Large scale visual search

Wu, S.

Chapter 4

Deep Binary Codes for Large Scale Image Retrieval

Recent studies have shown that image representations built upon deep convolutional layers in Convolutional Neural Networks (CNNs) have strong discriminative characteristics. In this chapter, we present a novel and effective method to create compact binary codes (deep binary codes) based on deep convolutional features for image retrieval. Deep binary codes are generated by comparing the response from each feature map and the average response across all the feature maps on the deep convolutional layers. Additionally, a spatial cross-summing strategy is proposed to directly generate bit-scalable binary codes. As the deep binary codes on different deep layers can be obtained by passing the image through the CNN and each of them makes a different contribution to the search accuracy, we then present a dynamic, on-the-fly late fusion approach where the top N high quality search scores from deep binary codes are automatically determined online and fused to further enhance the retrieval precision. Two strengths of the proposed methods are that the generation of deep binary codes is based on a generic model, which does not require additional training for new image domains, and that the dynamic late fusion scheme is query adaptive. Extensive experimental results on well known benchmarks show that the performance of deep binary codes are competitive with state-of-the-art approaches for large scale image retrieval. Moreover, it is shown that the dynamic late fusion scheme substantially enhances the search accuracy.

4.1 Introduction

Content-based image retrieval aims to find relevant images in an image database that share a similar appearance with a given query image. This is a challenging task for large scale visual search, because one must address both the typical appearance transformations such as changes in perspective, rotation and scale; and also minimize memory, computational cost, and response time.

Traditional image retrieval systems based on visual word representations mainly owe their success to locally invariant features and large visual codebooks. The Bag-of-Words (BoW) [1] approach is usually employed to encode local features into a histogram according to the occurrence frequency of each visual word. Peronnin *et al.* proposed the Fisher Vector [2]. The visual words in a Fisher Vector are constructed with a Gaussian mixture model (GMM) where the gradients of local features corresponding to particular parameters in GMM are summed. The Fisher Vector image representation is the concatenation of each accumulated gradient. Jegou et al. proposed a Vector of Locally Aggregated Descriptors (VLAD) [3] to capture more information from the image. VLAD and its variations [4, 5, 6] are viewed as a type of simplified Fisher Vector, and it accumulates the difference of each local feature to the visual words and concatenates these accumulated values to describe an image.

Visual word based approaches are challenging to scale to very large image databases, as they have significant computational and memory requirements. Hashing techniques, such as iterative quantization (ITQ) [137], locality-sensitive hashing (LSH) [138], spectral hashing (SH) [23], spherical hashing (SpH) [139], locality-sensitive hashing from shift-invariant kernels (SKLSH) [20], density sensitive hashing (DSH) [140] as well as PCA-random rotation (PCA-RR) [137] focus on learning compact yet powerful image representations for efficient large scale visual search. The basic idea of hashing-based approaches is to construct a hash function to map each visual object into a binary string code such that similar visual objects are mapped into similar binary codes. Unlike the above mentioned hashing approaches, which seek a linear function to project data into a binary vector, recent supervised hashing methods based on convolutional neural network (CNN) architectures [141, 142] seek to learn multiple hierarchical non-linear transformations to generate distinctive binary codes. However, most state-of-the-art hash function learning methods require additional training for each new image domain. This can require significant resources both for assembling the supervised training data and the learning process.

The recent CNN based image representation makes use of the transfer property of a CNN architecture that is pre-trained on a large scale dataset. It has been shown to provide a highly discriminative descriptor representing an image and to produce superior performance in various computer vision tasks, such as image classification, object detection and visual search [58, 143, 144, 145, 146]. Most of these research projects utilize the outputs from the fully connected layers to represent images (directly used or followed by PCA reduction [63]). In particular, visual representations from activations of deep convolutional layers have been shown to lead to high accuracy for image retrieval in real world image test sets. This is achieved by processing a max-pooling, spatial max-pooling [64, 147] or sum-pooling [65] operation on the deep convolutional layers. Better performance is obtained using deep convolutional features than if the features from the fully connected layers are used. This is mainly due to the fact that the activations in each channel of the convolutional layer correspond to receptive fields in the original image, i.e., having a direct semantic interpretation.

Inspired by the advantages of image representation through aggregating activations from deep convolutional layers, we propose a novel and efficient approach to construct bit-scalable binary codes from deep convolutional layers for highly efficient image retrieval (as shown in Figure 4.1). This idea is mainly based on the fact that similar visual objects have similar distributions of responses of feature maps on deep convolutional layers. In this chapter, we propose to generate the binary code on each convolutional layer according to the comparison between the response of each feature map and the average response across all the feature

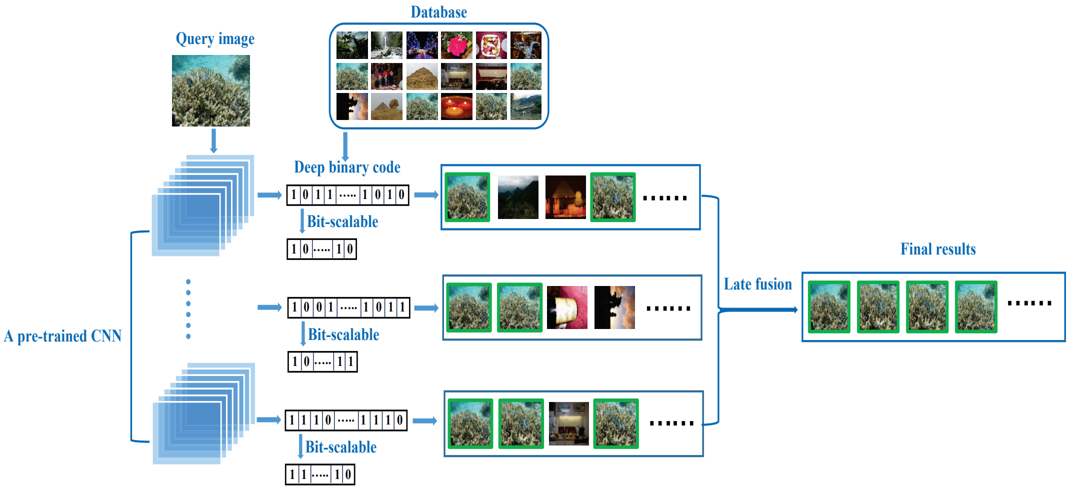

Figure 4.1: The proposed image retrieval framework. Our method consists of two main components. The first is the deep binary code generation on each deep convolutional layer of a pre-trained CNN. In the second component, we propose a dynamic late fusion scheme to further increase the search precision. Images with green rectangles are positive results.

maps on the same deep convolutional layer. Additionally, a strategy of spatial cross-summing is designed to generate bit-scalable deep binary codes. Extensive experiments on well-known image retrieval benchmarks demonstrate the effectiveness of the proposed binary code representation (referred to as deep binary codes) and show competitive results compared to state-of-the-art image retrieval approaches.

The strengths of the proposed deep binary codes are three-fold. First, the deep binary code is highly efficient regarding computational and memory costs. By passing a test image through a pre-trained CNN architecture, the compact binary codes on each deep convolutional layer can easily be generated. Second, the length of a deep binary code can be controlled by the spatial cross-summing operation. Third, available pre-trained CNN architectures (VGGNet [51], AlexNet [48] as well as GoogleNet [42]) can be directly employed to generate deep binary codes.

It is worth to note that during the procedure of passing an image through a pre-trained CNN architecture, all the deep binary codes from lower to higher layers can be obtained. The similarity scores given by deep binary codes from different layers vary largely. As illustrated in Figure 4.3, for a specific query image, the average precision score of each deep binary code is different, and it is difficult to determine in advance which deep binary code is the most robust one. Thus, we are motivated to investigate how to fuse the search scores returned by deep binary codes from different layers, to further improve the precision of visual search. Inspired by the idea proposed by Zhang et al. [148] which demonstrates that the score curve returned by a good feature shows an "L" shape, while that returned by a bad feature shows a gradually dropping tendency, the effectiveness of a feature can be estimated, as it is negatively related to the size of the area under the normalized and sorted score curve. In this chapter, we propose to optimize the operation of normalization in Zhang et al.'s method and design a new unsupervised dynamic late fusion scheme to choose the top N good features for a given query, and then aggregate the search scores of the top N candidates to improve the search precision.

The main contributions of this chapter are summarized as follows:

First, this chapter introduces a novel and compact deep binary representation which is generated from the convolutional layers of pre-trained CNN architectures and investigates the reasons underlying its success. The proposed approach creates bit-scalable deep binary codes in a data-independent manner in the sense that it uses a generic transferred model, which does not require additional training.

Second, image representations based on different pooling operations (such as max-pooling, average-pooling and sum-pooling) as well as various hashing function learning methods on the activations of the deep convolutional layers are evaluated. This results in both insights and a baseline for large scale image retrieval.

Third, the proposed adaptive and unsupervised dynamic (top N) score-level late fusion scheme is shown to significantly improve the image retrieval accuracy.

The remainder of this chapter is organized as follows. First, we briefly review related work in Section 4.2. Section 4.3 introduces the details of the proposed deep binary codes and the dynamic score-level late fusion scheme. The experimental results are presented in Section 4.4 and conclusions are given in Section 4.5.

4.2 Related Work

CNN based image representation: deep learning aims to learn higher semantic representations by passing an image into the architecture of a convolutional neural network [48]. The image features generated from the activations of the fully connected layers have already demonstrated their good performance in image retrieval tasks. Recent research projects further explore the features from the deep convolutional layers. Ng et al. [149] treated the channels from one deep convolutional layer as visual words and encoded them into a feature similar to a VLAD. Razavian et al. [64] and Azizpour et al. [147] proposed to aggregate the activations from the last convolutional layers by max-pooling or spatial max-pooling which show better performance than those from fully connected layers. Additionally, it was revealed that the image representation by sum-pooling and PCA whitened on the last convolutional layer leads to much better performance [65]. Giorgos et al. [150] proposed to extract a set of features by max-pooling at multiple scales on a deep convolutional layer and subsequently summing the collected features to describe an image.

Learning based hashing: the existing hash function learning methods can be classified into two categories: data-independent and data-dependent. LSH [138] is a representative data-independent method which proposed to use random projections to map data into binary codes. SKLSH [20] is an extension of LSH which extends Euclidean distance to other distances. For the data-dependent categories, the method of SH [23] was presented to obtain balanced binary codes by solving a spectral graph partitioning problem. ITQ [137] creates binary codes by simultaneously maximizing the variance of each bit and minimizing the quantization error. SpH [139] was proposed to preserve the data locality relationship to keep neighbors in the input space as neighbors in the Hamming space. CNN based hashing methods with supervisory information in the form of class labels have been further developed by Lai et al. [141] and Zhao et al. [142], which optimize the CNN architecture based on a loss function to preserve binary semantic similarity of the data. Compared with the hashing-based learning methods, the proposed deep binary codes achieved competitive and even better performance without requiring training data and supervisory information.

Late fusion approaches: Late fusion approaches fuse the search results from different features or different methods to increase the search accuracy. Nandakumar et al. [151] proposed a framework which optimally combines the genuine match scores through the likelihood ratio calculation. Zhang et al. [152] proposed a graph-based query specific fusion method where multiple retrieval lists obtained by different methods are merged and re-ranked by a graph model. Zheng et al. [148] proposed to determine the weight of different search scores based on the fact that the quality of a feature has a negative relationship to the area under the normalized score curve. Our proposed method is similar to [148], however the difference is that our late fusion approach only combines the search scores from the top N high quality features, without requiring expensive offline calculations for different features.

4.3 Proposed Approach

4.3.1 Generating Deep Binary Codes

In this section, we describe how to generate the deep binary codes. This starts with a pooling operation, which calculates a summary statistic (such as max-pooling, sum-pooling, multi-scale-max-pooling or multi-scale-sum-pooling) over a local spatial region on the deep convolutional layers. The main motivation behind the use of pooling is to promote invariance to local input transformations (such as translation, occlusion and truncation of the local stimulus), which could greatly improve the effectiveness of the deep convolutional layer representation. This is due to fact that the resulting outputs by pooling are invariant to their spatial location within the pooling region. This is particularly important for the

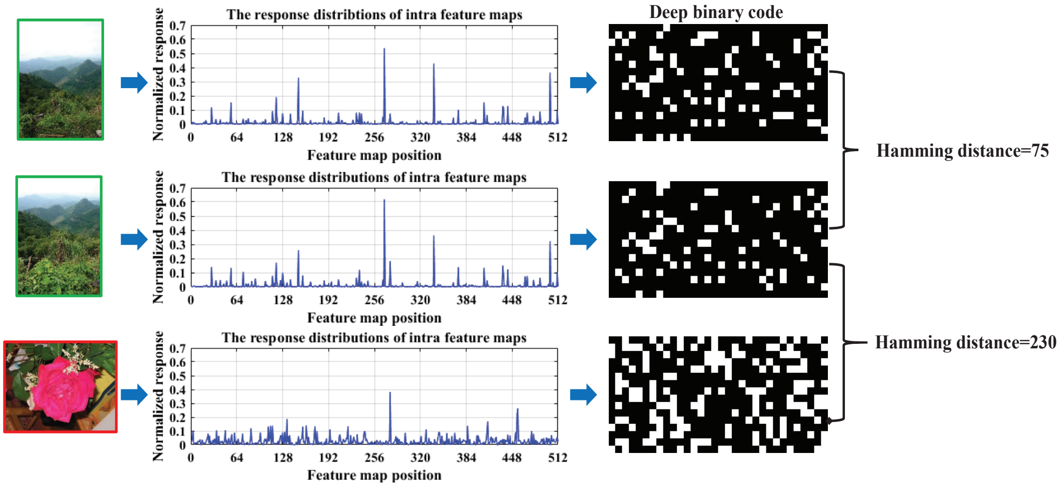

Figure 4.2: The intra feature map distribution between relevant images and irrelevant images. Relevant images have similar distributions and the peak values appear at the same positions, which results in similar deep binary codes.

performance of image search where local image transformations obfuscate object identity. An additional advantage of the pooling operation is that it reduces the spatial resolution, resulting in a lower-dimensional image representation.

Consider a pre-trained CNN architecture with L deep convolutional layers, and a given input image I. We pass I through the pre-trained network where the obtained feature maps can be denoted as $\bar{F}_i = \{F_{i,j}\}$ with $i = 1...L, j = 1...C_i$, where the $F_{i,j}$ is equal to the j^{th} feature map at the i^{th} deep convolutional layer and C_i is equal to the number of channels (or convolutional kernels) of deep layer i. Assume that $F_{i,j}$ has size $W_i \times H_i \times C_i$, where W_i and H_i are the width and height of each channel, respectively. We further associate each cell in the feature map from the i^{th} layer with a spatial coordinate (x, y) and the response at this position $f_i(x, y)$. Then, the image representation by max-pooling on a deep convolutional layer can be described as follows:

$$\mathbf{\bar{V}_i} = [\bar{V}_{F_{i,1}}...\bar{V}_{F_{i,j}}...\bar{V}_{F_{i,C_i}}] where\ \bar{V}_{F_{i,j}} = \max_{x,y \in F_{i,j}} (f_i(x, y)) \tag{4.1}$$

The max-pooling operation encodes the local maximum response from each feature map and leads to a compact feature vector with its dimension equal to the number of feature maps.

In contrast to max-pooling which only makes use of the local maximum response in the feature map, sum-pooling encodes all the responses into the feature vector. Sum-pooling on activations from a deep convolutional layer can be calculated as follows:

$$\hat{\mathbf{V}}_{\mathbf{i}} = [\hat{V}_{F_{i,1}} ... \hat{V}_{F_{i,j}} ... \hat{V}_{F_{i,C_i}}],\ \hat{V}_{F_{i,j}} = \sum_{x=1}^{H_i} \sum_{y=1}^{W_i} f_i(x,y) \tag{4.2}$$

As the activations from the convolutional layers can be interpreted as local features corresponding to particular original image regions, the simple max-pooling and sum-pooling do not consider the spatial and location information of the activations in the feature map, hence the generated feature vectors are only translation invariant. Furthermore, as the local regions appear at various scales in the images, the scheme of multi-scale-pooling on feature maps is utilized to capture information at different scales. In this way the image representation could be robust to scale transformations. Let R denote a region in a feature map, then the extracted feature in this region by pooling can be constructed as follows:

$$\mathbf{V}_{\mathbf{i,R}} = [V_{F_{i,1,R}} ... V_{F_{i,j,R}} ... V_{F_{i,C_i,R}}], V_{F_{i,j,R}} = \underset{x,y \in R}{P} \mid f_i(x,y) \mid \tag{4.3}$$

$$\mathbf{V}_{\mathbf{i}} = [V_{F_{i,1}} ... V_{F_{i,j}} ... V_{F_{i,C_i}}],\ V_{F_{i,j}} = \sum_{R \in F_{i,j}} \mathbf{V}_{\mathbf{i,R}} \tag{4.4}$$

The function $P \mid \cdot \mid$ can be max-pooling, sum-pooling or average-pooling on the region R. R is a square region of the feature map with width (height) from 1 to $\min(W_i, H_i)$. The extracted features from multiple scale regions are then summed, and subsequently l_2-normalized to represent the image.

We further observe that: for a pair of similar images, the feature maps with a high response appear at almost the same index positions on the deep layer (referred to as intra feature map distribution), as shown in Figure 4.2. Based on this observation, we propose to convert the image representation on the deep convolutional layers into binary codes $\mathbf{B_i} = [B_{F_{i,1}} ... B_{F_{i,j}} ... B_{F_{i,C_i}}]$. This binary code is constructed by comparing the response from each feature map with the average response across all the feature maps:

$$B_{F_{i,j}} = \begin{cases} 1, \ if \ V_{F_{i,j}} \geq average(\mathbf{V_i}) \\ 0, \ if \ V_{F_{i,j}} < average(\mathbf{V_i}) \end{cases} \tag{4.5}$$

Thus, the image representation by the deep convolutional layers is converted into binary codes in an unsupervised and training data independent way. Moreover, these binary codes have low memory requirements and allow for fast matching using the Hamming distance, hence binary codes are very suitable for large scale image search.

4.3.2 Spatial Cross-Summing

Based on the pre-trained CNN architecture, the bit-length of the deep binary codes is preset according to the number of feature maps in the deep layers. In real-world applications, binary codes with different bit-lengths allow researchers to make trade-offs between accuracy and efficiency. For example, real-time systems and devices with limited computational and storage resources require low dimensional binary codes, while higher dimensional binary codes are more appropriate for increased accuracy. The conventional PCA-operation is data-dependent and is not suitable for the generation of bit-scalable deep binary codes. To address these issues, we propose a spatial cross-summing strategy to create compact and bit-scalable deep binary codes from deep-layer features.

For a given deep-layer feature $\mathbf{V_i}$ with length C_i, the objective is to generate a bit-scalable deep binary code with length n, $n = C_i/2^m$, and $m = 1, \ldots, \log_2(C_i)$. This procedure starts by generating the deep-layer feature with length n by a spatial cross-summing strategy. For example, let $n = C_i/4$, then $\mathbf{V_i^{2n}} = \mathbf{V_i}[1, 2, ..., 2n] + \mathbf{V_i}[C_i, C_i - 1, ..., 2n + 1]$, and $\mathbf{V_i^n} = \mathbf{V_i^{2n}}[1, 2, ..., n] + \mathbf{V_i^{2n}}[2n, 2n - 1, ..., n + 1]$. Finally, the deep binary code $\mathbf{B_i}$ is calculated using Formula (4.5) on the vector $\mathbf{V_i^n}$. Algorithm 1 formalizes the procedure of bit-scalable deep binary code generation.

Algorithm 1 : Bit-scalable deep binary codes generation

```
Input: the $i^{th}$ deep-layer image feature: $\mathbf{V_i}$ with length $C_i$, and $n = C_i/2^m$, gives
    $m \in N \geq 1$
Output: deep binary codes $\mathbf{B}$ with $n$ bits
$X \Leftarrow C_i/2$, $\mathbf{V} \Leftarrow \mathbf{V_i}$
while $X \neq n$ do
    $bit \Leftarrow X$, $l \Leftarrow length(V)$
    $\mathbf{V^a} \Leftarrow [V_1, V_2, \cdots, V_{bit}], \mathbf{V^b} \Leftarrow [V_l, V_{l-1}, \cdots V_{l-bit+1}]$
    $\mathbf{V'} = \mathbf{V^a} + \mathbf{V^b}$
    $X \Leftarrow X/2$, $\mathbf{V} \Leftarrow \mathbf{V'}$
end while
Deep binary code $\mathbf{B}$ generation of size $n$ bits
for all $V_{bit}$ in $\mathbf{V}$ and $B_{bit}$ in $\mathbf{B}$ do
    if $V_{bit} \geq average(\mathbf{V})$ then
        $B_{bit} \Leftarrow 1$
    else
        $B_{bit} \Leftarrow 0$
    end if
end for
return  $\mathbf{B}$ of size $n$ bits
```

4.3.3 Dynamic Late Fusion

Another advantage of the proposed binary string representation is that the deep binary codes on different layers could all be generated by passing the input image through the pre-trained CNN just once, without additional re-feeding operations. Moreover, different deep binary codes make different contributions to the image search. As illustrated in Figure 4.3, the deep binary code from the conv5 layer gives a higher average precision score than that from the pool5 layer. Thus, the critical issue is how to automatically measure and compare the quality of each deep binary code, since no supervision and relevance feedback are available online, and the only accessible information is the search scores returned by different deep binary codes. Therefore, we aim to exploit these search scores to improve the

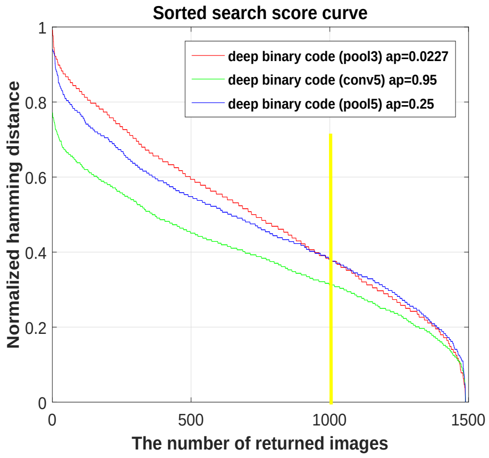

Figure 4.3: The deep binary codes from pool3, conv5 and pool5 are employed to obtain three sorted search scores respectively, where the code from conv5 produces good performance (AP =0.95) and has a smaller area under the curve than those from pool3 and pool5. Note that the curve from pool3 goes beneath that from pool5 after the marked yellow line (which needs to be avoided).

retrieval performance.

The authors of [148] show that the curve of a sorted search score returned by a good feature appears to have an “L” shape and the curve returned by a bad feature shows a gradually decreasing tendency. Furthermore, they showed that the size of the area under the sorted score curve can be used as an indicator to identify the quality of the features. Motivated by this, we fuse the search scores from the top N good deep binary codes.

For a specific query image I, together with a set of deep binary codes $\mathbf{B_i}$, $i = 1...L$, we can use the Hamming distance to measure similarity. Note that in case of the Hamming distance higher similarity corresponds to a lower value. We use a modified Hamming distance $\bar{H}_I = K - H_I$ such that $\bar{H}_I$ has a higher value for higher similarity. Here K is the size of the deep binary code and the sorted search score based on the modified Hamming distance returned by one deep binary code is represented by S_i. We further use max-min normalization on the sorted search scores returned by the modified Hamming distance, so that relevant images for a query give a max score equal to 1, while irrelevant images give a score of 0.

$$\bar{S}_i = \frac{S_i - min(S_i)}{max(S_i) - min(S_i)} \tag{4.6}$$

The size of the area under the curve $\bar{S}_i$ is calculated as:

$$area_i = \sum_{j=1}^{M} \bar{S}_{i,j} \tag{4.7}$$

where M denotes the top M nearest neighbors in each search score. We introduce the parameter M,to prevent the situation where the sorted curve from a good feature may go under that from a bad search score for a large M (as shown in Figure 4.3, the marked yellow line). This parameter controls the size of the area, and it is set as 400 in the experiments. Clearly, the calculated size of the area under each normalized score curve can be used to select the top N high quality features. We further assign an adaptive weight value to each of the top N scores:

$$weight_i = \frac{1}{area_i} \tag{4.8}$$

Finally, the fused search score from high quality deep binary codes is calculated as follows:

$$Score = \sum_{i=1}^{N} (S_i \times weight_i) \tag{4.9}$$

The proposed dynamic (top N) score-level late fusion scheme is adaptive and the quality of the deep binary code is automatically measured in an unsupervised manner. Clearly, it does not need any offline computation, thus the late fusion scheme is compatible with dynamic databases and suitable for large scale image search.

4.4 Experiments and Setup

In this section, we construct experiments and present the performance of our proposed image representation based on deep binary codes as well as the dynamic late fusion scheme in image retrieval.

The VGG network (VGGNet) [51] is employed to generate the deep binary codes (without fine-tuning). The deep convolutional layers: pool3, pool4, conv5 and pool5 from the VGGNet architecture are examined and the activations extraction from each deep convolutional layer is implemented using Caffe [153]. All images are resized to 224×224 before passing through the CNN network. The dimensions and the number of feature maps of the examined deep layers are summarized in Table 4.1. Max-pooling, sum-pooling, multi-scale-max-pooling, and multi-scale-sum-pooling are utilized to transform the activations from the feature maps to deep convolutional features, which are referred to as MP, SP, MSMP and MSSP, respectively. The deep binary codes are accordingly referred to as BMP, BSP, BMSMP and BMSSP. The cosine similarity measure is used to compare two images represented by their deep convolutional features (floating point values), while the Hamming distance is employed to compare the similarity based on the proposed deep binary codes (a binary string).

The experimental environment for the evaluation is a computer with an i7 CPU, 64GB of RAM, and an NVIDIA K40.

The source code of our bit-scalable deep binary codes and dynamic late fusion are released online at: http://press.liacs.nl/researchdownloads/.

Convolutional layer	The size of feature map	The number of feature maps
pool3	28×28	256
pool4	14×14	512
conv5	14×14	512
pool5	7×7	512

Table 4.1: Overview of the deep convolutional layers.

4.4.1 Datasets

We evaluate the performance of the deep binary code and the dynamic late fusion scheme on four publicly available datasets: INRIA Holidays [154], Oxford5K [122], UKbench [40] and MIRFLICKR 1M [136].

INRIA Holidays: this dataset consists of 1491 personal holiday photos that can be divided into 500 image groups, where the first image of each group is the query. The retrieval performance is measured in terms of mean Average Precision (mAP).

Oxford5K: this is a dataset composed of 5062 images which are downloaded from Flickr by searching 11 buildings or landmarks associated with Oxford. There are a total of 55 queries corresponding to 11 buildings and the performance is measured using mAP over the queries.

UKbench: a total of 10200 images are contained in this dataset, divided into 2550 groups. Each image is taken as the query in turn. The performance is measured by the average recall of the top four ranked images, referred to as N-S score.

MIRFLICKR 1M: this dataset includes one million images which are randomly retrieved from Flickr. We use this dataset to test the scalability of our deep binary code.

4.4.2 Evaluation of Deep Convolutional Feature Representation

We first evaluate the performance using deep convolutional representations (MP, SP, MSMP and MSSP), where the feature vectors generated by the operations of MP, SP, MSMP and MSSP are l_2-normalized. Table 4.2 summarizes the performance of the deep convolutional representations on each examined layer. For the single-scale pooling process, we observe that the sum-pooling operation achieves a better performance than max-pooling, while the multi-scale pooling scheme outperforms the single-scale operation. In general, the scheme of MSMP obtained the best search scores on each of the four benchmark datasets. It is also worth noting that the search precision from the lower layers to higher layers reveals an increasing trend, and the deep convolutional features on pool5 outperform features taken from other layers. This is mainly because each activation from higher deep layers correspond to a larger local region in the original image than those from a lower deep layer, hence more semantic information is represented at a high deep layer.

Dataset	Deep feature	pool3	pool4	conv5	pool5
Holiday dataset (mAP score)	MP	57.24	71.40	78.50	79.27
	SP	65.23	75.49	76.37	79.17
	MSMP	67.22	76.91	80.24	**80.65**
	MSSP	64.40	74.80	76.18	78.81
Oxford5K dataset (mAP score)	MP	22.67	31.70	46.38	49.03
	SP	31.68	47.28	54.73	56.55
	MSMP	33.74	49.39	57.39	**58.05**
	MSSP	32.57	50.29	54.48	57.18
UKbench dataset (N-S score)	MP	2.7	3.34	3.72	3.73
	SP	2.95	3.47	3.7	3.73
	MSMP	3.04	3.54	3.74	**3.75**
	MSSP	2.94	3.46	3.66	3.7

Table 4.2: The performance of various deep convolutional features on image retrieval on the benchmark datasets. The accuracy is measured by the mAP score for the Holiday and Oxford5K datasets and the N-S score for the UKbench dataset.

4.4.3 Performance of Deep Binary Codes

We further test the image retrieval accuracy of the proposed deep binary codes on the benchmark datasets and the results are displayed in Table 4.3. The results demonstrate the effectiveness of the deep binary codes. We can see that the deep binary codes from the same layer generated using SP, MSMP and MSSP have similar performance on each dataset, and they all give better results than the representation based on MP. Compared to the performance of the deep convolutional features in Table 4.2, the dimensions of deep binary code are significantly reduced from 256 float values (2048 bytes of memory) to 256 bits (32 bytes of memory) on pool3 layer and 512 float values (4096 bytes) to 512 bits (64 bytes) on pool4, conv5, and pool5 layers, respectively. Meanwhile, the computation-time cost of the cosine similarity between two deep convolutional features (512 float values) is 0.14ms, while the comparison of the Hamming distance measure between two deep binary codes (512 bits) costs 0.007ms computation-time. The performance of deep binary codes is very competitive to deep convolutional features on the Holiday and UKbench datasets, which verifies that the deep binary codes have significant advantages with respect to speed/storage trade-off over the deep convolutional features, especially in the case of large scale image search.

Dataset	Deep feature	pool3	pool4	conv5	pool5
Holiday dataset (mAP score)	BMP	45.02	63.32	70.98	71.05
	BSP	59.47	73.04	74.52	**75.5**
	BMSMP	60.47	73.79	74.83	74.69
	BMSSP	57.78	73.82	72.94	74.65
Oxford5K dataset (mAP score)	BMP	21.4	33.93	43.13	42.59
	BSP	29.78	46.54	49.26	49.92
	BMSMP	29.1	46.68	48.93	49.55
	BMSSP	29.3	47.26	50.33	**50.45**
UKbench dataset (N-S score)	BMP	2.26	3.1	3.54	3.56
	BSP	2.83	3.41	3.62	3.64
	BMSMP	2.85	3.45	3.63	**3.64**
	BMSSP	2.84	3.42	3.59	3.62

Table 4.3: The performance of various deep binary codes on image retrieval based on four benchmark datasets. The accuracy is measured by mAP score for the Holiday and Oxford5K datasets and N-S score for the UKbench dataset.

4.4.4 Comparison with Hashing Learning Approaches

In the research literature, the closest related competitive algorithms are the unsupervised hashing learning methods [20, 23, 137, 138, 139, 140]. Specifically, to make a trade-off towards accuracy, efficiency and storage requirements in large scale image retrieval, hash function learning methods map deep convolutional features to binary string representations. In this section, we evaluate the performance of bit-scalable deep binary codes by comparing them with seven unsu-

Method	Holiday dataset (mAP)											
	pool3 (bits)			pool4 (bits)			conv5 (bits)			pool5 (bits)		
	256	128	64	512	256	128	512	256	128	512	256	128
BMSMP	**59.86**	**54.17**	**42.33**	**73.91**	**71.42**	**64.32**	**74.79**	**70.77**	**65**	**75.32**	**72.22**	**64.67**
LSH	44.48	34.13	23.64	69.42	59.93	53.33	71.57	66.75	58.49	69.47	67.69	57.82
SKLSH	37.2	25.74	23.37	64.62	53.93	39.32	67.78	60.7	46.56	66.97	60.53	47.17
ITQ	39.73	27.63	15.7	71.99	64.2	52.07	74.26	70.39	63.72	74.5	72	64.15
PCAH	12.05	18.11	26.4	37.98	48.3	48.17	46.56	62.48	62.96	44.14	60.49	62.65
SH	57.9	52.81	41.39	71.52	69.61	63.96	72.48	70.1	64.75	71.29	72.05	64.44
PCA-RR	43.98	28.78	30.82	68.29	64	53.13	73.17	69.69	64.02	72.71	70.66	64.05
DSH	53.02	46.24	37.84	63.9	60.7	53.5	66.32	60.84	55.5	67.99	63.86	57.31

Table 4.4: Comparison with various unsupervised hash function learning methods on the Holiday dataset.

Method	Oxford dataset (mAP)											
	pool3 (bits)			pool4 (bits)			conv5 (bits)			pool5 (bits)		
	256	128	64	512	256	128	512	256	128	512	256	128
BMSMP	**29.1**	**22.6**	**19.06**	**46.68**	**40.71**	**35.41**	**48.93**	**48.02**	**41.45**	49.55	47.36	40.51
LSH	21.35	15.84	4.21	40.57	30.33	23.4	47.2	46.07	39.85	48.72	44.78	33.51
SKLSH	16.32	14.04	7.52	39.45	26.88	23.98	46.47	38.53	29.75	**50.15**	39.96	33.6
ITQ	12.44	7.17	6.55	42.36	34.67	21.98	48.42	47.99	41.29	48.8	47.3	40.99
PCAH	10.6	12.13	7.01	20.27	23.9	27.08	33.17	37.77	37.58	35.08	41.54	38.45
SH	24.31	22.25	17.01	44.73	40.21	34.84	48.64	47.31	41.44	49.17	**48.4**	**42.84**
PCA-RR	17.56	16.76	13.01	38.59	31.4	27.39	48.9	47.91	39.53	49.14	47.07	40.49
DSH	21.89	20.22	17.75	32.2	28.25	2.43	40.78	37.58	30.6	43.32	38.06	28.78

Table 4.5: Comparison with various unsupervised hash function learning methods on the Oxford5k dataset.

pervised hash function learning methods. The compared approaches include two categories: data-independent methods (LSH and SKLSH) and data-dependent methods (ITQ, PCAH, SH, PCA-RR and DSH). The implementation of these methods are provided by the authors. Considering that the image representation based on MSMP achieves the best performance (as the results demonstrated in Table 4.2) and in order to make an objective comparison, all evaluated hashing learning methods map the deep convolutional features generated by using the MSMP operation. Moreover, different sizes of binary representations are evaluated.

Table 4.4, 4.5 and 4.6 illustrate the search accuracy of all the evaluated approaches on the benchmark datasets with different numbers of bits. We observe that deep binary codes from deep convolutional layers pool3, pool4 and conv5 give better results than the other hashing learning methods. The deep binary codes with different bit sizes from all examined layers obtained the best results on both the Holiday and UKbench datasets. The deep binary codes with 512, 256 and 128

Method	UKbench dataset (N-S score)											
	pool3 (bits)			pool4 (bits)			conv5 (bits)			pool5 (bits)		
	256	128	64	512	256	128	512	256	128	512	256	128
BMSMP	**2.6**	**2.6**	**2.18**	**3.46**	**3.31**	**2.66**	**3.64**	**3.47**	**3.2**	**3.65**	**3.48**	**3.21**
LSH	2.39	1.76	1.16	3.31	3.0	2.41	3.50	3.32	2.97	3.52	3.32	2.94
SKLSH	2.18	1.74	0.84	3.15	2.76	2.18	3.39	3.10	2.51	3.38	3.04	2.58
ITQ	1.78	0.89	0.33	3.22	2.82	2.19	3.54	3.4	3.11	3.55	3.41	3.12
PCAH	1.59	1.61	1.48	2.77	2.64	2.35	3.42	3.38	3.19	3.41	3.38	3.18
SH	2.6	2.58	2.12	3.43	3.21	2.65	3.58	3.45	3.11	3.59	3.45	3.18
PCA-RR	2.40	2.06	1.45	3.31	3.08	2.63	3.54	3.43	3.19	3.56	3.42	3.15
DSH	2.42	2.18	1.92	3.03	2.84	2.61	3.41	3.21	2.91	3.41	3.25	2.99

Table 4.6: Comparison with various unsupervised hash function learning methods on the UKbench dataset.

bits on pool5 show competitive performance to SKLSH and SH on the Oxford5K dataset. Regarding the data-dependent hash function learning approaches, the computational complexity and the time-cost will be significantly increased when the amount of training data becomes large. The deep binary code does not suffer from this issue because it does not need retraining. Furthermore, it shows its competitiveness and in some cases even better performance on image search compared to the hash function learning approaches.

4.4.5 Evaluation of the Late Fusion Scheme

In this section, we verify the effectiveness of the proposed dynamic top N score-level late fusion approach. Both binary string features and float value features are evaluated.

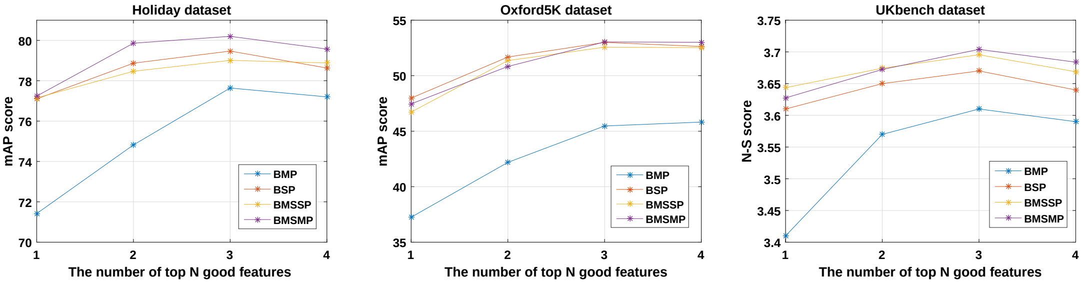

Figure 4.4: The search accuracy for different values of N. Four search scores from each of the compared methods are used, and most of them obtain the best fused score at value 3.

Method	Holiday dataset (mAP score)		Oxford5K dataset (mAP score)		UKbench dataset (N-S score)	
	Best	Fused	Best	Fused	Best	Fused
BMSMP	**75.32**	**80.3**	49.55	53.04	**3.65**	**3.71**
LSH	71.57	77.17	48.72	51.53	3.52	3.63
SKLSH	67.78	75.27	**50.15**	**54.27**	3.39	3.59
ITQ	74.5	77.85	48.8	50.75	3.55	3.61
PCAH	46.56	55.54	35.08	39.15	3.42	3.59
SH	72.48	79.52	49.17	54.17	3.59	3.68
PCA-RR	73.17	77.84	49.14	52.43	3.56	3.65
DSH	67.99	72.52	43.32	45.12	3.41	3.45

Table 4.7: The comparison of each evaluated method on image retrieval accuracy with and without top N score-level late fusion ($N = 3$).

The impact of the parameter N. First, we construct experiments to validate the influence of parameter N introduced in Formula (4.9). The normalized and sorted search scores from the deep binary codes generated by the operations of sum-pooling, max-pooling, multi-scale-sum-pooling and multi-scale-max-pooling on deep convolutional layers pool3, pool4, conv5 and pool5 are used. The dynamic late fusion for the deep binary codes is based on Formula (4.9) and the search accuracy on each test dataset is depicted in Figure 4.4. We find that the fusion accuracy from each search score increases steadily with N, while slightly decreasing at position *top*4. All the fused search scores show peak values at position *top*3, therefore, we set N equal to $\lambda - 1$, where λ is the number of fused features.

Then, we compare the retrieval performance of the binary string representation with the dynamic late fusion framework to the retrieval performance of the binary string representation without the dynamic late fusion framework. The deep binary codes and learned binary codes by hash functions are evaluated, and 256 bits on pool3, 512 bits on pool4, conv5 and pool5 are used in this comparison. The comparison results are shown in Table 4.7, “Best” denotes the best accuracy of each method from deep convolutional layers, “Fused” denotes that the search score is obtained using the dynamic late fusion scheme. We find that the search accuracy is significantly increased after the *top*3 late fusion. The deep binary codes obtain the best fused scores on the Holiday and UKbench datasets and show competitive results on the Oxford5K dataset. Moreover, the fused scores also show competitive performance when compared to deep convolutional features.

Comparison with other fusion schemes. In order to further verify the strength of our late fusion method, we evaluate the dynamic late fusion scheme on some search scores using real valued features and compare the retrieval accuracy with two state-of-the-art late fusion schemes: graph model late fusion [152] and query-adaptive late fusion [148]. The comparison is carried out on the Holiday and UKbench datasets, and using the features of BoW (a 20K visual words histogram generated from rootSIFT [104] local descriptors and the *tf-idf* weight scheme), GIST [155] (a 512-dimensional global GIST descriptor), CNN [153] (a 4096-dimensional feature extracted from the first fully connected layer in the Alex CNN architecture), RAND (a global descriptor generated through multiplying by a random transform matrix) and HS (a 1000-dimensional HSV color histogram), respectively. The implementation of search scores on the Holiday and UKbench datasets from the five category features are offered by [156].

Formula (4.9) fuses the sorted search scores from binary string features using the Hamming distance to measure the similarity. We then modified it in Formula (4.10) to satisfy the distribution of search scores from float value features (BoW, GIST, CNN, HS and RAND) when using the cosine distance to measure the similarity. The N in Formula (4.10) is set to 4, because the number of search scores is 5 and we should set it equal to $5 - 1 = 4$.

$$Score = \prod_{i=1}^{N} (S_i)^{weight_i} \tag{4.10}$$

For graph model late fusion and query-adaptive late fusion, we use the code released from the papers [152] and [148] respectively. In order to make an objective comparison, the parameter M in Formula (4.7) is set to 400 such that it is equal to the corresponding parameter in the query-adaptive late fusion scheme. On the Holidays dataset, our late fusion scheme outperforms graph model fusion and

Method	Holiday dataset (mAP score)	UKbench dataset (N-S score)
Graph model [152]	81.04	3.82
Query-adaptive [148]	87.98	3.84
Ours	**88.61**	**3.84**

Table 4.8: Results on benchmarks with different fusion methods. We compare our method with Graph Fusion [152] and Query-adaptive [148] approaches.

the query-adaptive method. On the UKbench dataset, our result is equal to the query-adaptive method and better than graph model fusion. The comparison results further illustrate that the proposed dynamic top N late fusion method is effective for the search scores from both the binary string representation and the real valued representation.

4.4.6 Performance on Large Scale Image Search

In order to evaluate the performance of deep binary codes on large scale image search, we further perform large-scale experiments by combining the MIR-FLICKR 1M dataset with the Holiday, Oxford5K, and UKbench datasets. The deep binary codes as well as the binary codes learned by hash functions with a bit size of 256 on pool3, and a bit size of 512 on pool4, conv5 and pool5 are utilized for the evaluation. The accuracy results and the average time-cost of learning the hash function are summarised in Table 4.9. On each of the datasets with more than one million images, the deep binary code obtains the best accuracy with and without the dynamic late fusion scheme. This is further showing that the deep binary code is suitable for large scale image search and the dynamic late fusion scheme could significantly improve the search accuracy without requiring offline calculation.

Method	Holiday+1M (mAP)		Oxford5K+1M (mAP)		UKbench+1M (mAP)		Learning time cost
	Best	Fused	Best	Fused	Best	Fused	average(s)
BMSMP	**71.76**	**77.19**	**49.18**	**52.04**	**90.48**	**91.56**	-
LSH	60.68	69.26	44.24	47.3	83.26	87.36	-
SKLSH	55.33	66.99	46.1	51.18	79.79	87.19	-
ITQ	57.47	64.32	43.31	44.75	83.02	86.23	1120
PCAH	64.71	72.45	46.7	49.95	83.92	89.01	125
SH	64.3	73.98	46.08	50.38	86.4	90.28	1500
PCA-RR	61.91	70.27	45.44	49.38	85.44	88.77	190
DSH	52.13	59.13	38.8	39.8	76.98	78.69	400

Table 4.9: Comparison of the accuracy of each evaluated method for large scale image retrieval with and without top N score-level late fusion ($N = 3$), score-level late fusion (N = 3), and the time-cost of learning the hashing function.

Method	#dimensions	Holiday dataset (mAP score)	Oxford5K dataset (mAP score)	Ukbench dataset (mAP score/N-S score)
VLAD+RootSift [4]	128float	62.5	44.8	–/–
VLAD+CSurf [157]	128float	73.8	29.3	83.0/–
mVLAD+Surf [157]	128float	71.8	38.7	87.5/–
FV+T-embedding [158]	128float	61.7	43.3	85.0/–
FV+T-embedding [158]	256float	65.7	47.2	86.3/–
Sum pooling+PCAW [65]	256float	80.2	**58.9**	–/3.65
Max pooling+l_1 dist [64]	256float	71.6	53.53	84.2/–
Deep fully connected [63]	256float	74.9	43.5	–/3.42
Deep fully connected+finetune [63]	256float	78.9	55.7	–/3.56
BMSMP	512bit	74.83	49.55	90.78/3.65
FBMSMP	1792bit	**80.3**	53.1	**92.15/3.71**

Table 4.10: Comparison with state-of-the-art compact image representations on three benchmark datasets. FBMSMP denotes deep binary codes after applying dynamic top N late fusion.

Method	#Dim	Memory cost (Flicker 1M)	Holiday+Flicker 1M (mAP score)	Oxford5K+Flicker 1M (mAP score)	Ukbench+Flicker 1M (mAP score)
VLAD+RootSift[4]	128float	0.48G	37.8	–	–
Geometric+VLAD[6]	128float	0.48G	60.7	43.8	–
BMSMP	512bit	0.06G	71.76	49.18	90.48
FBMSMP	1792bit	0.24G	**77.19**	**52.04**	**91.56**

Table 4.11: Comparison with state-of-the-art compact image representations on large scale dataset. FBMSMP denotes deep binary codes after applying dynamic top N late fusion.

4.4.7 Comparison with state-of-the-art

We then compare the image retrieval results from deep binary codes with some other important state-of-the-art low dimensional image features. The results are from the papers [64] and [65], and the comparison is displayed in Table 4.10 and Table 4.11. Note that, the size of deep binary codes is 256-bit on pool3, 512-bit on pool4, conv5 and pool5. The results show that our deep binary codes outperform hand-crafted image representations, such as VLAD and Fisher Vector, and even outperform some recent CNN-based features. Moreover, after applying the top N late fusion scheme on the deep binary codes, the performance has been further improved.

4.5 Conclusions

In this chapter, we proposed a novel image representation called deep binary codes which have several important advantages over deep convolutional feature representations, as they can be calculated using a generic transferred model and therefore do not require additional training unlike many of the competitive algorithms from the research literature. The experimental results on well-known datasets as well as a large scale dataset show that deep binary codes are competitive to state-of-the-art approaches and can significantly reduce memory requirements and computational costs for large scale image search. Second, the dynamic late fusion scheme estimates the quality of each feature in a query-adaptive manner which highlights the strengths of score-level fusion without needing supervision and offline calculations. In our experiments the dynamic late fusion scheme gave consistent improvements in accuracy.