Multimedia*, 1286–1295.

Yao, T.; Long, F.; Mei, T.; and Rui, Y. 2016. Deep semantic-preserving and ranking-based hashing for image retrieval. In *IJCAI*, 3931–3937.

Zhang, H.; Zhao, N.; Shang, X.; Luan, H.; and Chua, T. 2016. Discrete image hashing using large weakly annotated photo collections. In *AAAI*, 3669–3675.

Zhao, F.; Huang, Y.; Wang, L.; and Tan, T. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In *CVPR*.

Zhu, X.; Huang, Z.; Shen, H. T.; and Zhao, X. 2013. Linear cross-modal hashing for efficient multimedia search. In *ACM Multimedia*, 143–152.

Zhu, X.; Li, X.; Zhang, S.; Xu, Z.; Yu, L.; and Wang, C. 2017. Graph PCA hashing for similarity search. *IEEE Trans. Multimedia* 19(9):2033–2044.

Zhu, X.; Zhang, L.; and Huang, Z. 2014. A sparse embedding and least variance encoding approach to hashing. *IEEE Trans. Image Processing* 23(9):3737–3750.