Exploring images with deep learning for classification, retrieval and synthesis

Liu, Y.

Chapter 4

DeepIndex for Image Retrieval

The previous two research chapters focused on research about image-level and pixel-level classification. In this chapter, we turn our focus on the second research theme: *retrieval*, and answer how we can utilize deep visual representations for accurate and efficient image retrieval (RQ 3).

In a conventional image retrieval system, a number of local features are designed to describe key points in images. Then the well-known Bag-of-Words model is used to quantize the local features into visual words. In addition, an inverted index scheme is created to reduce the computational burden and query time. However, the local features are weak to distill high-level semantic concepts from the images. In the past few years, deep visual representations have shown powerful capabilities of bridging the semantic gap between low-level and high-level features. Inspired by this, in this chapter we exploit a DeepIndex framework for accurate and efficient image retrieval, by incorporating deep visual features into the inverted index scheme. DeepIndex can take advantage of the powerful discrimination of deep features and the fast search of the inverted index. To integrate more deep features, we further extend our framework to be a multiple DeepIndex. We find that the multiple DeepIndex can be viewed as a good attempt to couple different deep features. Extensive experiments on three benchmarks demonstrate the effectiveness of the proposed method. Our method is efficient in terms of memory cost and query time.

Keywords

Image retrieval, Convolutional neural networks, Bag of words, Inverted index

4.1 Introduction

Image retrieval is a practical and common application in the real world and therefore has triggered a massive amount of research activities in both multimedia and computer vision fields [19, 38, 39]. Bag-of-Words (BoW) is a traditional and efficient method in existing image retrieval systems, where local features, such as the SIFT [40] and color clues [41], are quantized to visual words with a pre-trained codebook. Then, similar to document retrieval [19, 39], an inverted index is built with the visual words to reduce computational and memory cost for scalable image search. Recently, Zheng *et al.* [164] performed low-level feature fusion with the SIFT and color features using a coupled inverted index framework. However, image retrieval remains challenging due to the well-known semantic gap between low-level image representations and high-level semantic concepts.

To bridge the semantic gap, recent works are dedicated to using more discriminative visual features learned in deep neural networks. The work of Wan *et al.* [42] finds that a deep CNN model pre-trained on a large dataset can be transferred for new content-based image retrieval (CBIR) tasks and that similarity learning can further boost the retrieval performance. Babenko *et al.* [45] focus on holistic descriptors where the whole image is mapped to a single deep feature vector. To extract richer regional features, Gong *et al.* [22] employed image patches at multiple scales, and then aggregated local patch responses at the finer scales via the VLAD [20] encoding. Yoo *et al.* [25] utilized multi-scale dense local CNN features to compute the Fisher Vector kernels. Zhang *et al.* [46] proposed a deep embedding method by incorporating the SIFT descriptor and CNN features. However, these prior works mainly focus on the accuracy and omit the importance of the retrieval efficiency, including memory cost and query time. To ensure both the accuracy and efficiency for image retrieval, we need to answer the question **RQ 3: How can we incorporate deep visual representations into the inverted index structure for accurate and efficient image retrieval?**

In this chapter, we propose a novel DeepIndex framework for accurate and efficient image retrieval, which can incorporate deep features into the inverted index scheme. We present a Bag-of-Deep-Features(BDF) model to cluster deep features into a number of visual words. In contrast to prior works that use resource-consuming algorithms for matching deep features, our DeepIndex(DPI) employs an efficient inverted index for fast image search, which can achieve competitive performance while reducing the computational time and memory cost. Furthermore, we extend DeepIndex by integrating different deep features and build a 2-D DeepIndex structure that consists of two kinds of variants: intra-CNN and inter-CNN. Intra-CNN uses two deep features from the same CNN architecture (*e.g.* AlexNet [157]), while inter-CNN selects the features from two different CNN architectures (*e.g.* AlexNet [157] and VGG [7]). The performance of inter-CNN is better than that of intra-CNN,

because the former one can fuse mutual deep features learned by different CNN models. However, intra-CNN is simpler and faster than inter-CNN. Notably, both intra-CNN and inter-CNN can serve as a solution to couple different deep features at an indexing level. Last but not least, we introduce a global image signature (GIS) into DeepIndex in order to enhance the query accuracy. In the experiments, we evaluate the proposed method on three datasets, where the results demonstrate its effectiveness and efficiency. Also, our results can compete with the state-of-the-art performance in terms of retrieval accuracy and computational cost.

The contributions of this work are as follows:

- We present a good attempt to incorporate deep features into the inverted index scheme and exploit a novel DeepIndex framework for accurate and efficient image retrieval.
- We present a 2-D DeepIndex variant that can be an alternative to effectively integrate different deep features at an indexing level.
- Our experiments show the promising advantages of leveraging deep visual representations to improve traditional image retrieval methods.

The rest of this chapter is structured as follows. Section 4.2 describes the bag-of-deep-features method. The DeepIndex framework is introduced in Section 4.3. The experimental results are reported in Section 4.4. Finally, Section 4.5 summarizes the conclusions.

4.2 Bag of Deep Features

Traditional image retrieval methods extract low-level features from images, such as SIFT and color descriptors, and employ them to construct the Bag-of-Features (BoF) or Bag-of-Words (BoW) model. However, few works have shown the utility of deep features into BoF. In this section, we present a Bag-of-Deep-Features (BDF) model, where visual words can be clustered based on CNN features.

4.2.1 Spatial patches

Generally, extracting only the global image feature is not discriminative enough for image retrieval, and may miss some local clues, such as spatial locations and contexts. Thus it is encouraged to extract rich regional features within finer scales. There are three common approaches to search for local regions in an image, including sliding window, region proposal and spatial pyramid.

Firstly, the sliding window approach is a quite common approach in object recognition and object detection that scans an image using windows of different scales,

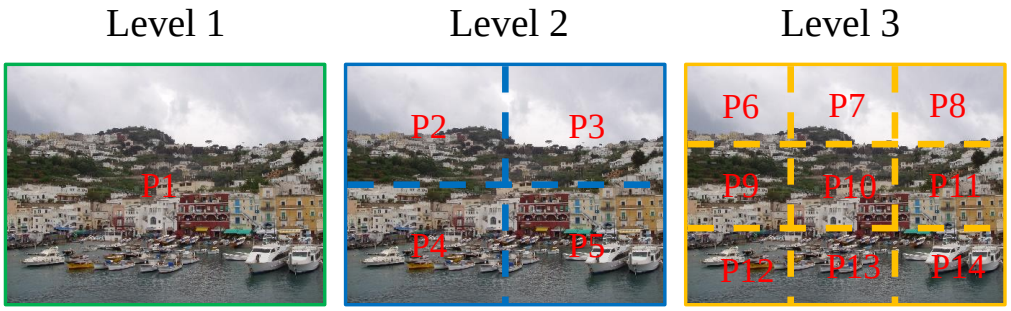

Figure 4.1: A three-level spatial pyramid. There are 14 image patches P_i in total, $i = 1, \ldots, 14$. we can describe each patch with a CNN feature.

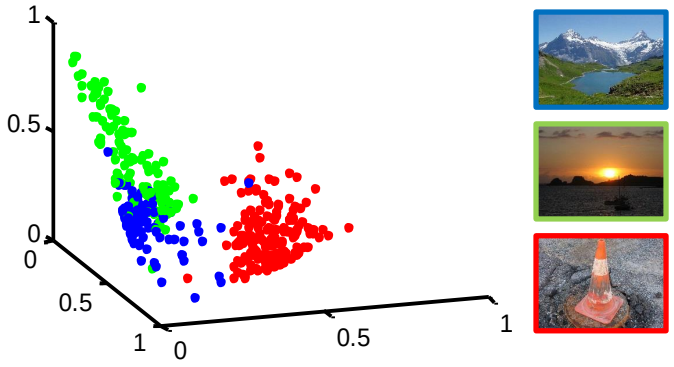

Figure 4.2: Visualizing the CNN features for three groups of images from the Holidays dataset [47]. Note that, each group has a few images and we show one of them in the right side. Each point in the 3D space represents an image patch, and its color corresponds to one of the three groups. We can see the clear separations of different groups.

locations, and aspect ratios. For example, Gong *et al.* [22] scan the whole image with two levels of overlapping windows that generates numerous local patches. Secondly, the region proposal approach can detect the objects of interest in images using fewer candidates than sliding windows. For object detection, RCNN [165] adopts the selective search into CNNs replacing sliding windows. Sun *et al.* [166] extract CNN features for object-like image patches with a region proposal detector. Thirdly, in contrast to the above two methods, the spatial pyramid approach [167] is an efficient way to preserve the spatial information in the images. Razavian *et al.* [6], 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.

Considering the above three approaches, we employ the spatial pyramid one to enrich the image representation because of its simplicity and efficiency. As seen in Figure 4.1, we partition one image into three levels: Level 1 contains only one patch P_1 that is the whole image; Level 2 divides the image into four patches (P_2, P_3, P_4, P_5) whose union covers the whole image; Level 3 consists of nine non-overlapping patches, denoted with P_6 to P_{14}. In total, there are 14 patches for one image, and their CNN features can be computed independently. In contrast to [6] which uses larger levels for training images than query images, we apply the same spatial levels for all images in the training and testing sets.

4.2.2 Feature extraction and quantization

The success of convolutional neural networks in image classification[157] has shown the strong efficiency and discriminative ability of learning deep visual features. It is common to extract the image representation from the fully connected (fc) layers [6, 168], because they are closer to class posteriors. However, it is still questionable about which fc layer is favorable for image retrieval. Different from prior works using either the first or the second fc feature, we aim to study the benefit of aggregating multiple fc features.

Specifically, we employ two common CNN architectures pre-trained on ImageNet [5]. The first one is AlexNet proposed by Krizhevsky *et al.* [157] in 2012. The second one is the VGG-19 model from Simonyan *et al.* [169] proposed in 2014. AlexNet has eight successive layers (5 convolutional layers, 3 fully connected layers), and the first and second fc layers are named by fc6 and fc7 respectively. VGG-19 consists of 16 convolutional layers and 3 fully connected layers and we name the first and second fc layers fc17 and fc18. To visually demonstrate deep features, we select three groups of images from the Holidays dataset[47]. The fc18 features of the image patches are extracted (4096-dimension). Then we map the features into a 3D space by the classical Multi-Dimensional Scaling (MDS). As seen in Figure 4.2, the separability of three groups is clear in the 3D space.

After feature extraction, we perform feature quantization based on the BoW model. Given an image I, x_i represents the feature vector of the i-th patch. After extracting the features of all image patches, we can learn a codebook with the k-means algorithm. Then the quantization function $q(\cdot)$ is used to map a patch feature x_i to its nearest visual word v_k in the codebook, *i.e.* $q(x_i) \mapsto v_k$. Note that, the codebooks associated with different fc features (*i.e.* fc6, fc7, fc17, fc18) are constructed independently. L_2 normalization is used to normalize the features.

4.3 DeepIndex

To reduce the retrieval time and memory cost, we propose the DeepIndex framework in which the inverted index is created based on the visual words. In addition, we can integrate multiple deep features with a multiple DeepIndex variant. Finally, the global image signature is utilized to increase the matching accuracy.

4.3.1 Single DeepIndex

We create an inverted index structure in which each entry corresponds to a visual word defined by the pre-computed codebook $\{v_i\}_{i=1}^{K}$. We represent the inverted index as $\mathcal{W} = \{W_1, W_2, \ldots, W_K\}$, where each entry W_i consists of a list

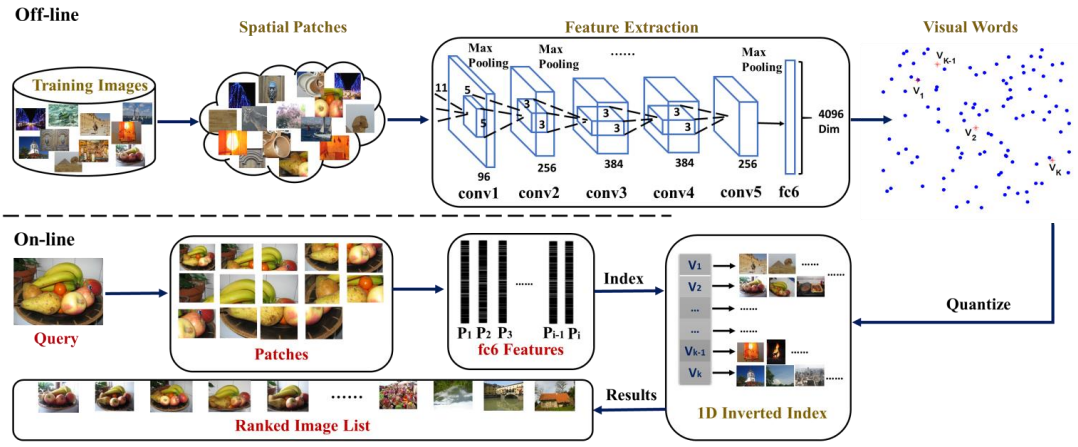

Figure 4.3: The flowchart of Single DeepIndex framework, including off-line and on-line stages. Here, the fc6 features from AlexNet are extracted to cluster the visual words, which are used to construct the inverted index structure. More details are in Section 4.3.

of indexed items, such as image ID, term-frequency (TF) score and other meta-data [39, 164, 170]. The indexed items following each entry W_i are counted as the retrieved candidates of the query feature. Therefore, the matching function $h_q(\cdot)$ for two deep features x and y can be expressed with

$$h_q(x, y) = \delta_{q(x),q(y)}, \tag{4.1}$$

where δ is the Kronecker delta response and $q(\cdot) \in [1, K]$. However, this matching function cannot weight the visual words according to their frequency. Generally, rare visual words are assumed to be more discriminative and should be assigned higher weights. Driven by the *tf-idf* scheme[19], we update the matching function

$$h(x, y) = \delta_{q(x),q(y)} \cdot idf(q(y))^2, \tag{4.2}$$

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

For simplicity, we call the proposed indexing scheme single DeepIndex (1-D DPI) because it uses one kind of deep features. Specifically, we present four variants of 1-D DPI, including DPI_6,DPI_7,DPI_{17} and DPI_{18}, depending on which fully-connected feature of AlexNet and VGG-19 is used. The entire procedure of DeepIndex for image retrieval is illustrated in Figure 4.3. It takes the DPI_6 method as an example, but it is suitable for other deep features as well. There are two stages: off-line stage and on-line stage. During the off-line stage, we mainly cluster the codebook with the training image patches, and construct the inverted index structure. The on-line stage will query an image with its patch features and obtain similar images by searching the inverted index structure.

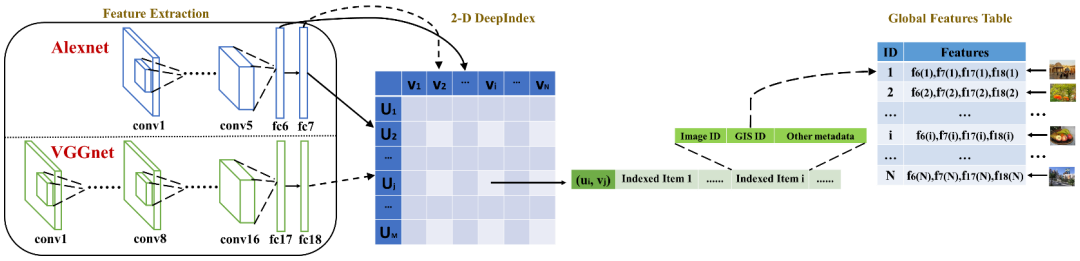

Figure 4.4: The framework of 2-D DeepIndex with deep features, including intra-CNN and inter-CNN. For intra-CNN, it uses the fc6 and fc7 jointly. For inter-CNN, the fc7 and fc18 are incorporated for indexing. Besides, the global image signature, being an additional clue in the indexed items, is stored in a table.

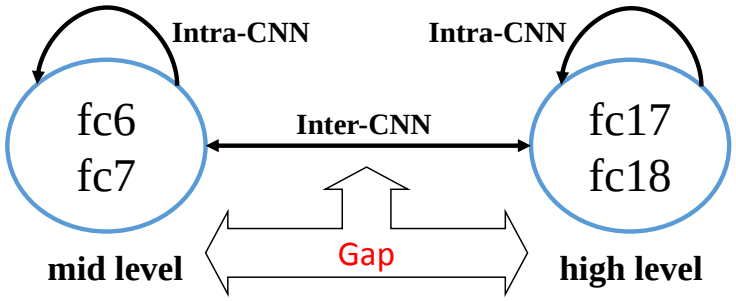

Figure 4.5: A conceptual comparison between Intra-CNN and Inter-CNN.

4.3.2 Multiple DeepIndex

Currently, most works mainly focus on comparing performance of different fully connected layers and choose a superior one. However, different neural layers imply different levels of abstraction of the image. Thus we utilize different deep features to compensate each other and to improve the retrieval accuracy. Based on this idea, we present an extended framework, called multiple DeepIndex (multi-DPI). The multi-index structure was first proposed in Babenko *et al.* [171]. It decomposes the SIFT descriptor into several blocks by product quantization. The multi-index structure is then organized around the codebooks of corresponding blocks. Similarly, Zheng *et al.* [164] built the coupled multi-index with traditional SIFT features and additional discriminative color names. Their results demonstrate that the feature fusion at the indexing level is better than the single indexing. Motivated by these works, we exploit a multiple DeepIndex that can incorporate multiple deep features into a multi-index structure. In this work, we take the two dimensional DeepIndex (2-D DPI) as an example.

To be specific, we denote $\mathcal{X} = [x^r, x^c]$ as a coupled deep features for a patch P_i, where x^r is extracted from one fc layer as the row indexing, and x^c comes from another fc layer as the column indexing. Then, two codebooks are pre-computed with different fc features separately, *i.e.* $\mathcal{U} = u_1, u_2, \ldots, u_M$ and $\mathcal{V} = v_1, v_2, \ldots, v_N$, where M and N are codebook sizes. The 2-D DPI structure contains $M \times N$

entries, where $\mathcal{W} = W_{11}, W_{12}, \ldots, W_{ij}, \ldots, W_{MN}, i = 1, 2, \ldots, M, j = 1, 2, \ldots, N$. After building the 2-D DPI, each feature tuple like $\mathcal{X} = [x^r, x^c]$ can be quantized into a visual word pair (u_i, v_j) based on the codebooks $\mathcal{U}$ and $\mathcal{V}$, where u_i and v_j are the nearest centroids to x^r and x^c, respectively. Similar to the 1-D DPI, additional clues (*e.g.* image ID and other meta-data) related to the feature tuple $\mathcal{X}$ are saved in the corresponding entry W_{ij}.

Given two feature tuples $\mathcal{X} = [x^r, x^c]$ and $\mathcal{Y} = [y^r, y^c]$, the matching function for 2-D indexing can be rewritten by

$$h(\mathcal{X}, \mathcal{Y}) = \delta_{q^r(x^r), q^r(y^r)} \cdot \delta_{q^c(x^c), q^c(y^c)} \cdot idf^2, \tag{4.3}$$

where $q^r(\cdot)$ and $q^c(\cdot)$ present two different quantization functions. Notice that, a right match is valid only if the two features tuples are similar in both row and column indexing. In this way, the 2-D DeepIndex can enhance the matching strength so as to improve the retrieval accuracy.

Moreover, we define two methods for selecting fc features, named **intra-CNN** and **inter-CNN**. (1) The Intra-CNN method uses two fc layers from the same CNN architecture. As the two black solid lines seen in Figure 4.4, fc6 activation is taken as column indexing, and the fc7 activation serves as row indexing. We can construct two Intra-CNN members: $\text{DPI}_{6,7}$ and $\text{DPI}_{17,18}$. (2) The Inter-CNN method chooses two fc layers coming from two different CNN architectures. For example, the two black dash lines in Figure 4.4, fc7 in Alexnet and fc18 in VGG-19 can serve as column and row indexing respectively. In total, we can have four Inter-CNN members, including $\text{DPI}_{6,17}$, $\text{DPI}_{7,17}$, $\text{DPI}_{6,18}$ and $\text{DPI}_{7,18}$.

We provide more insights into Intra-CNN and Inter-CNN in Figure 4.5. By comparing the depth of AlexNet and VGG-19, we categorize fc6 and fc7 as mid-level features, and fc17 and fc18 as high-level features. Intra-CNN is simpler to build than Inter-CNN. However, Inter-CNN can be viewed as a solution to bridge the gap between mid-level and high-level deep networks. More comparison and analysis about intra-CNN and inter-CNN is reported in the experiments.

4.3.3 Global image signature

To further improve the matching accuracy in the inverted index structure, we employ an additional discriminative feature to constrain the matching condition and filter out false matches, which is called the 'signature'. The most popular one is the hamming embedding signature [47] that uses a 64-D binary signature for each SIFT descriptor, and stores it in the meta-data of the inverted items. Also, Zheng *et al.* [164] use the hamming embedding for SIFT features and generate another signature for color names. The discrimination of deep image features has been demonstrated

in existing works [6, 42, 172], for example, only one fc feature vector extracted from the whole image can achieve desirable results for many tasks.

In this work, we propose to use this global deep feature as an additional signature for DeepIndex, called global image signature(GIS). Although the spatial patches already contains global feature representation at Level 1, they are used to enrich the representations of images and exploit more features at local regions. In addition, all the patch features are clustered into visual words and quantized to another space that is different from the original feature space. Thus, it is not a redundant process to use the global feature again. Since GIS is quite efficient, all the patches in one image can share the same GIS. We store all GIS features in a Global Features Table and search them by the GIS ID stored in the indexed items, as seen in Figure 4.4.

We compute the similarity of two GIS features with the root feature process [166, 173]. Specifically, we obtain the root feature by first L_1 normalizing the feature vector and then computing the square root per dimension. The distance $d(x, y)$ is computed using the Hellinger kernel $S(x, y) = \sum_{i=1}^{m} \sqrt[2]{x_i y_i}$:

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

Then we take GIS into DeepIndex, and add this distance to update the matching score. For 1-D DeepIndex, given two patch features x and y, we can update Eq. 4.2

$$h(x, y) = \delta_{q(x),q(y)} \cdot idf^2 \cdot c(x, y), \tag{4.4}$$

where $c(x, y) = exp(\alpha \cdot d(gis(x), gis(y)))$; $gis(\cdot)$ returns the corresponding global image feature of the current image patch; α adjusts the GIS matching strength. For 2-D DeepIndex, there are two feature tuples, its matching function becomes

$$h(\mathcal{X}, \mathcal{Y}) = \delta_{q^r(x^r),q^r(y^r)} \cdot \delta_{q^c(x^c),q^c(y^c)} \cdot idf^2 \cdot c(\mathcal{X}, \mathcal{Y}), \tag{4.5}$$

where $c(\mathcal{X}, \mathcal{Y}) = c(gis(x^r), gis(y^r)) \cdot c(gis(x^c), gis(y^c))$. We find that GIS is an efficient global constraint and can compensate for patch matching. Finally, two patches can be matched only when their visual words are identical and their GIS features are similar.

4.4 Experiments

We evaluate the proposed method on three datasets and conduct component analysis to verify its effectiveness. In addition, we present its computational complexity in terms of memory cost and query time.

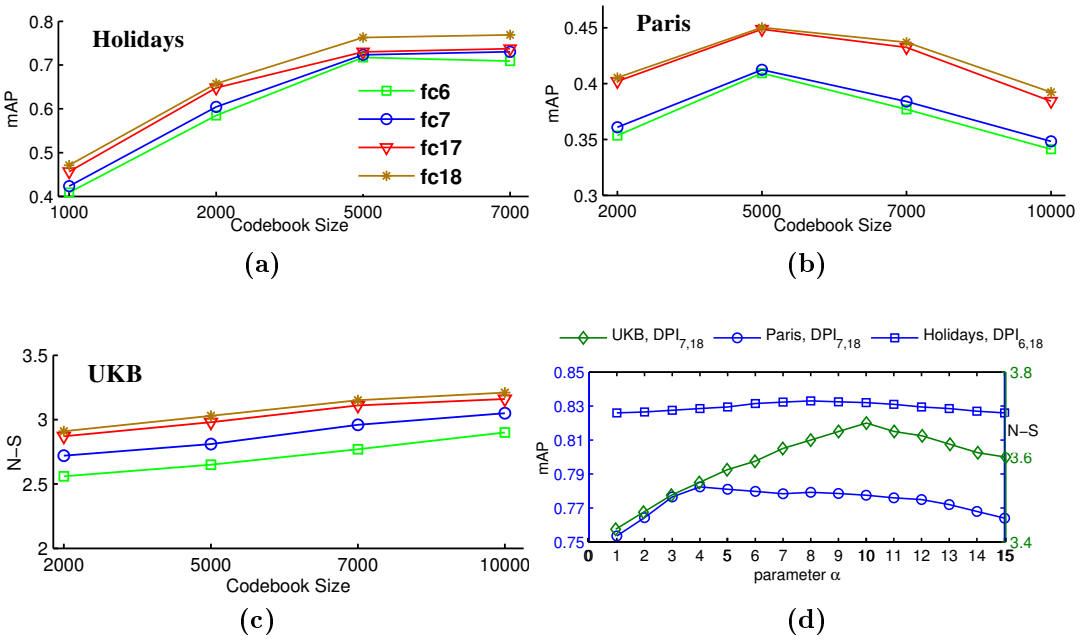

Figure 4.6: (a)-(c) Effect of codebook sizes on three dataset. The selected sizes are 5000, 5000 and 10000 for Holidays, Paris and UKB, respectively. (d) Influence of parameter α.

4.4.1 Datasets and metrics

Holidays [47] contains 1,491 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) over the provided queries (also seen in Section 2.4.3).

Paris [174] has 6,412 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 [175] includes 10,200 indoor photos of 2,550 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 (also seen in Section 2.4.3) . But some works still use mAP to measure the performance on this dataset.

4.4.2 Results and discussion

Codebook size.

The visual words are clustered using features from the training images. To elevate the efficiency of k-means, we use the algorithm from Fast Library for Approximate

Table 4.1: Quantitative results on the 1-D DeepIndex and 2-D DeepIndex. Multiple assignment (MA) is used to increase the retrieval recall. We compare the performance of four 1-D DPI methods, two Intra-CNN methods and four Inter-CNN methods. The best results on the datasets are in boldface.

Method	Holidays (mAP)			Paris (mAP)			UKB (N-S)		
	MA=1	MA=3	MA=5	MA=1	MA=5	MA=10	MA=1	MA=5	MA=10
DPI_6	71.73	73.54	72.01	40.94	56.89	65.21	2.90	3.03	3.02
DPI_7	72.34	74.90	73.58	41.24	57.45	65.78	3.05	3.12	3.04
DPI_{17}	73.02	73.22	72.62	44.87	61.01	70.24	3.16	3.19	3.15
DPI_{18}	76.31	76.72	75.63	45.03	61.23	71.33	3.21	3.25	3.19
DPI_{6+7}	72.00	78.88	77.17	29.35	62.89	71.20	3.02	3.13	3.05
DPI_{17+18}	75.75	79.96	79.34	32.28	63.29	71.69	3.16	3.25	3.26
DPI_{7+17}	74.01	80.53	80.20	33.45	64.12	73.24	3.21	3.25	3.19
DPI_{6+17}	73.32	81.62	81.15	33.95	65.08	74.35	3.22	3.26	3.22
DPI_{7+18}	74.66	81.23	81.74	36.56	66.18	**75.35**	3.26	**3.37**	3.32
DPI_{6+18}	73.82	81.64	**82.38**	34.12	65.40	74.52	3.19	3.23	3.29

Nearest Neighbors(FLANN) [176]. We test four kinds of 1-D DeepIndex (*i.e.* DPI_6, DPI_7, DPI_{17} and DPI_{18}) to find proper codebook sizes. The results are shown in Figure 4.6. To balance the accuracy and efficiency, we set the codebook size K =5000, 5000 and 10000 for Holidays, Paris and UKB, respectively. It is noteworthy that the codebook sizes of deep features are much smaller than traditional BoW with local features, because the number of image patches is much smaller than the number of key points.

Ablation study of DeepIndex

We report ablation results of 1-D DPI and 2-D DPI on the three datasets in Table 4.1. First, we analyze the effect of multiple assignment (MA) [170] on the performance, which is a common technique when retrieving the inverted index items. When MA=1, it means that only the nearest inverted index item can be retrieved. In this case, we can see that the 2-D method is not better than the 1-D method because of the low recall. To further improve the recall, we can increase the multiple assignment (MA) [170]. In this way, the 2-D DPI can perform better than the 1-D DPI, which demonstrates the benefit of integrating different features.

Next, we can observe that the inter-CNN methods are better than the intra-CNN ones. The reason is that two deep features in intra-CNN are from the same CNN architecture, such as fc6 and fc7 in AlexNet, and their implicit relationships (*i.e.* fc6 is the input of fc7) may limit the learning of the 2-D inverted index. For simplicity, we call the fc6 and fc7 features as 'mid-level' descriptions and the fc17 and fc18 features as 'high-level' descriptions. As a result of the mutual compensation of mid-level and high-level features in inter-CNN, it can bridge the gap between different CNNs at the 2-D inverted index level and achieve superior retrieval accuracy. In details, $DPI_{6,18}$ obtains 82.38% mAP on Holidays; $DPI_{7,18}$ has 75.35% mAP on Paris; $DPI_{7,18}$ achieves 3.37 N-S score.

Figure 4.7: Retrieval results on the Holidays and UKB datasets. The 2-D DPI method can have more relevant retrieved candidates than the 1-D DPI.

Table 4.2: Effect of PCA Compression on the performance of DeepIndex.

Dimensions	Holidays (mAP)	Paris (mAP)	UKB (N-S)
4096	83.30	78.24	3.68
2048	84.11	79.45	3.72
1024	84.63	80.65	3.74
512	**85.65**	**81.24**	**3.76**
256	83.67	78.75	3.71
128	82.72	77.24	3.65

Moreover, we study the influence of the global image signature on 2-D DPI. We choose to test the superior methods on each dataset, as listed in Table 4.1. The parameter α in GIS ranges from 1 to 15 and the results are shown in Figure 4.6d. For Holidays, the GIS increases $\text{DPI}_{6,18}$ to 83.3% mAP when α is 8. Similarly, the result of $\text{DPI}_{7,18}$ for Paris reaches 78.24% mAP with $\alpha = 4$. Also, the $\text{DPI}_{7,18}$ method gets 3.68 N-S score on UKB with $\alpha = 10$. All these results show that GIS can help in providing a global constraint to enhance the matching accuracy. All the following results contain the GIS process. In addition to the quantitative evaluation, we show two queries from Holidays and UKB in Figure 4.7. It can be seen that the 2-D DPI method can retrieve more relevant images than the 1-D DPI.

Dimensionality reduction

The deep visual features we use have 4096 dimensions. To reduce the feature dimensionality, we further study the influence of feature compression for deep features. Specifically, we conduct PCA compression for the 4096-Dimension deep features. Also, the GIS is compressed by PCA. We report the results in Table 4.2. Interestingly, when the dimension decreases to 512, we achieve the best results on all the

Table 4.3: Comparison results with other methods on three datasets.

Groups	Methods	Holidays (mAP)	Paris (mAP)	UKB (N-S and mAP)
Non-CNN	[41]	78.90	-	3.50
Non-CNN	[177]	80.86	-	3.60
Non-CNN	[170]	81.30	-	3.42(87.8)
Non-CNN	[178]	82.20	78.20	-
Non-CNN	[179]	83.90	-	3.54(90.7)
Non-CNN	[164]	84.02	-	3.71(94.7)
Non-CNN	[178]	88.00	80.50	-
CNN	[45]	74.70	-	3.43
CNN	[166]	79.00	-	3.61
CNN	[22]	80.20	-	-
CNN	[6]	84.30	79.50	-(91.1)
CNN	[42]	-	**86.83**	-
CNN	Ours	85.65	81.24	3.76
SIFT-CNN	[46]	85.30	-	3.79
SIFT-CNN	[46]	**88.08**	-	**3.85**

three datasets. Even in the extreme case where the dimensionality is down to 128, it can still obtain desirable results compared with many of SIFT-based methods. This implies that the original deep feature is discriminative while containing some redundant information for the retrieval tasks. Feature compression can help refine the feature representation and maintain the high performance. A similar observation is also suggested in other related works [42, 45].

4.4.3 Comparison with other methods

We compare our results with other state-of-the-art methods. We simply divide them into three groups: CNN methods, Non-CNN methods and SIFT-CNN methods. We do not consider and perform various post-processing algorithms, such as query expansion, spatial verification and graph fusion. For CNN methods, we do not consider fine-tuning for specific tasks. For fairness, we compare the results with other methods that exclude the post-processing and fine-tuning steps.

The whole comparison is listed in Table 4.3. For Holidays, our proposed method (85.56%) exceeds other CNN-based methods, and is in competition with the best results [178] and [46]. In the work by Tolias *et al.* [178], their representation takes several millions of features per image which is not scalable to large datasets. In Zhang *et al.* [46], they use both the SIFT descriptor and CNN features to increase the accuracy. On the Paris dataset, our result (81.24%) outperforms most methods, except [42] that introduces the similarity learning algorithm into deep learning. In UKB, our method (3.76) is better than the coupled multi-index method [164], and

Table 4.4: Memory cost (bytes) and query time (seconds) for one image on Holidays.

Complexity	[46]	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

is also competitive with [46].

Complexity analysis

Although our results are inferior to those of [46], our method is more efficient in terms of memory cost and query time. As seen in Table 4.4, we compare the computing complexity of DeepIndex with [46] on Holidays. Our experimental environment is Intel i7 CPU at 2.67Ghz with 12GB RAM and NVIDIA GTX 660 with 2GB GRAM. Zheng *et al.* [46] extracts 500 SIFT keypoints for each image. Considering the memory cost per image, both the 1-D DPI (2.06KB) and 2-D DPI (4.06KB) are more efficient than [46] that requires significantly more memory for the SIFT descriptors. Also, our average query time is shorter, *i.e.* less than 0.5 seconds compared to 2.3 seconds for [46]. These results are consistent with our motivation of exploiting an accurate and efficient image retrieval method.

4.5 Chapter Conclusions

In this chapter, we exploited the DeepIndex framework for accurate and efficient image retrieval that could incorporate deep features into the inverted index scheme. In addition, we integrated multiple deep features with the multiple DeepIndex which was able to bridge different deep representations at an indexing level. Experimental results showed that our method achieved competitive performance on the Holidays, Paris and UKB datasets, while retaining the retrieval efficiency in terms of memory cost and query time.

Future work. One the one hand, a straightforward improvement is to further extend multiple DeepIndex by using more deep features, *e.g.* 3-D DeepIndex and so on. But we should note that it will increase the computational cost. On the other hand, it is encouraged to integrate some traditional retrieval techniques with DeepIndex, such as query expansion and late fusion. We believe that deep learning approaches would be compatible with other traditional algorithms.