DeepIndex for Accurate and Efficient Image Retrieval

Yu Liu, Yanming Guo, Song Wu, Michael S. Lew
LIACS Media Lab, Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
{y.liu, y.guo, s.wu, m.s.lew}@liacs.leidenuniv.nl

ABSTRACT

In the well-known Bag-of-Words model, local features, such as the SIFT descriptor, are extracted and quantized into visual words. Then, an index is created to reduce computational burden. However, local clues serve as low-level representations that can not represent high-level semantic concepts. Recently, the success of deep features extracted from convolutional neural networks(CNN) has shown promising results toward bridging the semantic gap. Inspired by this, we attempt to introduce deep features into inverted index based image retrieval and thus propose the DeepIndex framework. Moreover, considering the compensation of different deep features, we incorporate multiple deep features from different fully connected layers, resulting in the multiple DeepIndex. We find the optimal integration of one mid-level deep feature and one high-level deep feature, from two different CNN architectures separately. This can be treated as an attempt to further reduce the semantic gap. Extensive experiments on three benchmark datasets demonstrate that, the proposed DeepIndex method is competitive with the state-of-the-art on Holidays(85.65% mAP), Paris(81.24% mAP), and UKB(3.76 score). In addition, our method is efficient in terms of both memory and time cost.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; I.4.7 [**Image Processing and Computer Vision**]: Feature Measurement

General Terms

Algorithm, Experimentation, Performance

Keywords

Image Retrieval, Convolutional Neural Networks, Bag of Deep Features, DeepIndex

ICMR'15, June 23–26, 2015, Shanghai, China.

http://dx.doi.org/10.1145/2671188.2749300.

1. INTRODUCTION

Image retrieval is one of the most significant and popular research tasks in the multimedia community[18, 22], and aims to quickly search for the similar images through their visual features. Bag-of-Words(BoW) is the most efficient model in existing state-of-the-art image retrieval systems, in which local features, such as the SIFT[19], and the color clues[32], are extracted and quantized to visual words with a pretrained codebook. Then, similar to document retrieval[27, 22], an inverted index is built to reduce computational costs and memory requirements for scalable image search. Recently, Zheng,*et al.*,[35] performed low-level feature fusion with the SIFT and color features using a coupled inverted index level. However, image retrieval remains one of the most challenging problems, which is mainly due to the well-known "semantic gap" challenge that exists between low-level image representations and high-level semantic concepts perceived by human.

Machine learning is treated as one promising technique that attempts to address this grand challenge. Recently, one important technique, called deep learning, has shown significant promise in computer vision and speech recognition. In deep learning, there are a family of machine learning algorithms that extract high-level abstractions in data by employing deep architectures composed of multiple neural layers[5]. An important model in deep learning is convolutional neural networks(CNN)[7], which has achieved top rankings in many international benchmarks for computer vision applications, such as image classification[16, 29] and object detection[9]. Also, many works[31, 4, 24, 6] have started to employ this kind of deep features for image retrieval and aim to evaluate how much retrieval improvement can be achieved by developing the deep learning techniques, and whether deep features are a desirable key to bridge the semantic gap in the long term[31].

Inspired by these successes and attempts, we propose a novel DeepIndex framework that explores deep features for accurate and efficient image retrieval based on the BoW model and inverted index. Firstly, we use the Bag-of-Deep-Features(BDF) model that clusters visual words directly with deep features. The spatial pyramid method is used to partition the image into many patches. Then every patch is equally input to extract activations from the fully connected(fc) layers in one pretrained CNN model, denoted as deep features. All the feature patches of the image dataset are clustered for the 'deep' codebook. In this work, we extract deep features with two widely-used pretrained CNN models that are Alexnet[16] and VGGnet[26]. Based on the different

number of neural layers, we categorize them as 'low' CNN and 'high' CNN, respectively. Secondly, different from other works based on CNN that use resource-consuming image matching and nearest neighbor search for retrieving images, our work, called DeepIndex(DPI), introduces an efficient inverted index as a search strategy for deep features, which can achieve promising performance while significantly reducing the computational cost and memory space. Thirdly, we use multiple deep features to narrow the semantic gap between mid-level and high-level features. Here, we integrate multiple fc layers, and then build a multiple DeepIndex(multi-DPI) structure. Specifically, The 2-D DeepIndex is performed in this paper consisting of two kinds of variants: intra-CNN and inter-CNN. The former one uses two fc layers from the same CNN architecture, such as Alexnet. In contrast, the latter one selects two fc layers from two different CNN architectures, such as Alexnet and VGGnet. It is found that the performance of inter-CNN is better than that of intra-CNN. There are two reasons to explain it. First, it is due to the close relationship of two fc layers from the same CNN model that can not fully develop the advantage of the 2-D inverted index. The second reason is that, for inter-CNN, we select one fc layer in Alexnet, and one fc layer in VGGnet. Alexnet is a kind of 'low' CNN(eight layers), so its fc feature can be treated as some kind of mid-level representation. However, VGGnet belongs to a kind of 'high' CNN with nineteen layers, and thus its fc feature corresponds closer to a high-level representation. Owing to the mutual compensation of mid-level and high-level features in inter-CNN, it obtains better results and bridges the semantic gap with feature fusion at the 2-D inverted index level. However, it is noteworthy that intra-CNN is simpler and faster than inter-CNN. Finally, a special signature for the global image, called global image signature(GIS), is integrated into DPI which enhances the matching precision.

Finally, experimental results demonstrate the advantages of DeepIndex, and the necessity and efficiency of multiple DeepIndex. In several well known, public datasets, our comparisons with the state-of-the-art clearly validate these advantages. The contributions of this paper can be summarized as follows:

- We introduce the DeepIndex framework that uses the inverted index scheme for deep features(this is the first to the best of our knowledge).
- We propose to use multiple deep features with the multiple DeepIndex which improves retrieval accuracy. It consists of two variants: intra-CNN and inter-CNN.
- We further employ the deep feature of the whole image as a signature for improving matching accuracy, which is called the global image signature.

The rest of this paper is organized as follows. Related works are briefly introduced in Section 2. Section 3 describes the bag-of-deep-features scheme. The DeepIndex framework is introduced in Section 4. The experimental results are summarized in Section 5. Finally, Section 6 summarizes the conclusions.

2. RELATED WORK

In this section, we will briefly discuss related works as well as emphasize our differences with them.

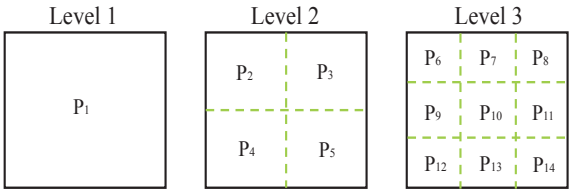

Figure 1: Spatial pyramid with three levels, and there are 14 different patches P_i in total, where $i = 1, \ldots, 14$.

In Gong,*et al.*[10], they proposed to extract patches at multiple scales, and then aggregated local patch responses at the finer scales via VLAD[14] encoding. In Razavian,*et al.*[24], their system first augmented the training set by adding cropped and rotated samples. Then for each image, it used crude search to extract multiple sub-patches of different sizes at different locations. Each sub-patch was computed for its CNN representation. The work of Wan, *et al.*[31] revealed that a deep CNN model pre-trained on a large dataset can be directly used for feature extraction in new CBIR tasks. When being applied for feature representation in new domain, it is found that similarity learning can further boost the retrieval performance. Another work by Sun, *et al.*[28] extracted object-like image patches with a general object detector. Then one CNN feature is extracted in each object patch. Without sliding windows or multiple-scales patches in Babenko,*et al.*[4], they focused on holistic descriptors where the whole image is mapped to a single feature vector. Recently, in Zhang,*et al.*[34], they proposed deep embedding that used deep feature as global and regional signatures instead of a Hamming embedding[11]. In fact, it belongs to an incorporation of the SIFT descriptor and CNN feature, and achieves desirable results. All these works only choose one fully connected feature, either the first or the second one. In Agrawal, *et al.*[1], they analyzed and compared the differences and discriminative abilities of different layers.

In contrast to the prior works, our work focuses on both search strategy and feature fusion at the indexing level. We directly cluster visual words with deep features and then introduce the DeepIndex scheme as the search strategy. Furthermore, we propose building the multiple DeepIndex to utilize multiple deep features which can compensate each other mutually toward improving the retrieval precision.

3. BAG OF DEEP FEATURES

Many state-of-the-arts image retrieval methods employ the low-level features, such as SIFT and color descriptors, and rely on the Bag-of-Features(BoF) or Bag-of-Words(BoW) model. However, few works have shown the usage of deep features into BoF. In this section, we use the Bag-of-Deep-Features(BDF) model, in which visual deep-words are clustered directly on CNN features.

3.1 Spatial Patches

Generally, extracting the only feature vector from the whole image is not discriminative enough for image retrieval tasks, and loses relatively useful information, such as contextual and spatial information. Thus it is quite important and necessary to consider features within finer scales. Generally,

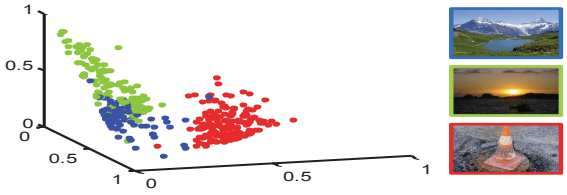

Figure 2: Visualizing the deep features for three groups of images from Holidays. One image from each group is shown in the right side, and the color of outer frame corresponds to the color of points in the 3D space. Best viewed in color.

there are many kinds of methods enlarging the number of features in one image, including "sliding windows", "region proposals", and "spatial pyramid".

Firstly, sliding windows is a quite common approach in object recognition and object detection that scans an image using windows of different scales, locations, and aspect ratios. For example, Gong, *et al.*[10], scanned the whole image with two levels of overlapping windows that generates numerous local patches. Secondly, the region proposals method is proposed to detect the objects of interest in images using fewer candidates than sliding windows. For object detection, RCNN[9] adopts this kind of selective search into CNN replacing sliding windows. In Sun, *et al.*[28], they also extract CNN features only on object-like image patches with a region proposal detector. Thirdly, in contrast to the above two methods, the spatial pyramid model[17] is an efficient way to preserve the spatial information. In Razavian, *et al.*[24], they first augment the datasets by cropping and rotating images in several directions, and then use spatial search to divide the whole image into different levels of patches whose union covers the whole image.

Comparison with three methods mentioned above, we employ the spatial pyramid to refine the image representation because of its simplicity and efficiency. As seen in Fig. 1 , we partition one image into three levels: Level 1 only contains a patch P_1 that is the whole image; Level 2 separates the image into four patches(P_1, P_2, P_3, P_4)whose union covers the whole image; Level 3 consists of nine non-overlapping patches, from P_5 to P_{14}. In total, there are 14 patches for one image, and their features are computed independently. In contrast to methods[24] which uses larger levels for training images than query images, we apply the same spatial levels for all images. Furthermore, we do not perform prior data augmentation, such as crops and rotations.

3.2 Feature Extraction and Quantization

The success of convolutional neural networks used in image classification[16] has shown the strong efficiency and discrimination of learning features by layer-wise architecture. Also, several works[9, 31] suggest that the features located in the upper layers of CNN can serve as good descriptors for various image applications.

It is common to prefer to use the fully connected layers for feature representations[8, 24], because of their closer to class posteriors, but there are also differences about which fc layer is favorable[1].

Different from these works using either the first or the second fully connected activations, we focus on how to incorporate multiple fully connected features. In this work, we employ two commonly CNN architectures pretrained on ILSVRC[25] as feature extractors. The one is proposed by Krizhevsky[16] in 2012 and implemented by the Caffe framework[15]. The other one is from Simonyan[26] in 2014 and released in the MatConvNet[26] framework. We refer to them as Alexnet and VGGnet respectively. In Alexnet, there are eight ordered layers(5 convolutional layers, 3 fully connected layers), thus we name the first and second fc layers as fc6 and fc7. Note that VGGnet has various configurations, and we choose to use the one with nineteen weight layers (16 convolutional layers, 3 fully connected layers), and the first and second fc layers are codenamed as fc17 and fc18.

To visually demonstrate the discrimination of deep features, we select three groups of images from the Holidays dataset[11]. The fc18 features for patches of these images are computed and mapped into a 3D space by classical Multi-Dimensional Scaling[1]. As seen in Fig. 2, the degrees of separations of data points from three groups are promising.

The feature extraction and quantization is conducted as in Fig. 3. Given an image I, x_i represents the feature vector of the ith patch, where $i = 1, \ldots, 14$. After extracting features of all image patches, we perform feature quantization to map deep features into visual deep-words. With the codebook clustered on the training dataset using the k-means algorithm, the quantization function $q(\cdot)$ maps a patch feature x_i to its nearest centroid v_k in the codebook, with $q(x_i) \mapsto v_k$, where v_k is the kth visual word. Note that codebooks for different fc features, including fc6, fc7, fc17, fc18, are trained independently. Typically, the dimension of the fc feature is usually 4096, and L_2 normalization is used for these features.

4. DEEPINDEX

To reduce the computational time and memory, we propose the DeepIndex(DPI) framework in which the inverted index is created based on visual deep-words. Afterwards, we integrate multiple deep features with the multiple DeepIndex. Finally, the global image signature is utilized to elevate the matching accuracy.

4.1 Single DeepIndex

Although several works have exploited deep features for image retrieval tasks, they typically focus on the usage of deep features themselves and the retrieving algorithm they use is simply nearest neighbor search performed by computing matching similarity among image(or patches). Our work also improves the search strategy by using inverted index for deep features.

We create an inverted index structure in which each entry corresponds to a visual deep-word defined in the precomputed codebook. Assume that there are a total of N images in an image database, denoted as $\{I_i\}_{i=1}^{N}$. Each image I_i has a set of patches features $\{x_j\}_{j=1}^{d_j}$, where d_j is the number of patches. Given a codebook $\{v_i\}_{i=1}^{K}$ of size K, we represent the inverted index as $\mathcal{W} = \{W_1, W_2, \ldots, W_K\}$. In $\mathcal{W}$, each entry W_i consists of a list of indexed items, such as image ID, term-frequency(TF) score and other metadata[22, 13, 35].

Given a query deep feature, its corresponding entry W_i is identified by feature quantization mapping. The indexed

[1]Here, we use the Matlab function 'cmdscale'.

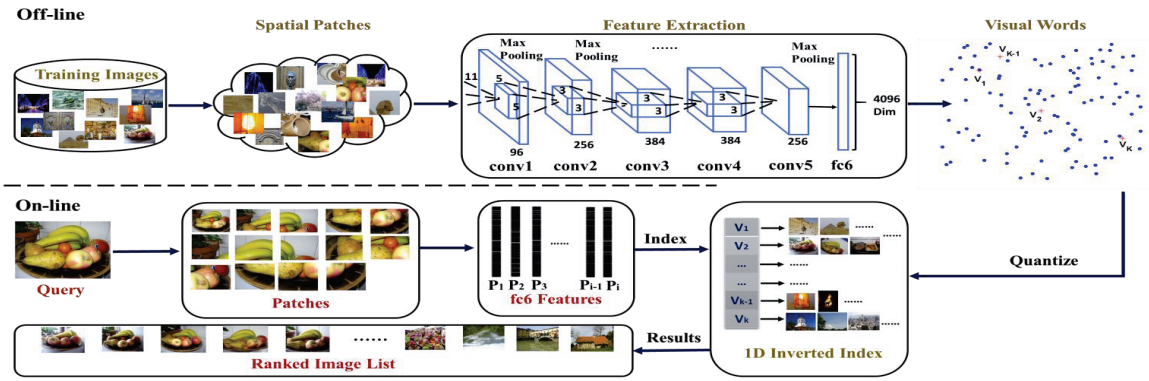

Figure 3: The flowchart of Single DeepIndex framework, including off-line and on-line stages. For example, The pretrained Alexnet serves as the feature extractor and fc6 activations are obtained for features. Image patches in different levels are shown in the same size.

items following entry W_i are counted as the candidate nearest features of the query feature. Therefore, the matching function $h_q(\cdot)$ of two deep features x and y with mapping function $q(\cdot)$ is computed as

$$h_q(x, y) = \delta_{q(x),q(y)}, \tag{1}$$

where δ is the Kronecker delta response. The voting score is accelerated by all $h_q(x, y)$ values.

At this point, the matching function do not consider the $tf-idf$ scheme[27], which weights the visual words according to their frequency. Generally, rare visual words are assumed to be more discriminative and should be assigned higher weights. In this case, the matching function can be updated as

$$h(x, y) = \delta_{q(x),q(y)} \cdot idf(q(y))^2, \tag{2}$$

where $idf(i) = N/n_i$, n_i is the number of images containing the visual word v_i.

We call the proposed indexing scheme as single DeepIndex (1-D DPI) because of using one kind of deep features. This work consists of four kinds of 1-D DPI, which are presented as DPI_6,DPI_7,DPI_{17}, and DPI_{18}. The whole procedure of image search with BDF and 1-D DPI is illustrated in Fig. 3. This figure takes the DPI_6 method for example, and it is also suitable for other deep features. There are two stages: off-line stage and on-line stage. The former one mainly cluster codebook with training patches, and generate the DeepIndex structure. The latter stage is to query one image by searching the inverted index and to return similar images.

4.2 Multiple DeepIndex

Currently, most works mainly focus on comparing performance of different fully connected layers, and only choose the favorable one. However, different neural layers implying different levels of abstracts for the input image demonstrate distinct aspects. It is not preferable to take one layer as the only representation for feature matching. Thus we utilize different layers to compensate each other and to improve the matching accuracy. Based on this idea, we propose to integrate multiple fully connected layers with the multiple DeepIndex(multi-DPI).

The structure of the multi-index was first proposed in Babenko, *et al.*[3]. It decomposes the SIFT descriptor into several blocks by product quantization. The multi-index is thus organized around the codebooks of corresponding blocks. Recently, Zheng, *et al.*[35], built the coupled multi-index with traditional SIFT features and additional discriminative color names. Their results demonstrate that the feature fusion at indexing level is better than the single indexing. Motivated by these works, we aim to exploit incorporation of deep features by multi-DPI structure. In this paper, we take the two dimensional DeepIndex(2-D DPI) as an example.

Considering the 2-D DeepIndex, we denote $\mathcal{X} = [x^r, x^c]$ as a coupled deep features for a patch P_i in Image I, where x^r is extracted from one fully connected layer as row indexing, and x^c comes from another fc layer as column indexing. Then, the row and column codebooks are generated with training images separately, noted by $\mathcal{U} = u_1, u_2, \ldots, u_M$ and $\mathcal{V} = v_1, v_2, \ldots, v_N$, where M and N are codebook sizes. As a result, this 2-D DPI structure contains $M \times N$ entries, as $\mathcal{W} = W_{11}, W_{12}, \ldots, W_{ij}, \ldots, W_{MN}, i = 1, 2, \ldots, M, j = 1, 2, \ldots, N$.

After building the 2-D DPI, all feature tuples like $\mathcal{X} = [x^r, x^c]$ are quantized into visual word pairs (u_i, v_j) using codebooks $\mathcal{U}$ and $\mathcal{V}$, where u_i and v_j are the nearest centroids to features x^r and x^c, respectively. Then other useful clues(e.g. image ID, and other metadata) related to the current feature tuple $\mathcal{X}$ are saved in the corresponding entry W_{ij}, similar to the 1-D DeepIndex.

Now, assume two feature tuples $\mathcal{X} = [x^r, x^c]$ and $\mathcal{Y} = [y^r, y^c]$, the matching function considering 2-D indexing is rewritten as

$$h(\mathcal{X}, \mathcal{Y}) = \delta_{q^r(x^r),q^r(y^r)} \cdot \delta_{q^c(x^c),q^c(y^c)} \cdot idf^2, \tag{3}$$

where $q^r(\cdot)$ and $q^c(\cdot)$ present quantization functions for two different features in row and column spaces, and idf is the 2D extension of the 1D one in Eq.(2). Here, a right match is valid only when the two features tuples are similar in both two indexes. Thus, the 2-D DeepIndex enhances matching strength and improve retrieval precision. Moreover, we define two models about how to select fc features, named intra-CNN and inter-CNN.

Intra-CNN model uses two fc layers from the same CNN architecture. As the two black solid lines seen in Fig. 4, fc6 activation is taken as column indexing, and the fc7 activation

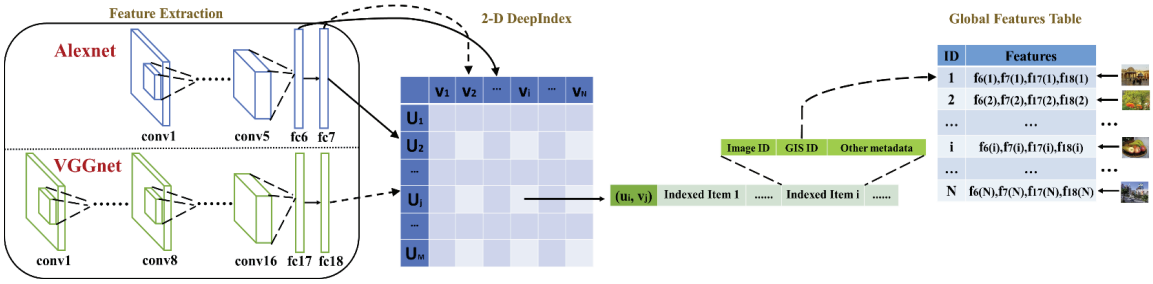

Figure 4: The framework of 2-D DeepIndex for deep features, including intra-CNN and inter-CNN. For intra-CNN, it uses the fc6 and fc7 jointly. For inter-CNN, the fc7 and fc18 are incorporated for indexing. Besides, global image signature serves as additional clue in indexed items, stored in a table.

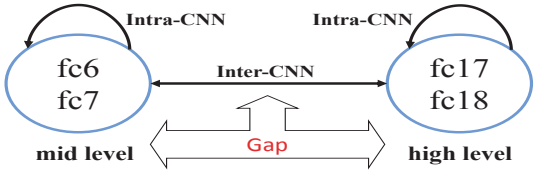

Figure 5: Comparison between the Intra-CNN and the Inter-CNN.

serves as row indexing. In this work, there are two kinds of Intra-CNN members, called $DPI_{6,7}$ and $DPI_{17,18}$.

Inter-CNN model chooses two fc layers coming from two different CNN architectures. For example the two black dot lines in Fig. 4, fc7 from Alexnet('low' CNN) and fc18 from VGGnet('high' CNN) represent column and row indexing, respectively. In total, there are four Inter-CNN members, including $DPI_{7,17}$,$DPI_{6,17}$,$DPI_{7,18}$, and $DPI_{6,18}$.

We draw one more example for Intra-CNN and Inter-CNN in Fig. 5. We categorize the fc6 and fc7 as mid-level features, and fc17, fc18 as high-level features. The Intra-CNN is relatively simple, but the importance and necessity of Inter-CNN is its ability to bridge the gap between mid-level and high-level features to some extent. More comparison and analysis about the performance of intra-CNN and inter-CNN is followed in the experiment of Sec. 5.3.

4.3 Global Image Signature

To further improve the matching accuracy in the inverted index structure, we employ additional discriminative feature to constrain the matching condition and to filter out false matches, which is called the 'signature'. The most popular one is the hamming embedding signature[11] that uses a 64-D binary signature for each SIFT descriptor, and stores it in the metadata of the inverted items. In zheng,*et al.*[35], they use the hamming embedding for SIFT features and generates another signature for color names.

Recently, the global discrimination of deep feature has been demonstrated in many works[24, 31, 1], in which only one fully connected feature vector(4096-D) extracted from the whole image can achieve desirable results. In this paper, we propose to use the deep feature for the whole images as an additional signature for DeepIndex, called global image signature(GIS). Although the spatial patches in Sec. 3.1 already consist of the global feature at Level1, the purpose of spatial patches is to rich the representations of images and exploit more features at object-level. Also, all the patches features are clustered into visual words and quantized to another space that are different from the original feature space. Thus, it is not a repeated process to use the global feature again. Moreover, GIS is quite efficient, because all the patches in one image share the same GIS. We store all GIS features in a pre-computed table searched by the indexed items, as seen in Fig. 4.

We compute the similarity of two GIS with the root feature process. Similar to prior work[2, 28], we obtain the root feature by first L_1 normalizing the feature vector and then computing the square root per dimension. The distance is computed with the Hellinger kernel $S(x,y) = \sum_{i=1}^{m} \sqrt[2]{x_i y_i}$. The distance is counted as:

$$d(x,y) = 2 - 2 \cdot S(x,y),$$

Then we take GIS into DeepIndex, and add this distance to update the matching score in Eq.(2). For 1-D DeepIndex, given two patches features x and y, the final matching function becomes:

$$h(x,y) = \delta_{q(x),q(y)} \cdot idf^2 \cdot c(x,y), \quad (4)$$

where $c(x,y) = exp(\alpha \cdot d(gis(x), gis(y)))$, $gis(\cdot)$ returns the corresponding global image feature of the current patch, and α measures the GIS matching strength. For 2-D DeepIndex, there are two feature tuples $\mathcal{X} = [x^r, x^c]$ and $\mathcal{Y} = [y^r, y^c]$. Finally, the 2-D matching function is rewritten as:

$$h(\mathcal{X},\mathcal{Y}) = \delta_{q^r(x^r),q^r(y^r)} \cdot \delta_{q^c(x^c),q^c(y^c)} \cdot idf^2 \cdot c(\mathcal{X},\mathcal{Y}), \quad (5)$$

where $c(\mathcal{X},\mathcal{Y}) = c(gis(x^r), gis(y^r)) \cdot c(gis(x^c), gis(y^c))$. when selecting different fc layers, $gis(\cdot)$ returns the corresponding global deep feature for one image.

In this case, two patches are really matched only when their visual deep-words are identical and their GIS features are similar. Indeed, GIS is a kind of global constraint and compensation for patches matching. A recent and similar work[34] extracts features from the whole image and different levels of images patches as signatures, named deep embedding, which becomes a strong constraint for inverted index built on the SIFT features. Moreover, they try to convert the feature into binary signatures. Different from them, we directly build the visual words and DeepIndex based on deep features and without using SIFT features, and we further exploit multiple DeepIndex to enhance matching accuracy. In addition, we use less image patches(14) than that of 81 patches in[34].

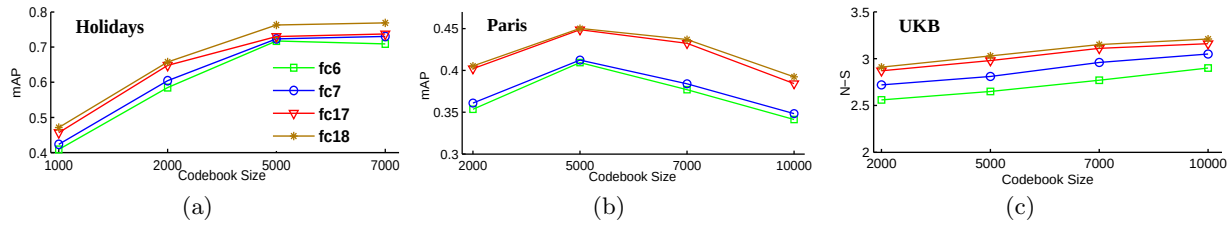

Figure 6: Codebook sizes on three dataset. The selected sizes are 5000, 5000, 10000 on Holidays, Paris, UKB.

Table 1: Ratio between training features and codebook size. Assume that each image has 500 SIFT keypoints.

Dataset	Training features	Codebook size	Ratio
Holidays	991×14	5000	2.8
Paris	6337×14	5000	17.7
UKB	10200×14	10000	14.3
[35] for SIFT	$60K \times 500$	20K	1500
[35] for color	$60K \times 500$	200	150K

5. EXPERIMENTS

In this section,we evaluate the proposed method on three public datasets: Holidays , Paris and UKB.

5.1 Datasets

Holidays[11] contains 1491 vacation photographs corresponding to 500 groups. There are 500 queries, most of which have 1-2 ground truth images which have been rectified to a natural orientation. The performance is measured by mean average precision(mAP)[22] over the provided queries.

Paris[23] has 6412 images obtained from Flickr. 55 images serve as queries. For each image and landmark, one of four possible labels is generated: good, ok, bad, and junk. The mAP is again used as the accuracy measurement.

UKB[21] includes 10200 indoor photos of 2550 objects(4 images per object). Each image is used to query the rest of the dataset in turn. The performance is reported by the average recall of the top four results, referred to as N-S score that is a number between 0 and 4. But some works still measure this dataset with mAP.

5.2 Codebook Generation

In this paper, the visual words are clustered with the training images from every dataset itself. To elevate the efficiency of k-means, we use the algorithm from Fast Library for Approximate Nearest Neighbors(FLANN)[20]. We perform four kinds of 1-D DeepIndex(fc6,fc7,fc17,fc18) to find desirable codebook sizes. The results are drawn in Fig. 6. Considering the accuracy and efficiency, we set codebook sizes $K = 5000, 5000, 10000$ for Holidays, Paris, and UKB.

It is noteworthy that the codebook sizes of deep features are smaller than traditional BoW with SIFT features. Also, the ratios between training deep-features and codebook sizes are relatively smaller. More details in Table 1, the ratios in this paper is under 20, but the ratios in[35] that uses coupled multi-index of SIFT and color features are quite large with about 1500 and 150K. This demonstrates that patches features has desirable discrimination, though there are relatively less training descriptors and smaller clusters.

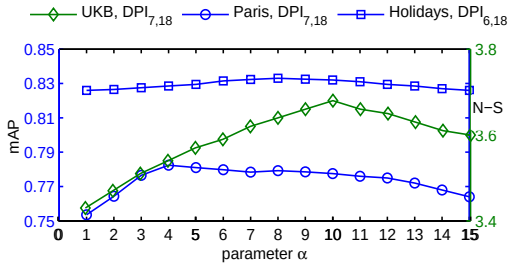

Figure 7: Influence of parameter α.

Table 3: Results with PCA Compression

Datasets	Holidays	Paris	UKB
Dim=4096	0.833	0.7824	3.68
Dim=2048	0.8411	0.7945	3.72
Dim=1024	0.8463	0.8065	3.74
Dim=512	**0.8565**	**0.8124**	**3.76**
Dim=256	0.8367	0.7875	3.71
Dim=128	0.8272	0.7724	3.65

5.3 Evaluation

In this experiment, we compare the performance of 1-D DPI and 2-D DPI in Table 2. It is shown that the 2-D method is not better than the 1-D method using the single multiple assignment(MA=1), due to the low recall. To further improve the recall, we use the multiple assignment(MA) method[13], which improves the 2-D DPI performance.

Furthermore, the inter-CNN methods are better than intra-CNN ones. The reason is that deep features in intra-CNN are from the same CNN architecture, such as fc6 and fc7, so they have close relationship(fc6 is the input of fc7) restraining the ability of 2-D inverted index. Here we call the fc6 and fc7 feature as 'mid-level' descriptions and the fc17 and fc18 features as 'high-level' descriptions because of their deeper network. The result shows that the integration of 'mid-level' and 'high-level' features plays a role in improving the overall representation which results in significant improvement compared with 1-D DPI. Finally, $DPI_{6,18}$ obtains 82.38% mAP on Holidays; $DPI_{7,18}$ has 75.35% mAP on Paris; $DPI_{7,18}$ achieves 3.37 N-S score. In Fig. 8, we show two queries from Holidays and UKB. It can be seen that the 2-D DPI method returns more relevant images than 1-D DPI.

Next, we consider to evaluate the influence of global image signature for 2-D DPI. We choose to test the superiority results on each dataset, as listed in Table 2. The parameter α in GIS ranges from 1 to 15 and the results are shown in Fig. 7. For Holidays, the GIS increases $DPI_{6,18}$ to 83.3% mAP

Table 2: Results with 1-D DeepIndex and 2-D DeepIndex

Dataset	Holidays Dataset			Paris Dataset			UKB Dataset		
Methods	MA=1	MA=3	MA=5	MA=1	MA=5	MA=10	MA=1	MA=5	MA=10
DPI_6	0.7173	0.7354	0.7201	0.4094	0.5689	0.6521	2.90	3.03	3.02
DPI_7	0.7234	0.7490	0.7358	0.4124	0.5745	0.6578	3.05	3.12	3.04
DPI_{17}	0.7302	0.7322	0.7262	0.4487	0.6101	0.7024	3.16	3.19	3.15
DPI_{18}	0.7631	0.7672	0.7563	0.4503	0.6123	0.7133	3.21	3.25	3.19
DPI_{6+7}	0.7200	0.7888	0.7717	0.2935	0.6289	0.7120	3.02	3.13	3.05
DPI_{17+18}	0.7575	0.7996	0.7934	0.3228	0.6329	0.7169	3.16	3.25	3.26
DPI_{7+17}	0.7401	0.8053	0.8020	0.3345	0.6412	0.7324	3.21	3.25	3.19
DPI_{6+17}	0.7332	0.8162	0.8115	0.3395	0.6508	0.7435	3.22	3.26	3.22
DPI_{7+18}	0.7466	0.8123	0.8174	0.3656	0.6618	**0.7535**	3.26	**3.37**	3.32
DPI_{6+18}	0.7382	0.8164	**0.8238**	0.3412	0.6540	0.7452	3.19	3.23	3.29

when α is 8. Similarly, the result of $DPI_{7,18}$ for Paris arrives at 78.24% mAP with $\alpha = 4$. Also, The $DPI_{7,18}$ method gets 3.68 N-S score on UKB with $\alpha = 10$. All these results valid the necessity of GIS that provides a global constraint to enhance the matching strength. Thus, all the following results contain the GIS process.

5.4 Dimensionality reduction

To evaluate the influence of feature compression for deep feature, we conduct PCA compression for the 4096-D deep feature with different dimensions (The GIS is also compressed by PCA). In Table 3, when the dimension is 512, the results obtain most improvement on Holidays(+2.35% mAP), Paris(+3% mAP), UKB(+0.08 score). Although the dimension is down to 128, it can even obtain desirable results compared with many of SIFT-based methods. This implies that PCA affects the performance of deep features less than the ones of SIFT-like methods.

5.5 Comparison with the state-of-the-art

We compare our results with the state-of-the-art methods that are simply divided into three groups: CNN methods, Non-CNN methods and SIFT-CNN methods. Note that we do not consider and perform various post-processing algorithms, such as query expansion, spatial verification, and graph fusion. For CNN methods, we also do not conduct fine-tuning for specific tasks in this work. Thus we compare the results in other methods that exclude the post-processing and fine-tuning stages.

The whole comparison is listed in Table 4. For Holidays, the proposed method(85.56%) exceeds other CNN-based methods, and is competitive to the best results[30] and[34]. In the work by Tolias, *et al.*[30], their representation would take about millions of features per image which is not scalable to large datasets. In Zhang,*et al.*[34], they use both SIFT and CNN features to improve matching precision. For Paris, our result(81.24%) outperforms most other methods, except the recent work that introduces the similarity learning algorithm into deep learning[31]. In UKB, the proposed method(score 3.76) is better than the coupled multiindex method[35], and is competitive to the state-of-the-art approaches[34].

5.6 Complexity analysis

To quantize the efficiency, we compare the computing complexity of the DeepIndex with the state-of-the-art[34] on Holidays, as listed in Table 5. Our experimental environment is CPU i7 at 2.67Ghz with 12GB RAM and NVIDIA Titan Black with 6GB GRAM. Assume that every images has 500 SIFT keypoints. Considering the memory cost per image, both 1-D DPI(2.06KB) and 2-D DPI(4.06KB) are lower than [34], which needs to spend extra memory for numerous SIFT features. In addition, due to the smaller codebook size, the average query time of our approach is shorter with less than 0.5 seconds [2]. This shows the efficiency and feasibility of our method.

Table 4: Comparison results with the state-of-the-art on three datasets. * means the improved results in the papers.

Groups	Methods	Holidays	Paris	UKB
CNN	[4]	74.70	-	3.43
CNN	[28]	79.00	-	3.61
CNN	[10]	80.20	-	-
CNN	[24]	84.30	79.50	-(91.1)
CNN	[31]	-	**86.83**	-
CNN	Ours	85.65	81.24	3.76
Non-CNN	[32]	78.90	-	3.50
Non-CNN	[33]	80.86	-	3.60
Non-CNN	[13]	81.30	-	3.42(87.8)
Non-CNN	[30]	82.20	78.20	-
Non-CNN	[12]	83.90	-	3.54(90.7)
Non-CNN	[35]	84.02	-	3.71(94.7)
Non-CNN	[30]*	88.00	80.50	-
SIFT-CNN	[34]	85.30	-	3.79
SIFT-CNN	[34]*	**88.08**	-	**3.85**

Table 5: Memory cost (bytes) and query time (seconds) for one image on Holidays dataset

Methods	[34]	1-D DPI	2-D DPI
ImageID	4×500	4×14	4×14
Signature	10.18KB	512×4	$512 \times 4 \times 2$
Total Memory	12.13KB	2.06KB	4.06KB
Query Time	2.32	0.25	0.45

6. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed the DeepIndex framework for accurate and efficient image retrieval that introduced the inverted index into deep features. Moreover, we integrated multiple deep features with the multiple DeepIndex which attempted to bridge the gap between mid-level and high-level representations. Experimental results showed that our method achieved competitive performance in the well-known and frequently benchmarked Holidays, Paris, UKB datasets.

[2]Our query time does not include the feature extraction.

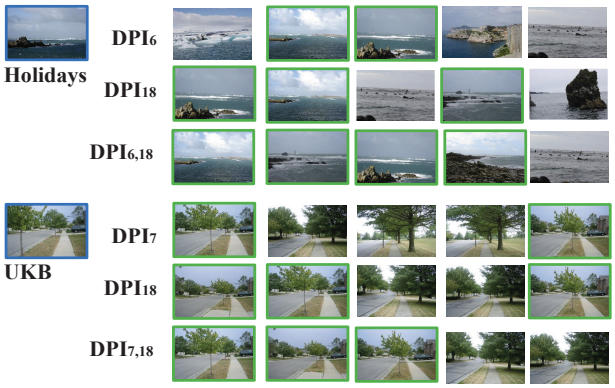

Figure 8: Retrieval results in Holidays and UKB datasets. The left image with blue frame is query one, and the images with green frame are right results. Best viewed in color.

In the future, we think it is promising to investigate the possibilities of using posting-processing or finetuning stage within the DeepIndex approach. Codes and pre-computed data are released on our website[3].

7. ACKNOWLEDGMENTS

This work was supported by the LIACS Media Lab at Leiden University and the China Scholarship Council(CSC).

8. REFERENCES

[1] P. Agrawal, R. Girshick, and J. Malik. Analyzing the performance of multilayer neural networks for object recognition. In *ECCV*, 2014.

[2] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

[3] A. Babenko and V. S. Lempitsky. The inverted multi-index. In *CVPR*, 2012.

[4] A. Babenko, A. Slesarev, A. Chigorin, and V. S. Lempitsky. Neural codes for image retrieval. In *ECCV*, 2014.

[5] Y. Bengio, A. C. Courville, and P. Vincent. Unsupervised Feature Learning and Deep Learning: A Review and New Perspectives. *CoRR*, abs/1206.5538, 2012.

[6] K. Chatfield, R. Arandjelović, O. M. Parkhi, and A. Zisserman. On-the-fly learning for visual search of large-scale image and video datasets. *IJMIR*, 4(2), 2015.

[7] L. Cun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Handwritten digit recognition with a back-propagation network. In *NIPS*, 1990.

[8] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *ICML*, 2014.

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

[10] Y. Gong, L. Wang, R. Guo, and S. Lazebnik. Multi-scale orderless pooling of deep convolutional activation features. In *ECCV*, 2014.

[11] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. In *ECCV*, 2008.

[12] H. Jégou, M. Douze, and C. Schmid. On the burstiness of visual elements. In *CVPR*, 2009.

[3]press.liacs.nl/lml/deepindex.

[13] H. Jégou, M. Douze, and C. Schmid. Improving bag-of-features for large scale image search. *IJCV*, 87(3):316–336, 2010.

[14] H. Jégou, M. Douze, C. Schmid, and P. Pérez. Aggregating local descriptors into a compact image representation. In *CVPR*, 2010.

[15] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional Architecture for Fast Feature Embedding. In *ACM Multimedia*, 2014.

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR*, 2006.

[18] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *TOMCCAP*, 2(1):1–19, 2006.

[19] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[20] M. Muja and D. G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application*, 2009.

[21] D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.

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

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

[24] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. In *CVPR, DeepVision workshop*, 2014.

[25] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *arXiv:1409.0575*, 2014.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[27] J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.

[28] S. Sun, W. Zhou, H. Li, and Q. Tian. Search by detection: Object-level feature for image retrieval. In *ICIMCS*, 2014.

[29] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[30] G. Tolias, Y. Avrithis, and H. Jégou. To aggregate or not to aggregate: selective match kernels for image search. In *ICCV*, 2013.

[31] J. Wan, D. Wang, S. C. H. Hoi, P. Wu, J. Zhu, Y. Zhang, and J. Li. Deep learning for content-based image retrieval: A comprehensive study. In *ACM Multimedia*, 2014.

[32] C. Wengert, M. Douze, and H. Jégou. Bag-of-colors for improved image search. In *ACM Multimedia*, 2011.

[33] M. Yang, X. Wang, Y. Lin, and Q. Tian. Semantic-aware co-indexing for near-duplicate image retrieval. In *ICCV*, 2014.

[34] L. Zheng, S. Wang, F. He, and Q. Tian. Seeing the big picture: Deep embedding with contextual evidences. *CoRR*, abs/1406.0132, 2014.

[35] L. Zheng, S. Wang, Z. Liu, and Q. Tian. Packing and padding: Coupled multi-index for accurate image retrieval. In *CVPR*, 2014.